Digilent Parallel Transfer Interface (DPTI)

Revision: June 1, 2016

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This document describes the operation and specifications of the Digilent Parallel Transfer Interface (DPTI). This interface is implemented in various Digilent products to provide a high-bandwidth communication and data transfer interface between a personal computer and a Digilent programmable logic system board. Check the board reference manual to see if DPTI is supported on the board in question.

DPTI is the sum of two interfaces: the application programming interface (API) on the PC side and the hardware interface on the Digilent system board. The API is part of the Adept 2 SDK and is readily available to the user. The hardware interface on the other hand requires that certain logic be implemented in the FPGA/CPLD on the Digilent system board to provide the peripheral side of the interface. This document describes the requirements of the hardware interface. For more details on the API see the DPTI Programmer's Reference Manual.

Terms

"Host" – PC running the application calling the DPTI API.
"Peripheral" or "Device" – Digilent FPGA system board configured with a design implementing the Digilent Parallel Transfer Interface (DPTI).
"Upload" - device-to-host transfer.
"Download" - host-to-device transfer.
"Read transfer" – FPGA reading data over DPTI
"Write transfer" – FPGA writing data over DPTI

Functional Description

The DPTI subsystem provides the ability to transfer data between PC applications' software and logic implemented in the gate array of a supported Digilent system board. It uses an 8-bit bidirectional data bus and four or five control signals to implement an asynchronous or synchronous parallel interface between the host and the device.

The Digilent Parallel Transfer Interface uses the FTDI USB controller's FT245 FIFO communications interface and it can be configured to work either in synchronous or asynchronous mode. The most common arrangement is a dual-port FT2232H chip with channel A wired as FT245 FIFO to user I/O FPGA pins, while channel B as JTAG to dedicated FPGA pins. The two channels are mostly independent with the notable exception that if synchronous mode is enabled on channel A, channel B is disabled. The asynchronous mode does not have this restriction.
FT245 on channel A also shares pins with a SPI interface accessible through the DSPI API. This makes DSPI and DPTI mutually exclusive.

This FIFO-style interface is controlled by the FPGA acting as master, while the USB controller acts as slave. Flags tell the FPGA when a read or write transaction can be started, but the actual read or write

is initiated by the FPGA. A read can only be executed when data is available in the Rx buffer. Similarly, a write needs empty space in the Tx buffer to be executed. These buffers are emptied and filled, respectively, on the other side under the control of the API running on the PC.
The FT2232H includes one Tx and one Rx data buffer of 4 KiB each.

In synchronous mode the clock is input to the FPGA. The USB controller outputs a 60MHz clock signal. All data and control signals are synchronous with this clock. With a data bus width of 8 bits and a 60MHz transfer rate, the maximum theoretical bandwidth is 480 Mbps, the same as for USB 2.0.

In asynchronous mode, data is transferred using only the RX – TX signal pairs: PROG_RXFN and PROG_RDN for download (read), and PROG_TXEN and PROG_WRN for upload (write). Because data is transmitted without using the 60MHz clock, transfer rates are much lower, somewhere between 80 and 160 mbps.

The order of events of using DPTI is as follows:

- PROG_SPIEN signal can be verified to determine whether DPTI or DSPI is active. If only one of the APIs is used on the PC, this is not necessary. This signal outputs logic low when DSPI API is enabled. Enabling the DPTI API will cause it to output logic high instead.
- The host reveals its status using the PROG_RXFN and PROG_TXEN signals. When PROG_RXFN is low, the host is ready to deliver data towards the peripheral (download). When PROG_TXEN is low, then data can be transmitted to the host (upload).
- Launching a download from the API fills the DPTI Rx buffer and PROG_RXFN goes low.
- Since the data bus is bidirectional, PROG_OEN is used to establish the transmission's direction, when the interface is configured in synchronous mode. When the signal is low, then data can be downloaded (read) and when it is high, data can be uploaded (write). This signal is generated by the FPGA.
- A FIFO-style handshake is created with the PROG_RDN (paired with PROG_RXFN for downloading data) and PROG_WRN (paired with PROG_TXEN for uploading data). These two signals are also generated by the peripheral. A data download is only valid when the signals PROG_RXFN and PROG_RDN are low. A data upload is valid when PROG_ TXEN and PROG_ WRN are low.
- Write transfers initiated from the FPGA fill the Tx buffer. Data is transferred over USB when the API launches an upload.

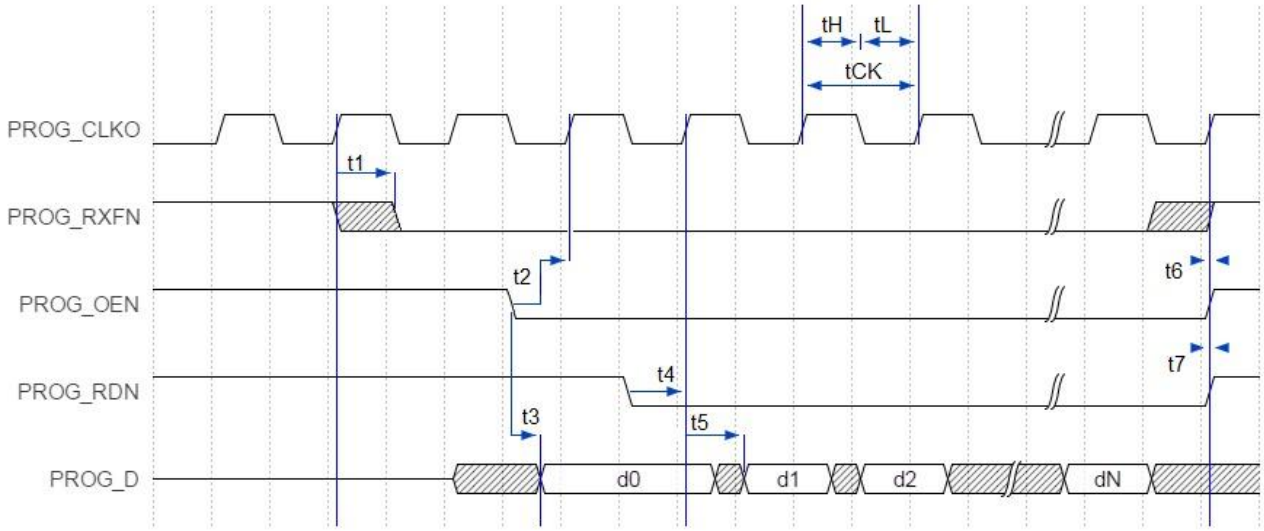
Signal Descriptions

In the following description, the signals are described relative to the FPGA. The following signals make up the interface:

Name	Type	Description
PROG_CLKO	input	60MHz interface clock. The synchronous FIFOs operate on the rising edge of this clock. The peripheral should use this clock to generate control signals and sample the flags.
PROG_OEN	output	Output enable. This signal specifies the data bus direction. When low, data is “downloaded” from the host and when high, data is “uploaded” to the host.
PROG_D [7:0]	bidirectional	Data bus. The host is the source during read (download) cycles and the peripheral is the source during write (upload) cycles.
PROG_RXFN	input	FIFO flag that shows the empty state of the OUT (download) FIFO. While this flag is high the peripheral should not read from the host FIFO.
PROG_TXEN	input	FIFO flag that shows the full state of the IN (upload) FIFO. While this flag is high the peripheral should not write to the host FIFO.
PROG_RDN	output	Enables the current FIFO data byte to be driven onto the on the D0...D7 pins when RD# goes low.
PROG_WRN	output	Enables the data byte on the D0...D7 pins to be written into the transmit FIFO buffer when WR# is low.
PROG_SIWUN	peripheral	Send Immediate / Wake Up pin. If USB is in suspend mode (PWREN# = 1) and remote wakeup is enabled in the EEPROM, strobing this pin low will cause the device to request a resume on the USB Bus. This can be used to wake up the Host PC. During normal operation (PWREN# = 0), if this pin is strobed low any data in the device TX buffer will be sent out over USB on the next Bulk-IN request from the drivers regardless of the pending packet size.
PROG_SPIEN	input	SPI enable. The data bus also doubles as an SPI bus. D0/SCK, D1/MOSI, D2/MISO, D3/SS. This signal tells the peripheral if either the DPTI interface or the SPI interface is enabled. When the signal is high, the DPTI interface is activated and when it is low, the SPI interface is used.

Timing Diagrams

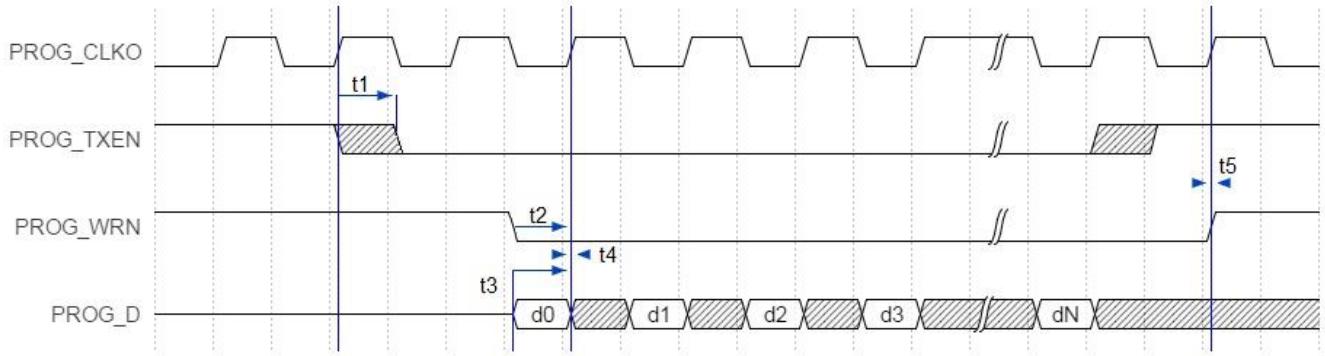
Synchronous Read



Parameter	Description	Min.	Typ.	Max.	Unit
tCK	CLKO period		16.67	16.67	ns
tH	CLKO high period	7.5	8.33	9.17	ns
tL	CLKO low period	7.5	8.33	9.17	ns
t1	CLKO to PROG_RXFN	1		7.15	ns
t2	PROG_OEN setup time	8		16.67	ns
t3	PROG_OEN to PROG_D valid (read)	1		7.15	ns
t4	PROG_RDN setup time to CLKO	8		16.67	ns
t5	CLKO to PROG_D valid (read)	1		7.15	ns
t6	PROG_OEN hold time	0			ns
t7	PROG_RDN hold time	0			ns

A read operation is started when the host drives PROG_RXFN low. The peripheral system can then drive PROG_OEN low to turn around the data bus drivers before acknowledging the data with the PROG_RDN signal going low. The first data byte is on the PROG_D bus after PROG_OEN is low. The peripheral system can burst the data out of the host by keeping PROG_RDN low or it can insert wait states in the PROG_RDN signal. If there is more data to be read it will change on the clock following PROG_RDN sampled low. Once all the data has been consumed, the host will drive PROG_RXFN high. Any data that appears on the data bus after PROG_RXFN is high is invalid and should be ignored.

Synchronous Write



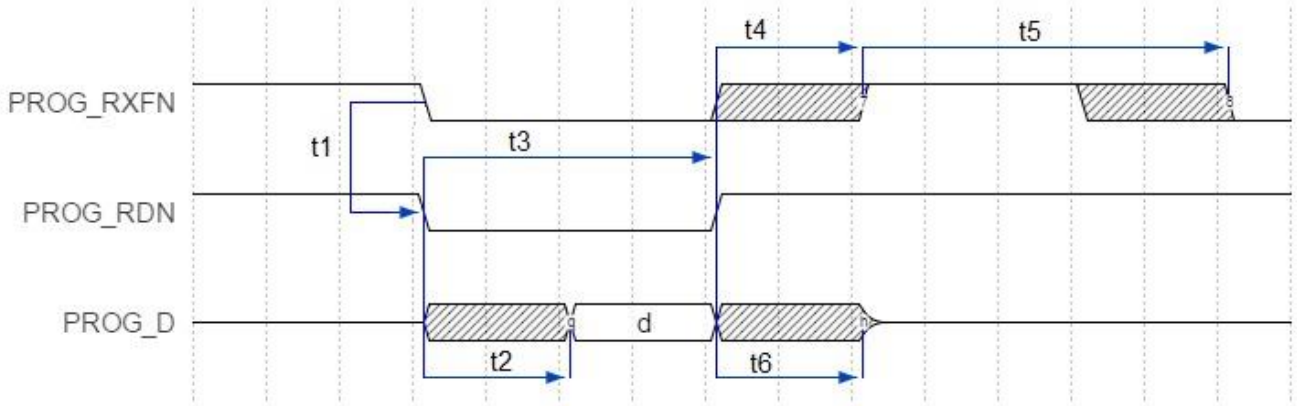
Parameter	Description	Min.	Typ.	Max.	Unit
t1	CLKO to PROG_TXEN	1		7.15	ns
t2	PROG_WRN setup time to CLKO	8		16.67	ns
t3	PROG_D setup time (Write)	8		16.67	ns
t4	PROG_D hold time (Write)	0			ns
t5	PROG_WRN hold time	0			ns

A write operation can be started when PROG_TXEN is low. PROG_WRN is brought low when the data is valid. A burst operation can be done on every clock providing PROG_TXEN is still low. The peripheral system must monitor PROG_TXEN and its own PROG_WRN to check that data has been accepted. Both PROG_TXEN and PROG_WRN must be low for data to be accepted.

Asynchronous mode

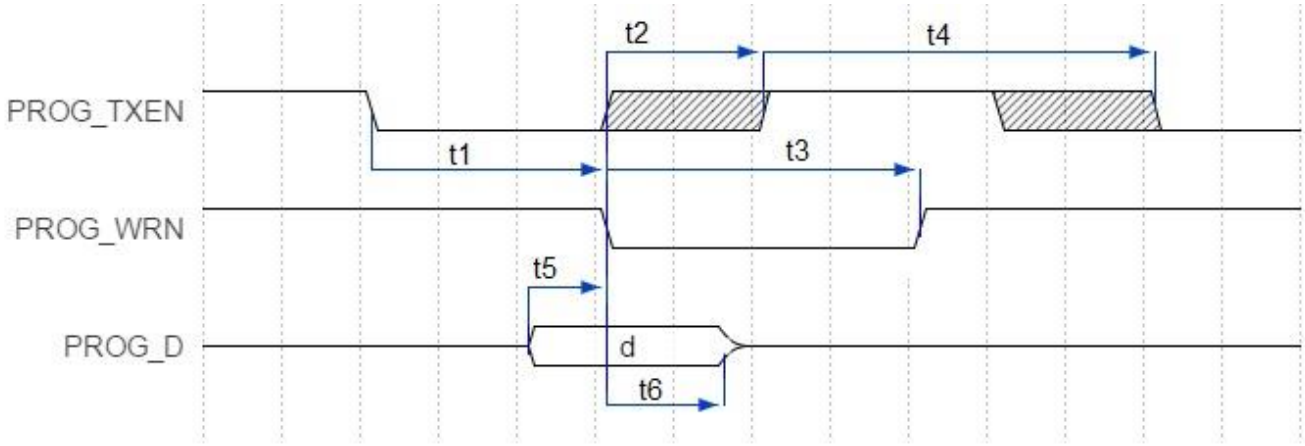
The host can be configured as a dual channel asynchronous FIFO interface. This mode is similar to the synchronous FIFO interface with the exception that the data is written to or read from the FIFO on the falling edge of the PROG_WRN or PROG_RDN signals. This mode does not provide a CLKO signal and it does not expect a PROG_OEN input signal. The following diagrams illustrate the asynchronous FIFO mode timing.

Asynchronous Read



Parameter	Description	Min.	Typ.	Max.	Unit
t1	PROG_RDN active after PROG_RXFN	0			ns
t2	PROG_RDN to PROG_D	1		14	ns
t3	PROG_RDN active pulse width	30			ns
t4	PROG_RDN inactive to PROG_RXFN	1		14	ns
t5	PROG_RXFN inactive after PROG_RDN cycle	49			ns

Asynchronous Write



Parameter	Description	Min.	Typ.	Max.	Unit
t1	PROG_WRN active after PROG_TXEN	0			ns
t2	PROG_WRN active to PROG_TXEN inactive	1		14	ns
t3	PROG_WRN active pulse width	30			ns
t4	PROG_TXEN inactive after PROG_WRN cycle	49			ns
t5	PROG_D to PROG_WRN active setup time	5			ns
t6	PROG_D hold time after PROG_WRN inactive	5			ns

Design Tips

Here are some design tips to consider when implementing a DPTI interface on a Digilent system board:

- Try using the VHDL components included with Adept 2 SDK in your design to implement the DPTI interface and save some development time. This component implements a synchronous interface.
- SIWU does not need to be used for most designs. Simply drive it high.
- You can find the FPGA pin mappings of the DPTI signals for your system board in the Master XDC found on your board's Resource center.
- Note that the PROG_OEN pin must be driven low at least 1 clock period before asserting RD# low.