Digilent Adept Parallel Interface (DPTI) Programmer's Reference Manual

Revision: June 6, 2015

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

Introduction

This document describes the programming interface to the Digilent Adept Parallel Interface (DPTI) subsystem for version 2 of the Digilent Adept software system. It describes the capabilities of the DPTI subsystem and the API functions used to access its features.

The DPTI subsystem provides the ability to transfer data between PC applications software and logic implemented in the gate array of a 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 protocols used by each type of interface described in the document titled "Digilent Parallel Transfer Interface (DPTI)".

Due to limitations of the underlying hardware a single DPTI port cannot switch between asynchronous and synchronous mode. As a result, a DPTI port will either support the asynchronous parallel interface or the synchronous parallel interface, but not both. Since both interfaces share the same data bus and control signals (the synchronous interface has one extra control signal and a clock) a device that supports an asynchronous parallel interface may also be capable of supporting a synchronous parallel interface. Device's that support both types of interfaces will contain multiple DPTI ports.

At the time of writing all devices that support DPTI implement an asynchronous interface on DPTI port 0 and a synchronous interface on DTI port 1. These interfaces share the same data bus and control signals, and as a result, they cannot be enabled simultaneously. Future devices may implement only an asynchronous interface, only a synchronous interface, or implement the interfaces using separate hardware allowing them to be enabled simultaneously. For more information about which interfaces and port combinations are supported, please see the documentation for the applicable Digilent system board.

DPTI Port Properties

The port property bits are used to indicate which type of DPTI interface (asynchronous or synchronous) is supported by a given port.

The following port properties bits are defined for DPTI ports: These values are defined in the header file *dpti.h*.

dprpPtiAsynchronous	This bit indicates that the port implements the asynchronous parallel interface.
dprpPtiSynchronous	This bit indicates that the port implements the synchronous parallel interface.

DPTI API Functions

The following API functions make up the DPTI interface.

DptiGetVersion(char * szVersion)

Parameters

szVersion	- pointer to buffer to receive version string

This function returns a version number string identifying the version number of the DPTI DLL. The symbol cchVersionMax declared in dpcdecl.h defines the longest string that can be returned in *szVersion*.

DptiGetPortCount(HIF hif, INT32 * pcprt)

Parameters

hif	- open interface handle on the device
pcprt	- pointer to variable to receive count of ports

This function returns the number of DPTI ports supported by the device specified by interface handle *hif*.

DptiGetPortProperties(HIF hif, INT32 prtReq, DWORD * pdprp)

Parameters

hif	- open interface handle on the device
prtReq	- port number to query
pdprp	- pointer to variable to return port property bits

This function returns the port properties bits for the specified DPTI port. The port properties bits indicate the specific features of the DPTI specification implemented by the specified port.

DptiEnable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to enable the default DPTI port (port 0) on the specified device. This function must be called before any functions that operate on the DPTI port may be called for the specified device.

DptiEnableEx(HIF hif, INT32 prtReq)

Parameters

hif	- open interface handle on the device
prtReq	- DPTI port number

This function is used to enable a specific port on devices that support multiple DPTI ports. This function must be called before any functions that operate on the DPTI port may be called. The *prtReq* parameter specifies the port number of the DPTI port to enable.

DptiDisable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to disable and end access to the currently enabled DPTI port on the specified interface handle.

DptiSetChunkSize(HIF hif, DWORD cbChunkOut, DWORD cbChunkIn)

Parameters

hif	- open interface handle on the device
cbChunkOut	- maximum number of bytes to attempt to write from the host to device (0 to set default)
cbChunkIn	- maximum number of bytes to attempt to read from the device to the host (0 to set default)

This function is used to set the out (host to device) and in (device to host) chunk sizes for the enabled DPTI port. The out chunk size specifies the maximum number of bytes that the host will attempt to send to the device in a single write operation. The in chunk size specifies the maximum number of bytes that the host will attempt to receive from the device in a single read operation.

If an application requests that more than *cbChunkOut* bytes be sent to the device then the data transfer will be split into multiple transactions, each of which consists of no more than *cbChunkOut* bytes being written at once. Similarly, if an application requests that more than *cbChunkIn* bytes be received from the device then the data transfer will be split into multiple transactions, each consisting of no more than *cbChunkIn* bytes being read at once.

When an application requests a bi-directional data transfer the host will attempt to send up to *cbChunkOut* bytes to the device, and wait a maximum of 2 seconds for the data transfer to complete. After the data transfer completes (or times out) the host will attempt to receive up to *cbChunkIn* bytes from the device by performing a non-blocking read. Since the read is non-blocking it completes almost immediately and has little to no impact on write performance. If an application expects the FPGA logic

to consume data at a slower rate than it produces data, then overall data throughput may be increased by decreasing the out chunk size.

By default both the out and in chunk sizes are set to 64K. Application designers may need to experiment with different settings in order to obtain the highest data throughput possible for the application being implemented. Please note that setting an excessively large chunk size is not recommended, as it may make it difficult for the host to abort a transaction in the event that it wishes to do so. An excessively large chunk size is defined as any chunk size that causes writing or reading a single chunk to take more than a couple of seconds to complete.

DptiGetChunkSize(HIF hif, DWORD * pcbChunkOut, DWORD * pcbChunkIn)

Parameters

hif	- open interface handle on the device
pcbChunkOut	- pointer to return the out chunk size
pcbChunkIn	- pointer to return the in chunk size

This function is used to get the current out and in chunk sizes for the enabled DPTI port. The out chunk size specifies the maximum number of bytes that the host will attempt to send to the device in a single write operation. The in chunk size specifies the maximum number of bytes that the host will attempt to receive from the device in a single read operation.

DptiIO(HIF hif, BYTE * pbOut, DWORD cbOut, BYTE * pbIn, DWORD cbIn, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
pbOut	- pointer to buffer containing data to send from the host to the device
cbOut	- number of bytes to send to the device
pbIn	- pointer to buffer receive data from the device
cbIn	- number of bytes to receive from the device
fOverlap	- TRUE if operation should be overlapped

This function performs a data transfer between the host and the device using the asynchronous or synchronous parallel interface on the enabled DPTI port. Data may transferred from the host to the device by passing in the address of a buffer containing the data using the *pbOut* parameter and specifying a non-zero byte count for *cbOut*. Similarly, data may be transferred from the device to the host by passing in the address of a buffer to receive the data using the *pbIn* parameter and specifying a non-zero byte count for *cbIn*. If no host to device data transfer is desired then NULL should be specified for *pbOut* and a *cbOut* should be specified as 0. If no device to host data transfer is desired then NULL should be specified for *pbIn* and a *cbIn* should be specified as 0.