DS 463 (v2.1), January 2003

OPB ZBT Controller Design Specification

Summary

This document describes the specifications for a ZBT controller core for the On-chip Peripheral Bus (OPB). This document applies to the following peripherals:

opb_zbt_controller	v1.00a

Overview

The ZBT Memory Controller is a 32-bit peripheral that attaches to the OPB (On-chip Peripheral Bus) and has the following features:

Features

- OPB V2.0 bus interface with byte-enable support
- Supports 32-bit bus interfaces
- Supports memory width of 32-bits

Operation

The OPB ZBT Controller provides an interface between the OPB and external ZBT memories. The controller supports OPB data bus widths of 32bits, and memory subsystem widths of 32 bits. This controller supports the OPB V2.0 byte enable architecture.

OPB ZBT Controller Parameters

To allow you to obtain an ZBT Controller that is uniquely tailored for your system, certain features can be parameterized in the ZBT Controller design. This allows your to configure a design that only utilizes the resources required by your system, and operates with the best possible performance. The features that can be parameterized in the Xilinx ZBT Controller design are shown in Table 1.

Table 1: **ZBT Controller Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
ZBT Memory Base Address	C_BASEADDR	Valid Address Range(2)	None(1)	std_logic_vector
ZBT Memory Address Width	C_ZBT_ADDR_SIZE	2-31	17	integer
Implement ZBT Clock synchronization	C_EXTERNAL_DLL	0 = Do implement 1 = Do not implement	0	integer

Notes:

1. Address range specified by C_BASEADDR must be a power of 2
2. No default value is specified for C_BASEADDR to insure that the actual value is set; if the value is not set, a compiler error is generated. These generics must be a power of 2.

ZBT Controller I/O Signals

The I/O signals for the ZBT Controller are listed in Table 2.

Table 2: **ZBT COntroller I/O Signals**

Signal Name	Interface	I/O	Description
OPB_Clk	OPB	I	OPB Clock
OPB_Rst	OPB	I	OPB Reset
OPB_ABus(0:31)	OPB	I	OPB Address Bus
OPB_BE(0:3)	OPB	I	OPB Byte Enables
OPB_DBus(0:31)	OPB	I	OPB Data Bus
OPB_RNW	OPB	I	OPB Read, Not Write
OPB_select	OPB	I	OPB Select
OPB_seqAddr	OPB	I	OPB Sequential Address
ZBT_DBus(0:31zbt)	OPB	O	ZBT Controller Data Bus
ZBT_errAck	OPB	O	ZBT Controller Error Acknowledge
ZBT_retry	OPB	O	ZBT Controller Retry
ZBT_toutSup	OPB	O	ZBT Controller Timeout Suppress
ZBT_xferAck	OPB	O	ZBT Controller Transfer Acknowledge
ZBT_Clk_FB	IP Core	I	Feedback ZBT Clock for clock synchronization
ZBT_Clk_FBOut	IP Core	O	Feedback ZBT Clock for clock synchronization
ZBT_Clk	IP Core	O	ZBT Memory clock
ZBT_CKE_N	IP Core	O	ZBT Memory clock enable
ZBT_OE_N	IP Core	O	ZBT Memory output enable
ZBT_ADV_LD_N	IP Core	O	ZBT Memory Control signal
ZBT_LBO_N	IP Core	O	ZBT Memory Control signal
ZBT_CE1_N	IP Core	O	ZBT Memory Select signal
ZBT_CE2_N	IP Core	O	ZBT Memory Select signal
ZBT_CE2	IP Core	O	ZBT Memory Select signal
ZBT_RW_N	IP Core	O	ZBT Memory Read/Write signal
ZBT_A(0:C_ZBT_ADDR_SIZE-1)	IP Core	O	ZBT Memory Address
ZBT_BW_N(0:3)	IP Core	O	ZBT Memory Byte Enable
ZBT_IO_I(0:31)	IP Core	I	ZBT Memory Input Data Bus
ZBT_IO_O(0:31)	IP Core	O	ZBT Memory Output Data Bus
ZBT_IO_T	IP Core	O	ZBT Memory Output 3-state Signal
ZBT_IOP_I(1:4)	IP Core	I	ZBT Memory Input Parity Data Bus
ZBT_IOP_O(1:4)	IP Core	O	ZBT Memory Output Parity Data Bus
ZBT_IOP_T	IP Core	O	ZBT Memory Output Parity 3-state Signal

Connecting to Memory

The following table shows how to connect a ZBT controller and SAMSUNG K7N163601M-OC15 (512kword x 32). The ZBT controller does not support sleep mode, burst mode, parity checking, or parity generating.

Table 3: **Signal Connection**

ZBT Controller	SAMSUNG K7N163601M-OC15
ZBT_A(0:19)	A(0:19)
ZBT_Clk	CLK
ZBT_CKE_N	$\overline{\text{CKE}}$
ZBT_RW_N	$\overline{\text{WE}}$
ZBT_ADV_LD_N	ADV
ZBT_OE_N	$\overline{\text{OE}}$
ZBT_CE1_N	$\overline{\text{CS1}}$
ZBT_IO(24:31)	DQA(0:7)
ZBT_IO(16:23)	DQB(0:7)
ZBT_IO(8:15)	DQC(0:7)
ZBT_IO(0:7)	DQD(0:7)
ZBT_BW_N(4)	$\overline{\text{BWA}}$
ZBT_BW_N(2)	$\overline{\text{BWB}}$
ZBT_BW_N(1)	$\overline{\text{BWC}}$
ZBT_BW_N(0)	$\overline{\text{BWD}}$
(Need to be tied to VCC)	CS2
(Need to be tied to GND)	$\overline{\text{CS2}}$
(Need to be tied to GND)	ZZ
(Need to be tied to GND)	$\overline{\text{LBO}}$

Address Mapping

The generic C_ZBT_ADDR_SIZE specifies the number of address signals to the ZBT memory. Since all accesses are word, the generic specifies the number of address bits for word accesses. For example, 512 KWord memory needs 19 address bits (2^19 = 512288). The address decoding uses OPB_A(0 to 29-C_ZBT_ADDR_SIZE) and the generic C_BASEADDR to determine if the OPB access is to the ZBT memory.

Timing Diagrams

A read uses five OPB clock cycles to complete, and a write uses 2 OPB clock cycles. The write is faster because pipelining is used for the ZBT controller and the ZBT memories. Figure 1 shows a simple read access. Figure 2 shows two successive writes to illustrate the write pipelining.

Figure 1: **Read Cycle**

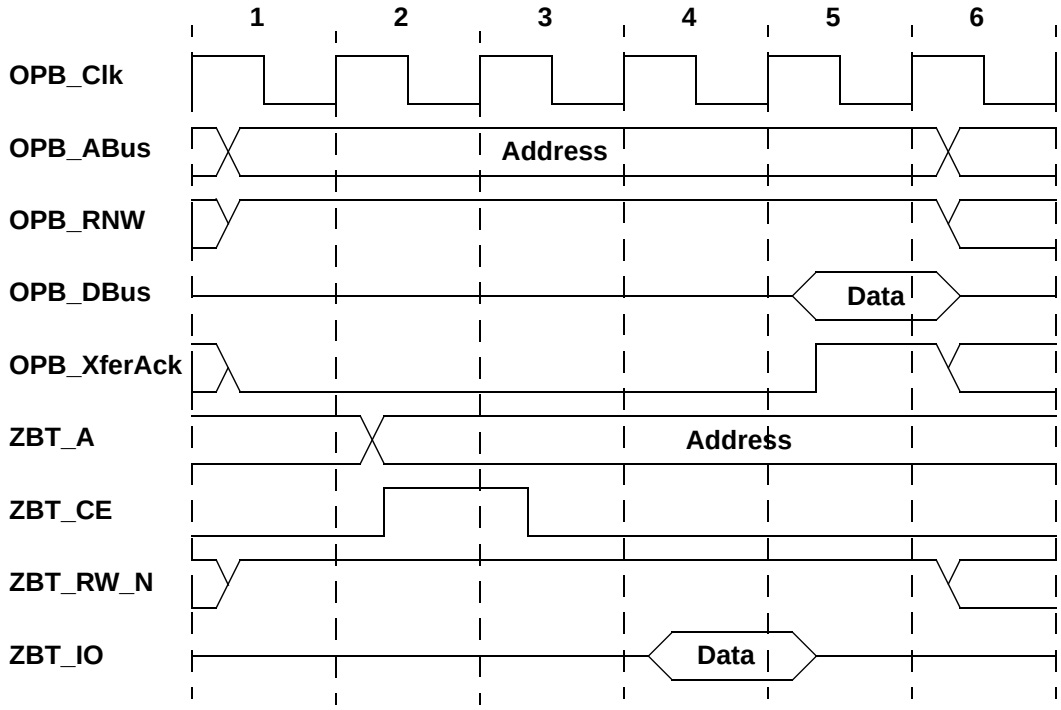
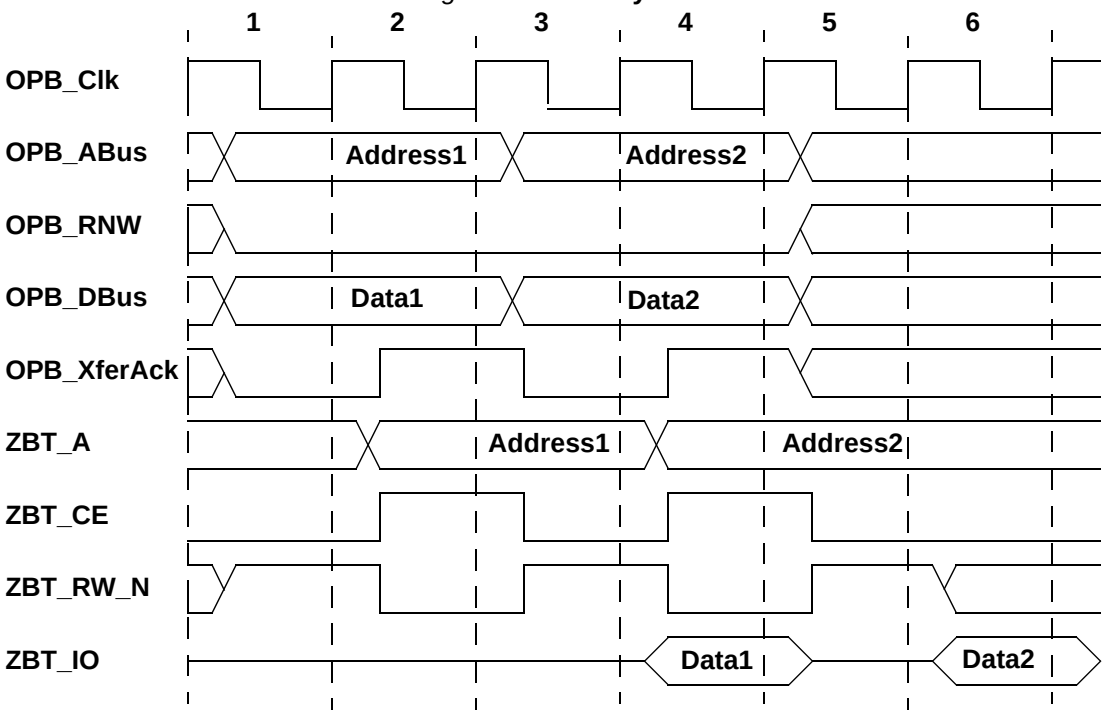


Figure 2: **Write Cycle**

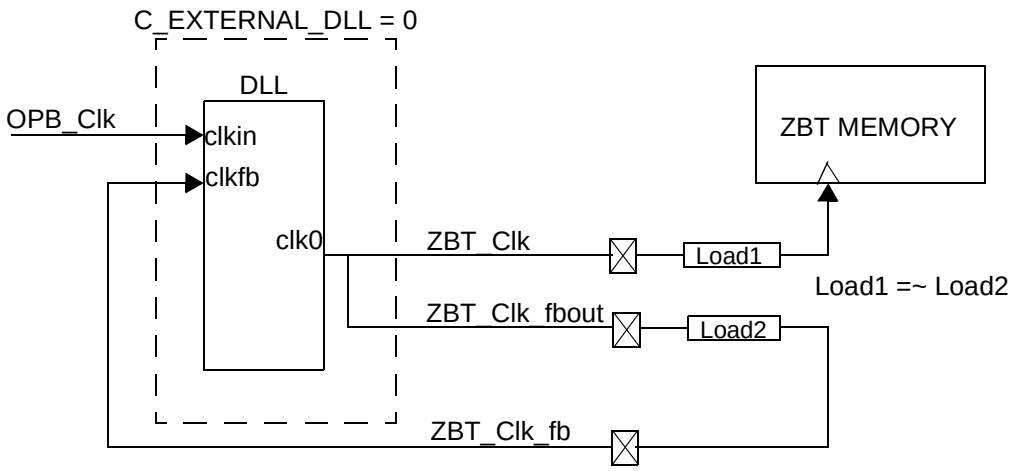


Clock Handling

Since ZBT memories are clocked and all timing on signals to the memories is referencing the ZBT clock, careful handling of the clock is important. Since all signals to the ZBT are driven from the OPB_Clk, the ZBT clock must be synchronized to the OPB_Clk. You can do this with a DLL, as shown in Figure 3

The controller has a generic C_EXTERNAL_DLL that specifies if the peripheral implements the DLL. If there is more than one ZBT controller in a design, the number of DLLs can exceed the number of available DLLs. Consequently, a more centralized handling of the clocks is required. In this case, one DLL synchronizes all ZBT clocks and each of the controllers must inhibit the implementing of the DLL (C_EXTERNAL_DLL = 1). This centralized DLL drives all ZBT clocks, and requires that the loads on each of the ZBT clocks is equal.

Figure 3: **Clock Synchronization**



Programming Model

Register Data Types and Organization

The ZBT controller is organized as big-endian data. The bit and byte labeling for the big-endian data types is shown in Figure 4.

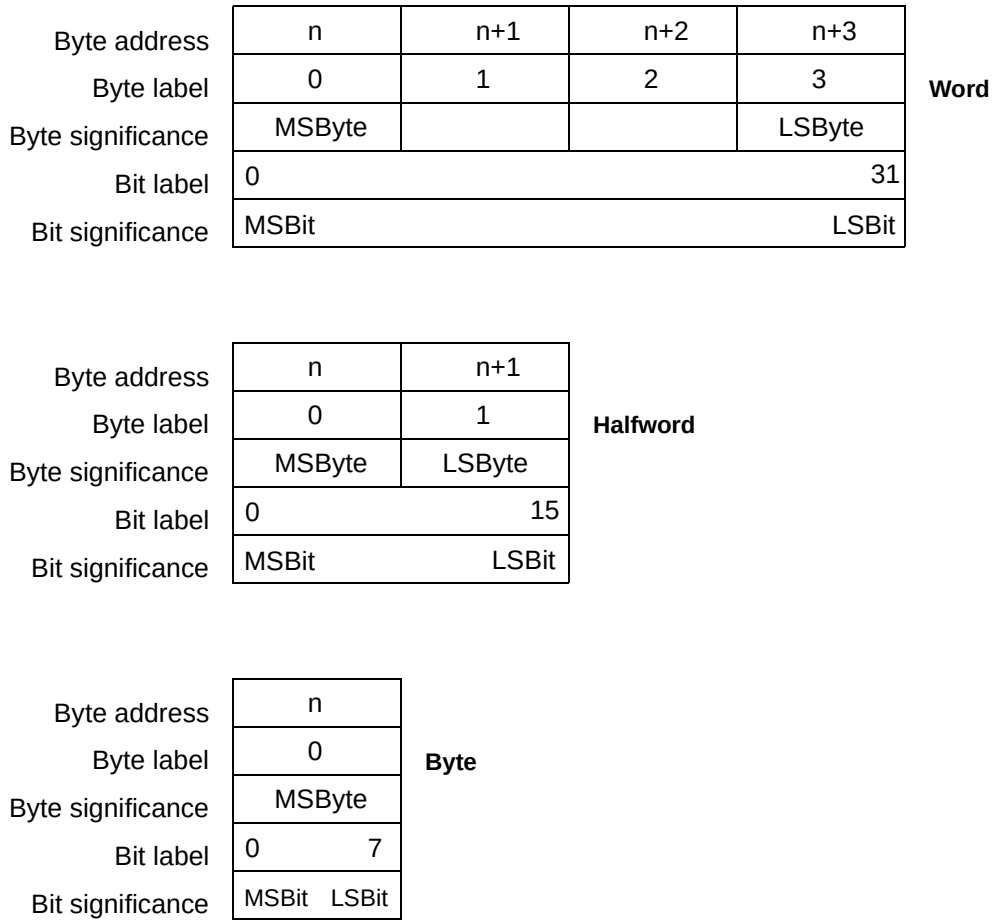


Figure 4: **Big-Endian Data Types**

Implementation

Target Technology

The intended target technology is a Virtex II FPGA.

Device Utilization and Performance Benchmarks

This section will be updated when the design has been completed. It will contain the resources and timing for various values of the parameters.

Since the ZBT Controller is a module that will be used with other design pieces in the FPGA, the utilization and timing numbers reported in this section are just estimates. As the ZBT Controller is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the ZBT Controller design will vary from the results reported here.

The ZBT Controller benchmarks are shown in Table 4 for a VirtexII -5 FPGA.

Table 4: **ZBT Controller FPGA Performance and Resource Utilization Benchmarks (VirtexII -5)**

Parameter Values		Device Resources			f_{MAX}	
C_BaseAddr	C_ZBT_ADDR_SIZE	Slices	Slice Flip-Flops	4-input LUTs	f_{MAX_CMB}	f_{MAX_REG}

Notes:
1. These benchmark designs contain only the ZBT Controller with registered inputs/outputs without any additional logic. Benchmark numbers approach the performance ceiling rather than representing performance under typical user conditions.

Design Tips

To achieve the highest fmax on the ZBT controller, the Xilinx implementation tools must force flip-flops to the IO pads. This option is turned off by default.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
1/7/02	1.0	Initial release.
3/20/02	2.0	Updated for MDK 2.2
01/07/03	2.1	Update for EDK SP3