Self Describing Wishbone Bus Specification

Manohar Vanga (BE/CO/HT)

May 6, 2011

This document describes the specification for the self describing wishbone bus.

1 Introduction

It is advantageous from a software standpoint to have wishbone peripherals that can be probed automatically. This allows for a clean design of operating system drivers and possibly for advanced features such as hot plugging and removal.

This specification introduces a standard that allows the wishbone bus to be probed easily.

2 Specification

The specification has three parts:

- Header block
- ID block
- Device descriptor block

The following points should be noted:

- All values are big endian.
- The presence of 64 bit registers does not imply the requirement for a 64 bit wide data bus. Multiple reads can be done while using a smaller data bus width (eg. 2 reads on a 32 bit data bus).

2.1 Header

The wishbone header contains the locations of the device descriptor block and ID block in the Wishbone address space.

The header block can be placed anywhere within the Wishbone address space as long as there is some way for the operating system to find its location. It is recommended that the address of the header be placed into the configuration space of the parent bus. For example, a BAR in the case of PCI.

The structure of the header block is described in Table 1.

Table 1: Wishbone header block structure

Offset	Size (in bytes)	Name	Access	Value	Description
0x00	0x08	WBHDR_MAGIC	RO	0x5344574248656164	Magic number used to ensure that there is a valid header present.
0x08	0x08	WBIDB_ADDR	RO	-	Address of the Wishbone ID block. See section 2.2 for more information.
0x10	0x08	WBDDB_ADDR	RO	-	Address of the Wishbone device descriptor block. See section 2.3 for more information.

Note that most designs use address 0 for the boot vector. Placing the wishbone header at address 0 should not be done in these cases.

WBHDR_MAGIC (Offset: 0x08)
This field contains a unique value that allows software to ensure that the header contains valid data. If the magic number does not match the expected value, the software should abort immediately.

The magic number in the current version of the specification is expected to be 0x5344574248656164 or the ASCII string "SDWBHead" without the string terminator.

WBIDB_ADDR (Offset: 0x10)
This field contains the address of the Wishbone ID block (see Section 2.2 for more information). The address width can be anything less than or equal to 64 bits. The software reading this field should know what address width to expect.

WBDDB_ADDR (Offset: 0x18)
This field contains the address of the device descriptor block (see Section 2.3 for more information). The address width can be anything less than or equal to 64 bits. The software reading this field should know what address width to expect.

2.2 ID Block

The ID block contains information that uniquely identifies the bitstream within the FPGA.

The ID block additionally contains information about the parent board. This information allows clients unable to access the parent bus memory to know the metadata of the hardware they are accessing.

The structure of the ID block is described in Table 2.

Table 2: Wishbone ID block structure

Offset	Size (in bytes)	Name	Access	Value	Description
0x00	0x08	PCB_ID	RO	-	The PCB ID of the parent board.
0x08	0x08	BSTREAM_TYPE	RO	-	The bitstream type identifier.
0x0C	0x04	BSTREAM_VERSION	RO	-	The version of the specific bitstream.
0x10	0x04	BSTREAM_DATE	RO	-	The synthesis date of the bitstream.
0x14	0x14	BSTREAM_RELEASE	RO	-	The SVN revision number or the SHA-1 hash of the Git release.

PCB_ID (Offset: 0x00)
The PCB ID should be a value that specifies the ID of a specific board within a class of those boards. The format can be specified in any way by a vendor as the software reading this field will be specific to each vendor (selected based on the metadata stored in the parent board) and is expected to know how to decode this field.

BSTREAM_TYPE (Offset: 0x08)
The bitstream type should hold a value that uniquely identifies the type of bitstream present in the FPGA. Note that different bitstreams can have the same type with differing versions (see below).

BSTREAM_VERSION (Offset: 0x0C)
The bitstream version should hold a value that specifies the version of tje bitstream present in the FPGA. This field along with the BSTREAM_TYPE field, should uniquely identify a specific bitstream.

BSTREAM_DATE (Offset: 0x10)
The bitstream date should be set to the hex-readable date of synthesis of the bitstream. An example of a hex-readable date is 0x20111225 (25th December 2011).

BSTREAM_RELEASE (Offset: 0x14)
This field should specify an identifier for the revision control software used to manage the HDL code for the bitstream type. This allows for the possibility of automatic checking out and loading of bitstreams.

2.3 Device Descriptor Block

The device descriptor block describes all the Wishbone peripherals present on the bus. The block is made of an array of device descriptors which have a structure as described in Table 3.

WBD_MAGIC (Offset: 0x00)
This is a unique value used to identify a valid wishbone device structure. If an invalid magic value is found, it is assumed that there are no more devices to be discovered and the autodiscovery is ended. Thus, it is used as the array terminator for the wishbone device block.

The magic number in all versions of the specification can be expected to be 0x5742 or the ASCII string "WB" without the string terminator.

WBD_VER_MAJOR (Offset: 0x02)
The major version of the device descriptor format. This field is incompatible between versions. This means that a change in the descriptor structure itself leads to an increase in the major version. An example of a major version change is the extension of HDL_BASE and HDL_SIZE to 16 bytes (128 bits).

Table 3: Wishbone device descriptor structure

Offset	Size (in bytes)	Name	Access	Value	Description
0x00	0x02	WBD_MAGIC	RO	-	Magic number used to identify a valid device descriptor.
0x02	0x01	WBD_VER_MAJOR	RO	-	The major version of the descriptor format.
0x03	0x01	WBD_VER_MINOR	RO	-	The minor version of the descriptor format.
0x04	0x08	VENDOR	RO	-	The vendor ID of the vendor of the device.
0x0C	0x04	DEVICE	RO	-	The device ID of the device.
0x10	0x08	HDL_BASE	RO	-	Base address (wishbone) of the device.
0x18	0x08	HDL_SIZE	RO	-	Size (in bytes) of the device address space.
0x20	0x04	WBD_FLAGS	RO	-	Device flags.
0x24	0x04	HDL_CLASS	RO	-	Bitstream class.
0x28	0x04	HDL_VERSION	RO	-	Bitstream version.
0x2C	0x04	HDL_DATE	RO	-	Bitstream date.
0x30	0x10	VENDOR_NAME	RO	-	Vendor name (ASCII string)
0x40	0x10	DEVICE_NAME	RO	-	Device name (ASCII string)

WBD_VER_MINOR (Offset: 0x03)

The minor version of the device descriptor format. This field is compatible between versions. This means that a change only in the minor number means the structure is preserved. An example of a minor version change is the addition of a new flag in the WBD_FLAGS field.

This field and WBD_VER_MAJOR can be coalesced into a single WBD_VERSION field if needed.

VENDOR (Offset: 0x04)

The vendor ID of the wishbone device. The vendor space is a 64 bit space. The space is divided into two sections; *reserved* and *free*.

The *reserved* vendor space includes all vendor values with the highest bit unset (0x0000000000000000 - 0x8FFFFFFFFFFFFFFF).

The *free* vendor space includes all vendor values with the highest bit set (0x8000000000000000 - 0xFFFFFFFFFFFFFFFF).

Vendors are free to choose any value within the free space. There is no guarantee of collisions of ID's with other vendors within the free space. In the future, there could possibly be a central repository that allows for guaranteed vendor ID's within the reserved space. The current version of the specification however, provides no guarantee of collisions within the reserved space. If you wish to use ID's from the reserved space, you might need to make modifications to your device in future version of this specification.

DEVICE (Offset: 0x0C)

The device ID of the wishbone device. Together with the vendor ID, the device ID may be used to match device drivers. The format can be specified in any way by a vendor as the software reading this field will be specific to each vendor (selected based on the metadata stored in the parent board) and is expected to know how to decode this field.

HDL_BASE (Offset: 0x10)

This field contains the base address of the Wishbone device. The address width can be anything less than or equal to 64 bits. The software reading this field should know what address width to expect.

This field is writable and can be filled by devices that are responsible for mapping wishbone devices in memory (eg. INTERCON).

HDL_SIZE (Offset: 0x18)
: This field contains the size of the address space of this device. The address width can be anything less than or equal to 64 bits. The software reading this field should know what address width to expect.

  This field is writable and can be filled by devices that are responsible for mapping wishbone devices in memory (eg. INTERCON).

WBD_FLAGS (Offset: 0x20)
: Currently undefined.

HDL_CLASS (Offset: 0x24)
: The class of the wishbone device. The class is used to identify a device with a specific register map, so a host driver can handle all devices of the same class, irrespective of vendor and device numbers. This is similar to PCI or USB devices.

HDL_VERSION (Offset: 0x28)
: This field specifies the version of the device. The format can be specified in any way by a vendor as the software reading this field will be specific to each vendor (selected based on the metadata stored in the parent board) and is expected to know how to decode this field.

HDL_DATE (Offset: 0x2C)
: The bitstream date should be set to the hex-readable date of synthesis of the bitstream. An example of a hex-readable date is 0x20111225 (25th December 2011).

VENDOR_NAME (Offset: 0x30)
: The ASCII string representation of the vendor name. The string should be terminated with the ASCII value 0, thus allowing for a maximum of 15 characters.

DEVICE_NAME (Offset: 0x40)
: The ASCII string representation of the device name. The string should be terminated with the ASCII value 0, thus allowing for a maximum of 15 characters.