An introduction to WISHBONE: A chip-level microcomputer bus

By Wade Peterson

WISHBONE is a System-on-Chip (SoC) standard that solves the basic integrated circuit design problem of assembling systems in a simple, flexible, and portable way. The circuit functions are called cores. System integrators can buy the cores, download them from the Internet, or make them. These cores are integrated together in much the same way that microcomputer boards are integrated into a system.

Development groups can also share their WISHBONE designs. This powerful capability allows large teams to collaborate on projects. It’s a groupware concept that also makes it possible to build online trading systems for cores, such as the one at www.opencores.org. There, open source WISHBONE cores are traded just like open source software in the GNU/Linux world. This makes WISHBONE the first truly open SoC technology.

WISHBONE origins

The goal of making chip-level systems work like mezzanine cards inspired WISHBONE, which was originally conceived as an internal bus for VMEbus interface chips. By the late 1990’s the size and affordability of Field Programmable Gate Arrays (FPGAs) made it possible for a microcomputer bus to run on FPGAs. WISHBONE is a portable system, meaning that chip designs can be moved from one brand of device to another, or to other devices known as Application Specific Integrated Circuits (ASICs).

This portability is a key feature of WISHBONE systems and enables an open marketplace similar to those enjoyed by VMEbus, CompactPCI, and the desktop PC. In open marketplaces the system integrator can shop for the best price, performance, delivery, service, and quality from a number of suppliers. This direct competition also keeps prices low.

Portability also means that WISHBONE designs can be quickly moved into chips with other physical characteristics. Examples include devices with different speeds, system voltages, temperature ranges, and radiation hardening.

Soft systems

Chip-level system integration closely resembles VMEbus system integration. Both let a system integrator build semi-custom computer systems from off-the-shelf components. This saves time and money since many users share the development costs. The major difference is that VMEbus components are hardware modules, and WISHBONE components are software cores. This makes WISHBONE a soft system.

Soft systems have some very intriguing properties. For one thing, they’re usually programmed with Hardware Description Languages (HDL) such as VHDL or Verilog. Software engineers completely describe each system component with these languages, then synthesize and route the components to run on FPGA or ASIC hardware. WISHBONE systems can be built with schematic tools, too.

The ability to be distributed as software lends soft systems a powerful advantage. If the system uses a field-modifiable FPGA chip, integrators can access design changes on a disk or upload changes from networked devices. This significantly cuts design change and field upgrade costs.

With chip technology developing and evolving on a daily basis, emphasis on simplicity and flexibility of circuit design has become more focused than ever. A standard that helps introduce this new level of simplicity and flexibility is WISHBONE. In this article, Wade provides a concise overview of the WISHBONE architecture and describes what WISHBONE is and how it works.

### Table 1

<table>
<thead>
<tr>
<th>System Topology</th>
<th>Target Device</th>
<th>Data Transfer Rate (MAX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit Point-to-point Interconnection</td>
<td>FPGA Xilinx Spartan-II</td>
<td>428 Mbytes/sec</td>
</tr>
<tr>
<td></td>
<td>FPGA Xilinx Virtex-II</td>
<td>812 Mbytes/sec</td>
</tr>
<tr>
<td>32-bit Shared Bus Interconnection with 4 MASTERS and 4 SLAVEs</td>
<td>FPGA Xilinx Spartan-II</td>
<td>220 Mbytes/sec</td>
</tr>
<tr>
<td></td>
<td>FPGA Xilinx Virtex-II</td>
<td>404 Mbytes/sec</td>
</tr>
<tr>
<td>64-bit Shared Bus Interconnection with 4 MASTERS and 4 SLAVEs</td>
<td>FPGA Xilinx Virtex-II</td>
<td>802 Mbytes/sec</td>
</tr>
</tbody>
</table>
When doing system design in this way it’s important to look at the FPGA as a computing machine. High-level languages that generate machine instructions describe all of the system functions. A process called synthesis converts the functions into logic gates. Similarly, microprocessor-based software relies on the compilation of high-level languages such as C into binary op-code machine instructions.

**The WISHBONE architecture**

WISHBONE uses a MASTER/SLAVE architecture that’s similar to that of VMEbus. Cores with MASTER interfaces initiate data transfers to participating SLAVEs. They communicate with each other through a common interconnection core called the INTERCON.

The INTERCON is analogous to a VMEbus backplane because it connects to the MASTER and SLAVE interfaces and carries the system bus traffic. However, its software base allows INTERCON to be changed and makes it much more flexible. This flexibility gives the user complete control over the system topology. The WISHBONE specification defines four types of INTERCON structures:

- Point-to-point
- Data flow
- Shared bus
- Crossbar switch

This flexible structure stands out from traditional VMEbus and CompactPCI microcomputer buses. Those buses use printed circuit backplanes with hardwired connectors and can’t be changed very easily. VMEbus and CompactPCI structures also severely limit how bus modules communicate with each other. In most cases only the number of slots can vary.

The WISHBONE interconnections are nothing more than big, synchronous circuits. They can theoretically work over a nearly infinite frequency range (from nearly DC to light). However, every integrated circuit has physical properties that limit the maximum frequency of the WISHBONE clock. Table 1 shows some benchmarks for WISHBONE systems running on FPGA target devices.

The address and databus widths on WISHBONE interfaces are configured in software with the WISHBONE specification defining address and data widths up to 64 bits. However, the basic architecture can theoretically enable any bus size. Databases can support BIG ENDIAN or LITTLE ENDIAN data byte ordering.

**Flexible interconnection**

The WISHBONE interface’s design flexibility makes it highly efficient and easy to use. The interface signals support a
point-to-point interconnection that needs no glue logic whatsoever. Figure 1a shows the formation of this interconnection. Every system requires a system controller core, called a SYSCON, to provide the system clock and reset signal.

With no glue logic to slow the interface, the point-to-point interconnection (Figure 1b) is the fastest of the WISHBONE bus topologies. The point-to-point interconnection only connects a single MASTER to a single SLAVE interface, however. Designers most often use this topology for testing. For example, designers can link the WISHBONE side of a PCI target interface to a memory core and completely test the interface from the PCI bus.

The WISHBONE interconnection topologies are limited only by the imagination of the system designer. Figure 2 shows four standard topologies defined by the WISHBONE specification.

System designers use data flow interconnection (for pipelined processes in which each core has its own MASTER and SLAVE interface tied together to form a pipeline (see Figure 2b).

The data flow architecture uses parallelism to speed process execution time. For example, if each of the cores in Figure 2b is a floating-point processor unit (FPU), then the system can operate at three times the speed of a single unit. This assumes, of course, that each step in the process requires an equal amount of time to execute and that the problem can be solved in a sequential manner. In actual practice this may or may not be true, but it does show how the data flow architecture provides a high degree of parallelism when solving problems.

The data flow architecture is very popular with PCI interfaces because these interfaces are optimized to work with pipelines connected by First-In/First-Out (FIFO) logic.

Figure 2c depicts a WISHBONE shared bus interconnection. The shared bus interconnection works almost like a chip-level VMEbus. Each core with a MASTER interface can obtain bus ownership from a system arbiter. Once the bus is granted, the module can transfer data to and from SLAVE interfaces. This very popular interconnection will be examined in greater detail later in this article.

As shown in Figure 2d users can arrange WISHBONE cores into crossbar switch configurations. This increases the entire system’s data transfer rate through parallelism. Stated another way, two channels of 100 Mbytes/sec can operate in parallel, to achieve a 200 Mbytes/sec transfer rate. This makes the crossbar switch inherently faster than a shared bus. Most crossbar architectures are scalable, meaning that families of crossbars can be integrated into switched fabrics.

This interconnection scheme variety is only possible because WISHBONE is a soft system that can be defined in a much more abstract way than printed circuit backplanes are. Many chip systems combine these topologies in the same SoC.

### Interface signals

The WISHBONE interface is analogous to the connector on a VMEbus module. However, as a soft system, WISHBONE lets the system integrator completely define the signals. The signals are easy to use and very efficient to implement. WISHBONE interface signals support the system’s various bus cycles and interconnection topologies.

All signals are completely synchronous to a system clock. All use active high logic and have either an _I or _O character attached to them. The characters indicate an input (to the core) or an output (from the core). This nomenclature makes it easier to integrate cores into systems. In this article, all timing waveforms are named from the perspective of a MASTER interface.

For example, the SYSCON functional module generates the MASTER clock...
CLK-O, and so has a _O appended to it. A SLAVE interface uses the same clock as an input, but its signal name is CLK-I. The net connecting the two (in the INTERCON) would simply be called CLK.

Signal arrays (buses) are identified by a name followed by the array boundaries in parentheses. For example, ADR_I(31...0) indicates a 32-bit address bus with array boundaries between zero and thirty-one.

A handshaking mechanism allows either the MASTER or the participating SLAVE to throttle the speed of a data transfer and cause WISHBONE bus cycles to run at the speed of the slowest participating interface.

Figure 3 shows the bus handshaking protocol for the WISHBONE Classic Bus Cycles. There, a MASTER interface asserts its strobe output [STB_O] when it is ready to transfer data. It remains asserted until the SLAVE asserts the acknowledge input [ACK_I]. Either interface can delay the assertion of these signals, thereby allowing it to throttle the data transfer.

All interfaces must support [ACK_I]. This signal indicates a normal cycle termination. Bus transactions can also be terminated with optional error [ERR_I] or retry [RTY_I] inputs.

Table 2 lists the other basic signals. These signals work with a variety of interconnection topologies and are generally attached to either three-state buses or multiplexer logic. The choice depends on what makes sense in the application and what’s available in the integrated circuit. However, multiplexer logic has a significant advantage over three-state logic because it ports across many FPGA or ASIC device brands.

### The WISHBONE bus cycles

The WISHBONE interface currently supports two styles of bus cycles, Classic Cycles and Burst Cycles. The interconnection differentiates the two cycles using cycle tags. The tags work similarly to the VMEbus Address Modifier (AM) code and the PCI Command Byte Enable (C/BE) bits. Table 3 shows the cycle tags.

Cycles in 2002 the OpenCores community added a second class of bus cycles to WISHBONE. They include the SINGLE READ/WRITE, BLOCK READ/WRITE, and RMW (read-modify-write) bus cycles.

In 2002 the OpenCores community added a second class of bus cycles to WISHBONE. They include the SINGLE READ/WRITE, BLOCK READ/WRITE, and RMW (read-modify-write) bus cycles.

The SINGLE READ/WRITE cycles are for the most basic forms of data transfer. There, a MASTER begins the cycle immediately after clock edge 0 (the clock edge numbers are located with the [CLK_I] signal). At that time the MASTER places an address onto address bus [ADR_O], negates the [WE_O] line to indicate a READ cycle, drives the data select line(s) [SEL_O] high or low depending upon the location that is being read, and asserts [STB_O] and [CYC_O] to indicate the start of a new bus cycle.

The interconnection logic decodes the address and asserts [STB_I] to the SLAVE. The SLAVE then places data on its data output bus [DAT_O] and terminates the cycle by asserting [ACK_O]. This informs the MASTER that it should latch the data on the next rising edge of [CLK_I]. After latching the data, the MASTER terminates the cycle by negating [STB_O].

The WRITE cycle is similar to the READ cycle. However, in this case the MASTER negates [WE_O] and presents valid data.
out on [DAT_O()] at the beginning of the cycle. In response, the SLAVE asserts [ACK_O] when it is ready to latch the data at the next rising edge of [CLK_I].

System designers use the [CYC_O] signal mainly for multi-MASTER systems. This signal is asserted for the duration of each bus cycle, and is intended to be used by a system bus arbiter. Figure 4 also shows how the SLAVE can throttle the data transfer speed by holding off the assertion of [ACK_O]. In this way the SLAVE can insert wait states into the cycle (indicated by the -WSS- symbol in the timing diagram). Similarly, the MASTER can throttle the data transfer speed by holding off the assertion of [STB_O] at the beginning of the cycle.

User-defined tags
System integrators can modify the WISHBONE interface with user-defined signals using tagging. A well-known concept in the computer industry, tags allow user-defined information to be associated with an address, a data word, or a bus cycle.

All tag signals must conform to a set of guidelines known as TAG TYPEs. Table 4 lists all of the defined MASTER TAG TYPEs along with their associated data sets. Figure 4 shows the timing. When a tag is added to an interface it is assigned a TAG TYPE from the table that explicitly defines how the tag operates.

Tags serve a variety of purposes. For example, an address tag could include an address parity bit. A data tag could include bits for Error Correction Codes (ECC) or a time stamp, and a cycle tag could indicate whether it should be cached.

Shared bus example
Figure 5 gives an example of a complete WISHBONE shared bus INTERCON. This is one of the more popular interconnections because it works like a classic microcomputer bus. The figure also identifies how the MASTER and SLAVE interfaces can be combined and viewed like slots on a microcomputer bus backplane.

After the initial system reset, one or more MASTER interfaces request the bus by asserting the [CYC_O] signal. If the bus isn’t being used, then the arbiter grants the bus to the MASTER that requested it. It does this by asserting grant lines GNT0 – GNTN and GNT(N,...0). This tells the INTERCON which MASTER can own the bus.

WISHBONE allows any variety of arbitration method. The most popular methods are priority and round robin arbiters.

Once the bus is arbitrated, the INTERCON routes the output signals from the selected MASTER to a common shared bus using multiplexers. For example, if MASTER #0 obtains the bus, then the address lines [ADR_O()] from MASTER #0 are routed to a shared bus [ADR()]. The same thing happens to other signals generated by MASTER #0.

During this interval the address lines from the MASTER drive the common address bus [ADR()]. An address comparator that splits the address space into N sections decodes these. The outputs from the comparator select the SLAVE interface to be accessed by asserting the appropriate [STB_I] signal. At the same time another multiplexer routes data from the selected SLAVE back to the MASTER that initiated the bus transaction.

Once a SLAVE is selected, it participates in the current bus cycle generated by the MASTER. In response to the cycle, the SLAVE must assert its [ACK_O], [RTY_O], or [ERR_O] output. These are routed back to the MASTER with multiplexers, along with any data.

Once the MASTER owning the bus has received the termination signal, it ends the bus cycle by negating [STB_O]. If the MASTER is performing a SINGLE READ/WRITE cycle, then the MASTER terminates the cycle by negating its [CYC_O] signal. This informs the arbiter that it can re-arbitrate the bus.

Conclusion
WISHBONE is a System-on-Chip that operates like a microcomputer bus. It’s a soft system that has a portable and flexible interface. It’s also an open system where integrators can buy cores, make their own, or download them from the Internet. This makes it the first chip-level system that supports open source code sharing just as the GNU/Linux software community does.

<table>
<thead>
<tr>
<th>Description</th>
<th>TAG TYPE</th>
<th>Associated With</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address tag</td>
<td>TGA_O()</td>
<td>ADR_O()</td>
</tr>
<tr>
<td>Data tag, input</td>
<td>TGD_I()</td>
<td>DAT_I()</td>
</tr>
<tr>
<td>Data tag, output</td>
<td>TGD_O()</td>
<td>DAT_O()</td>
</tr>
<tr>
<td>Cycle tag</td>
<td>TGC_O()</td>
<td>Bus cycle</td>
</tr>
</tbody>
</table>

Table 4
Wade Peterson is president and CEO of Silicore Corporation, a consulting firm specializing in electronic design, sensors, and semiconductor cores. Wade is the principle author of the WISHBONE System-on-Chip architecture and has more than 20 years of design experience in industrial controls, sensors, software, and chips. Wade holds a BSE degree in Electrical Engineering from the University of Minnesota.

For further information, contact Wade at:

Silicore Corporation
6310 Butterworth Lane
Corcoran, MN 55340
Tel: 763-478-3567
Fax: 763-478-3568
E-mail: wadep@silicore.net
Web site: www.silicore.net

Figure 5