



WB_INTERCON Configurable Wishbone Interconnect

Summary

This document provides detailed reference information with respect to the configurable Wishbone Interconnect peripheral device.

Core Reference
CR0150 (v2.0) March 12, 2008

The WB_INTERCON peripheral device provides a means of accessing one or more Wishbone-compliant slave devices over a single Wishbone interface. Connecting directly to either the External Memory or Peripheral I/O Interfaces of a processor, the device facilitates communication with physical memory devices or I/O peripherals, respectively.

The WB_INTERCON device can be used with any of the 32-bit processors available in Altium Designer.

Features

- Completely configurable from the schematic sheet
- 1-to-n multiplexing (1 Wishbone Master interface, multiple Wishbone Slave interfaces)
- Ability to control decoder address width
- Automatic hardware decoder generation
- Ability to define specific mapping into Processor address space
- 8-, 16- and 32-bit slave peripheral support
- Configurable addressing modes – allowing a slave device to be either byte or “word” addressed.

Available Devices

The WB_INTERCON device can be found in the FPGA Peripherals integrated library (FPGA_Peripherals.IntLib), located in the \Library\Fpga folder of the installation.

Functional Description

Symbol

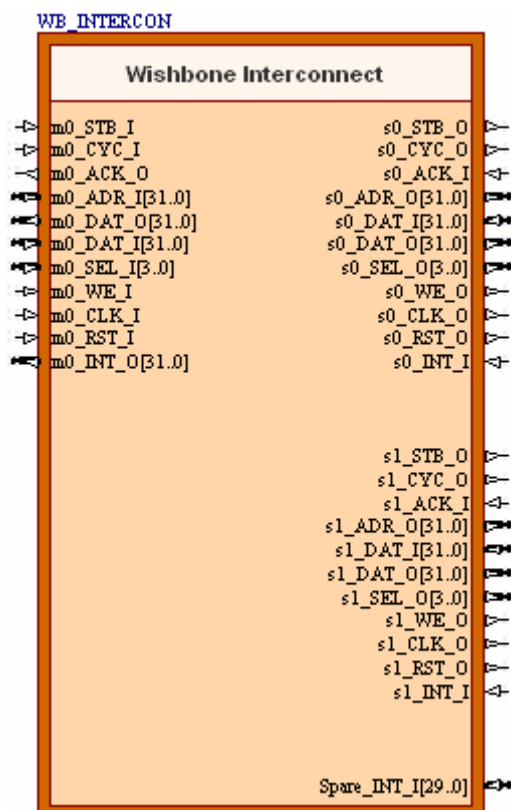


Figure 1. WB_INTERCON symbol

Note: The image in Figure 1 is an example to show all of the available pins that can be present for the device. The actual pins, number of slave interfaces and data/address widths will vary, depending on how the device is configured and whether it is being used to connect memory devices or peripheral I/O devices.

Pin Description

Table 1 summarizes the function of each of the pins available for the device. Only the pins for one slave interface are listed. The function of each of the pins of subsequently added slave interfaces is identical – only the pin names change with respect to their prefix (i.e. s0 for slave interface 0, s1 for slave interface 1, and so on).

Table 1. WB_INTERCON Pin description

Name	Type	Polarity/ Bus size	Description
Wishbone Master Interface Signals			
m0_STB_I	I	High	Strobe signal. When asserted, indicates the start of a valid Wishbone data transfer cycle
m0_CYC_I	I	High	Cycle signal. When asserted, indicates the start of a valid Wishbone bus cycle. This signal remains asserted until the end of the bus cycle, where such a cycle can include multiple data transfers
m0_ACK_O	O	High	Standard Wishbone device acknowledgement signal. When this signal goes High, the connected Wishbone slave device has finished execution of the

Name	Type	Polarity/ Bus size	Description
			requested action and the current bus cycle is terminated
m0_ADR_I	I	24/32 ¹	Standard Wishbone address bus. The address provided on this bus is decoded to select the correct slave device that the processor wishes to communicate with and also which memory address/internal register of that device is accessed
m0_DAT_O	O	32	Data to be sent to the connected Wishbone master device
m0_DAT_I	I	32	Data received from the connected Wishbone master device
m0_SEL_I	I	4/High	Select input, used to determine where data is placed on the m0_DAT_O line during a Read cycle and from where on the m0_DAT_I line data is accessed during a Write cycle. Each of the data ports is 32-bits wide with 8-bit granularity, meaning data transfers can be 8-, 16- or 32-bit. The four select bits allow targeting of each of the four active bytes of a port, with bit 0 corresponding to the low byte (7..0) and bit 3 corresponding to the high byte (31..24)
m0_WE_I	I	Level	Write enable signal. Used to indicate whether the current local bus cycle is a Read or Write cycle. 0 = Read 1 = Write
m0_CLK_I	I	Rise	External (system) clock signal
m0_RST_I	I	High	External (system) reset signal
Wishbone Slave Interface Signals			
s0_STB_O	O	High	Strobe signal. When asserted, indicates the start of a valid Wishbone data transfer cycle
s0_CYC_O	O	High	Cycle signal. When asserted, indicates the start of a valid Wishbone bus cycle. This signal remains asserted until the end of the bus cycle, where such a cycle can include multiple data transfers
s0_ACK_I	I	High	Standard Wishbone device acknowledgement signal. When this signal goes High, the connected Wishbone slave device has finished execution of the requested action and the current bus cycle is terminated
s0_ADR_O	O	0-32 ²	Standard Wishbone address bus. When the WB_INTERCON is used to connect to one or more slave memory devices, this bus is used to select an address in the connected memory device for writing to/reading from. When the WB_INTERCON is used to connect to one or more slave peripheral devices, this bus is used to select an internal register in the connected peripheral device for writing to/reading from.
s0_DAT_I	I	8/16/32 ³	Data received from the connected Wishbone slave device
s0_DAT_O	O	8/16/32	Data to be sent to the connected Wishbone slave device
s0_SEL_O	O	4/High	Select output, used to determine where data is placed on the s0_DAT_O line during a Write cycle and from where on the s0_DAT_I line data is accessed during a Read cycle. Each of the data ports is 32-bits wide with 8-bit granularity, meaning data transfers can be 8-, 16- or 32-bit. The four select bits allow

¹ m0_ADR_I will be 24 bits when connected to the processor's Peripheral I/O interface and 32 bits when connected to the External Memory interface. Use the **Master Address Size** option when configuring the Interconnect device to define this address bus sizing.

² The number of address bits is determined by the value entered for the **Address Bus Width**, when defining the properties for a slave device.

³ The size of the data bus is determined by the value entered for the **Data Bus Width**, when defining the properties for a slave device.

WB_INTERCON Configurable Wishbone Interconnect

Name	Type	Polarity/ Bus size	Description
			targeting of each of the four active bytes of a port, with bit 0 corresponding to the low byte (7..0) and bit 3 corresponding to the high byte (31..24)
s0_WE_O	O	Level	Write enable signal. Used to indicate whether the current local bus cycle is a Read or Write cycle. 0 = Read 1 = Write
s0_CLK_O	O	Rise	External (system) clock signal (identical to m0_CLK_I), made available for connecting to the CLK_I input of the connected slave device. Though not part of the standard Wishbone interface, this signal is provided for convenience when wiring your design
s0_RST_O	O	High	Reset signal made available for connection to the RST_I input of the connected slave device. This signal goes High when an external reset is issued to the unit on its m0_RST_I pin. Though not part of the standard Wishbone interface, this signal is provided for convenience when wiring your design
Interrupt Signals (Multiplexed Peripheral I/O Devices only)			
m0_INT_O	O	32	Interrupt signals sent to the connected Wishbone master device
sn_INT_I	I	p^4	The number of interrupt input signals (p) received from the associated slave peripheral device (where n represents the particular slave interface)
Spare_INT_I	I	$32 - q^5$	Spare interrupt signals input. If the connected slave peripheral devices do not use all 32 individual interrupt signals that can be passed to the processor, those remaining can be made available to receive interrupt signals from other circuitry in the design

⁴ This number is determined by the value entered for the **Used Interrupts**, when defining the properties for a slave device.

⁵ Where q represents the total number of interrupt pins assigned to all connected slave devices.

Configuring the Device from the Schematic Design

The WB_INTERCON device can be configured after placement on the schematic sheet. Simply right-click and choose the command to configure the device from the pop-up menu that appears (e.g. **Configure U_INTERCON_MEM (WB_INTERCON)** for a device with designator U_INTERCON_MEM). Alternatively, click on the **Configure** button, available in the *Component Properties* dialog for the device.

The *Configure (Wishbone Intercon)* dialog will appear, as shown in Figure 2.

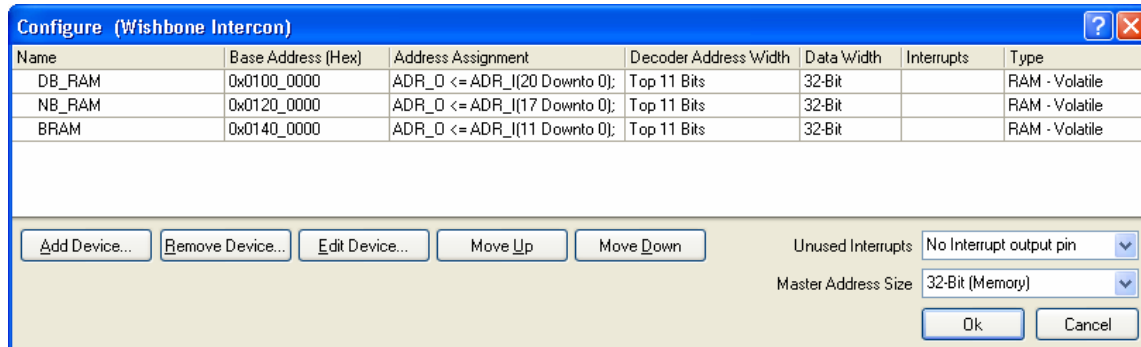


Figure 2. Configuring the structure of the Interconnect

Use this dialog to add and define the slave devices that you wish to connect to the processor.

The main region of the dialog presents a spreadsheet-like listing of all slave devices that are currently added. For each device, the following information is displayed:

- Name
- Base Address (Hex)
- Address Assignment
- Decoder Address Width
- Data Width
- Interrupts
- Type.

For detailed information with respect to each of these fields, refer to the section [Adding a Slave Device](#).

Note: The order of the devices in the dialog determines the position of their corresponding interface on the WB_INTERCON's schematic symbol.

If you are using the WB_INTERCON device to connect to slave memory devices, ensure that the **Master Address Size** option is set to 32-Bit (Memory). If using the device to connect to slave peripheral I/O devices, ensure that this option is set to 24-Bit (Peripheral I/O).

When connecting to slave peripheral devices, one or more of those devices may generate interrupts, the number of which can be configured separately for each device, appearing as additional sn_INT_I inputs for the relevant slave interfaces. If the connected slave devices do not use all of these available interrupts, you can specify what is to happen to the remainder using the **Unused Interrupts** option. The available options are:

- **Add SPARE INT input pin** – use this option to make the unused interrupt signals available as an additional input pin. Interrupts from additional circuitry in the design can be wired into this pin.
- **Connect to GND** – use this option to internally connect all unused interrupts to GND. The interrupt output signal pattern that is sent to the Wishbone Master device will contain '0' for each of these unused interrupts.
- **No Interrupt output pin** – use this option to effectively disable marshalling of interrupts to the processor. Any configured interrupt input pins for slave devices will be removed from the symbol, as well as the spare interrupts pin (where applicable) and the output pin to the Wishbone Master.

Adding a Slave Device

To add a slave device to the WB_INTERCON, simply click on the **Add Device** button in the main *Configure (Wishbone Intercon)* dialog or choose **Add Slave** from the dialog's right-click menu. The *Device Properties* dialog will appear (Figure 3).

Device Properties

Slave Name and Type

Identifier

Type

Choose the identifier and type of the device.

Address Base

This is the base address of the device.
For peripheral in the I/O space this is a 24-bit hexadecimal number and for memory devices this is a 32-bit number.

Address Bus Mode

Byte addressing will result in master ADR_I(0) will be translated to slave ADR_O(0).
Word addressing will be dependent on the data width:
For 32-bit wide devices master ADR_I(2) will be translated to slave ADR_O(0) providing 4-byte words at each address.
For 16-bit wide devices master ADR_I(1) will be translated to slave ADR_O(0) providing 2-byte words at each address.
For 8-bit wide devices this mode is the same as byte addressing.

Decode Addressing

Controls how many of the address bits are decoded to select the peripheral. Decoders are generated automatically.
For example, a value of 8 would mean that ADR(31 DownTo 24) (or ADR(23 DownTo 16) for 24 bit) is compared against the upper 8-bits of the Decode Address to select the peripheral.
The smaller the number, the lower the hardware overhead but the less the number of different devices can be used.

Address Bus Width

This represents the number of address bits that are required to drive the slave.

Graphical Attributes

Controls the extra space after the slave's bank of pins.

Data Bus Width

The width of the data bus on the slave device.

Used Interrupts

Any processor interrupt lines that are not used by the slaves will be available as a single Spare_INT_I pin on the WB_INTERCON

OK Cancel

Figure 3. Defining the interface properties for a connected slave device

Use the dialog to define properties of the slave device as required. After clicking **OK** the device will appear in the main list region of the *Configure (Wishbone Intercon)* dialog and the schematic symbol for the device will be updated accordingly, to include an additional Wishbone interface for this slave device.

The following sections explore each of the various regions of this dialog, which collectively define the interface to the slave device being connected.

Slave Name and Type

This region of the dialog allows you to assign a unique identifier for the slave device being connected. The name should be meaningful so that you can readily identify the device when listed alongside all other connected devices in the main *Configure (Wishbone Intercon)* dialog. The identifier is also used when configuring processor address space and importing slave device address information from the schematic.

Use the **Type** field to define what type of slave Wishbone device is being connected. Choose from the following:

- Peripheral
- ROM
- RAM – Volatile
- RAM – Non-Volatile.

Address Bus Mode

This region of the dialog allows you to specify the addressing mode employed when mapping the address line between the master (m0_ADR_I) and a slave (sn_ADR_O) connected to the interconnect device. Two modes are available:

- Word Addressing – $ADR_O(0) \leq ADR_I(1 \text{ or } 2)$
- Byte Addressing – $ADR_O(0) \leq ADR_I(0)$

In byte addressing mode, all of the lower address lines are passed to the slave, no matter what the resolution of its data bus. The slave device will handle “byte-to-word” management.


In word addressing mode, the mapping of the address lines is dependent on the resolution of the slave device’s data bus width:

- **32-bit wide devices** – the two lowest address bits are not connected to the slave device. ADR_I(2) from the master is mapped to ADR_O(0) of the slave, providing sequential word addresses (or addresses at every 4 bytes). Registers/address locations in such devices can be read and written using the LW and SW 32-bit load/store instructions
- **16-bit wide devices** – the lowest address bit is not connected to the slave device. ADR_I(1) from the master is mapped to ADR_O(0) of the slave, providing sequential half-word addresses (or addresses at every 2 bytes). Registers/address locations in such devices can be read and written using the LHU and SH 16-bit load/store instructions
- **8-bit wide devices** – all address bits are connected through to the slave device. ADR_I(0) from the master is mapped to ADR_O(0) of the slave, providing sequential byte addresses. This is identical to byte addressing. Registers/address locations in such devices can be read and written using the LBU and SB 8-bit load/store instructions.

Address Base

This region of the dialog allows you to specify a decoder base address for the slave device. A designated portion of this address – specified by the Decode Addressing value – will be compared against the corresponding bits of the incoming m0_ADR_I signal, to determine whether the slave is being addressed by the processor or not.

For a slave memory device, the address base is a 32-bit hexadecimal number, entered as 8-digits. For a slave peripheral device, it is a 24-bit hexadecimal number, entered as 6-digits.

 It is possible to specify invalid addresses. If the device is a peripheral and an address greater than FF_FFFFh is entered (i.e. greater than 24-bit), then the higher bits will be ignored.

If the device is memory and an address lower than 0100_0000h is supplied, then the device will never be selected because the internal processor decoder routes all such addresses to the processor's Internal Memory space. Such addresses will therefore never appear on the processor's Wishbone External Memory interface.

Decode Addressing

This region of the dialog enables you to define decoder address width. The value entered determines the number of upper address bits on the m0_ADR_I line that are decoded to select the correct slave device. This value therefore also determines the number of slave devices that can be connected to the interconnect device.

Considering a bank of slave peripheral devices, a value of 4 would mean that m0_ADR_I(23..20) would be compared against the upper 4 bits of the 24-bit values defined for each peripheral's Address Base.

For a bank of slave memory devices, with 32-bit addressing, a value of 4 would mean that m0_ADR_I(31..28) would be compared against the upper 4 bits of the 32-bit values defined for each device's Address Base.

The decoders are generated automatically. Using a smaller number of bits for comparison will lower the hardware overhead, but also limit the number of slave devices that can be used. The optimal scenario would be to make the decode address width as small as possible while allowing enough slave devices to be added to the interconnect device, in accordance with design requirements. For example, if you need to use an interconnect device to connect to 8 peripheral devices only, set the decode address width to 3 and define the Address Base values for the peripherals to provide an even spread through the 16MB of Peripheral I/O space. 3-bit comparators will be generated to perform the decoding.

Table 2 illustrates this example, showing the comparison bits (upper 3 bits of peripheral Address Base), to which the upper 3 bits of the incoming address from the processor (m0_ADR_I(23..21)) will be compared.

Table 2. Example decode addresses for a bank of slave peripheral devices

Peripheral Device	Address Base (6-digit Hex)	Comparison bits
1	000000	000
2	200000	001
3	400000	010
4	600000	011
5	800000	100
6	A00000	101
7	C00000	110
8	E00000	111

Note: The Decode Addressing value determines the number of devices that can be connected as follows:

WB_INTERCON Configurable Wishbone Interconnect

- for 8-bit devices – entering a value n allows for 2^n devices
- for 16-bit devices – entering a value n allows for 2^{n-1} devices
- for 32-bit devices – entering a value n allows for 2^{n-2} devices.

Address Bus Width

This region of the dialog allows you to specify the number of address bits required to drive the connected slave device. For slave memory devices – which are connected via the appropriately configured Memory Controller device – you need to set the address bus to the same width as the ADR_I line for the Memory Controller. The Memory Controller will automatically size its ADR_I line according to the size of the physical memory it is connecting to.

For slave peripheral devices, you also just need to set the address bus to the same width as the ADR_I line for the peripheral.

Data Bus Width

This region allows you to specify the resolution of the data bus for the slave device being connected. 8-bit, 16-bit and 32-bit data bus widths are supported.

Used Interrupts

This region allows you to specify how many, and more precisely which, of the processor's interrupt pins the slave device requires. By allowing you to specify which interrupt line you wish to use, you are effectively determining the priority of interrupt-generating devices connected to the Interconnect device.

Click on the ... button to the right of the available field to access the *Use Interrupts* dialog (Figure 4).

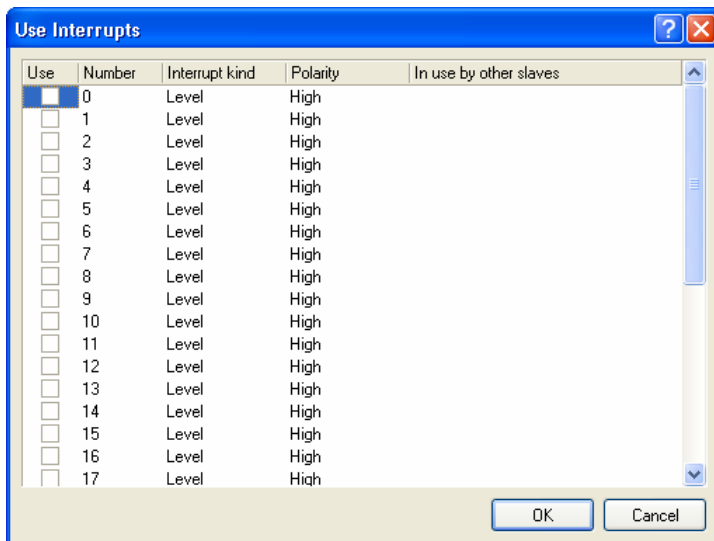


Figure 4. Specifying interrupt usage.

This dialog allows you to choose which interrupt(s) to use for the slave device in question, simply by checking the box to the left of the required interrupt pin number(s). The dialog will display which interrupt lines are currently in use by other slave devices connected to the same Wishbone Interconnect device, helping to prevent multiple use of the same line(s).

For each interrupt line, you can also define its type – whether it is level-sensitive or edge-triggered – as well as its polarity. Interrupts generated by Altium Designer Wishbone peripheral devices have positive polarity and are level-sensitive, and this is therefore the default setting.

You can also specify the required interrupt pins to be used by directly typing within the available text field – in the *Device Properties* dialog. The entry required will depend on how you want the interrupt configured and whether multiple interrupts are required:

- Level-sensitive, positive polarity (default) – enter the number (e.g. 1 for interrupt line 1).
- Level-sensitive, negative polarity – enter the number with a '-' prefix (e.g. -1 for interrupt line 1).
- Edge-triggered, rising edge – enter the number with an 'E' prefix (e.g. E1 for interrupt line 1).
- Edge-triggered, falling edge – enter the number with an '-E' prefix (e.g. -E1 for interrupt line 1).
- To specify multiple interrupt pins, separate the entries with a comma (e.g. 2, 3, 4 or E1, -E2, -3).

Any defined interrupts will appear as part of the overall 32-bit interrupt input bus sent to the Wishbone Master (e.g. a 32-bit host processor).

Graphical Attributes

This region of the dialog enables you to alter the amount of blank space that is inserted after the bank of pins for the slave currently being defined. This allows you to space the slave interfaces appropriately, so that each slave device can be directly connected to its corresponding interface, without the need for additional external wiring.

Editing a Slave Device

To edit the definition of an existing slave device that is connected to the WB_INTERCON, simply select the entry for that device in the main list region of the *Configure (Wishbone Intercon)* dialog and click on the **Edit Device** button or choose **Properties** from the dialog's right-click menu. The *Device Properties* dialog will appear, from where you can make modifications as required.

Removing a Slave Device

To remove an existing slave device that is connected to the WB_INTERCON, simply select the entry for that device in the main list region of the *Configure (Wishbone Intercon)* dialog and click on the **Remove Device** button or choose **Remove Slave** from the dialog's right-click menu. A dialog will appear asking for confirmation to proceed with the deletion. Clicking **Yes** will remove the device entry from the main list and also remove the Wishbone interface for that slave from the schematic symbol for the device.

Changing Slave Interface Ordering

After defining the slave device interfaces, you may find when connecting the actual slaves that you want to place them in a different order, in terms of connection to the WB_INTERCON device. Rather than having to delete and re-add device definitions, or rely on less-than-desirable wiring to achieve your goal, support for slave interface re-ordering is provided through two additional buttons in the main *Configure (Wishbone Intercon)* dialog – **Move Up** and **Move Down**. Simply select a slave device entry in the main list region of the dialog and use these buttons to change its corresponding interface position on the WB_INTERCON's schematic symbol.

Using the Wishbone Interconnect Device in a Design

The WB_INTERCON device can, as previously discussed, be used to connect to one or more slave memory or peripheral I/O devices. The following sections provide examples of where the interconnect device can be used in a design.

Connecting Single Slave Devices

Although a single memory or peripheral I/O device can be connected directly to the respective Wishbone interface of a processor (with additional wiring), use of a Wishbone Interconnect device greatly simplifies matters, not only from its ease of physical connection on the schematic, but also from its ability to handle the address mapping from the processor to the slave device.

Figure 5 shows the use of a Wishbone Interconnect device to connect the SRAM on a Daughter Board to the External Memory interface of a TSK3000A processor. The physical SRAM is connected to the interconnect via an appropriately-configured Memory Controller device (WB_MEM_CTRL).

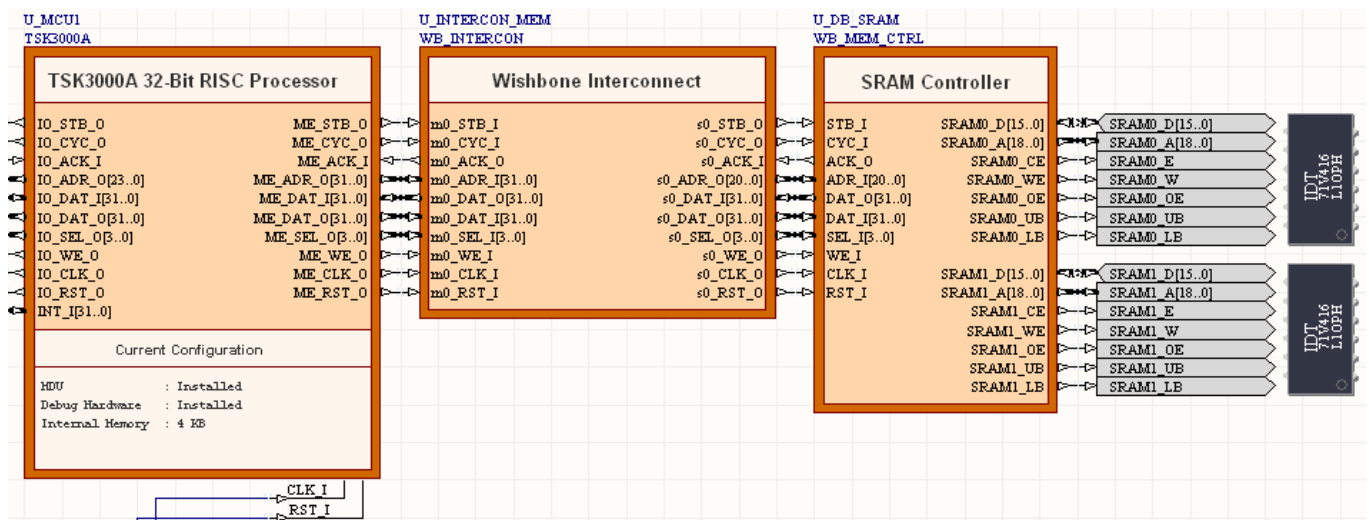


Figure 5. Using a Wishbone Interconnect device to facilitate simple connection to memory.

For further information on the Wishbone Memory Controller peripheral, refer to the [WB_MEM_CTRL Configurable Wishbone Memory Controller](#) core reference.

Figure 6 shows similar use of a Wishbone Interconnect device to connect a single slave peripheral device (a VGA Controller) to a TSK3000A's Peripheral I/O interface.

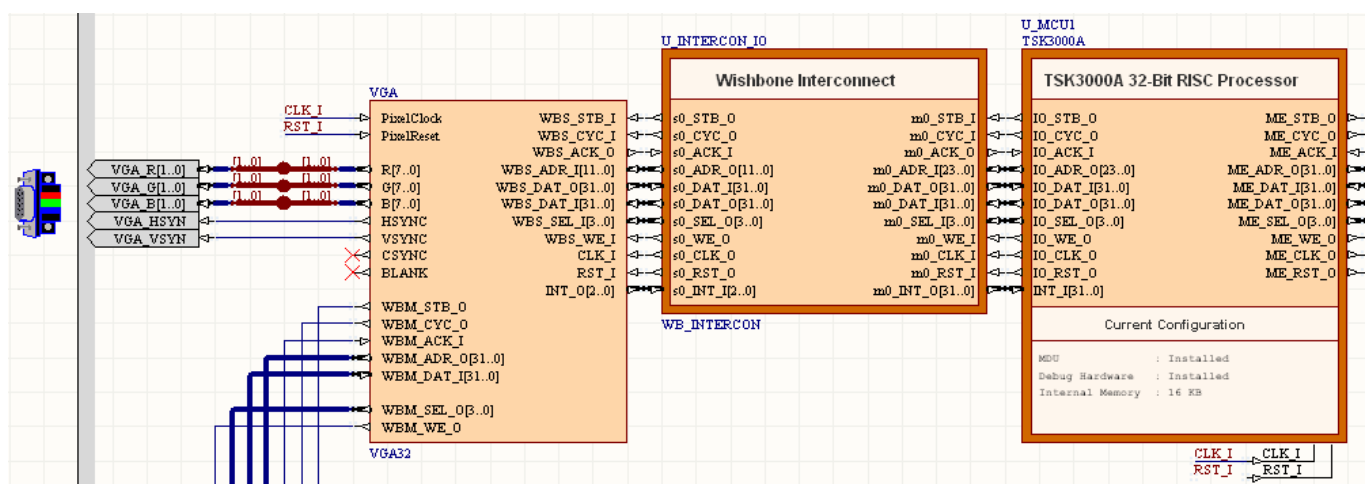


Figure 6. Using a Wishbone Interconnect device to facilitate simple connection to a single slave peripheral

Connecting Multiple Memory Devices

The nature of your design may warrant the use of several memory devices, possibly of differing type, each of which requires to be mapped into a specific location within the processor's address space. This can be readily achieved through the use of a Wishbone Interconnect device. Simply connect the Wishbone Master interface of the WB_INTERCON directly to the processor's External Memory Interface and then add and configure interfaces to each slave memory device as required.

Figure 7 illustrates the use of a Wishbone Interconnect device to connect to static RAM devices on a Daughter Board, one of the static RAM devices on the NanoBoard and a dedicated single-port block of RAM within the design. In each case, the respective Memory Controller (configured as either an SRAM Controller or a BRAM Controller) sits between the Wishbone Interconnect device and the physical memory device(s).

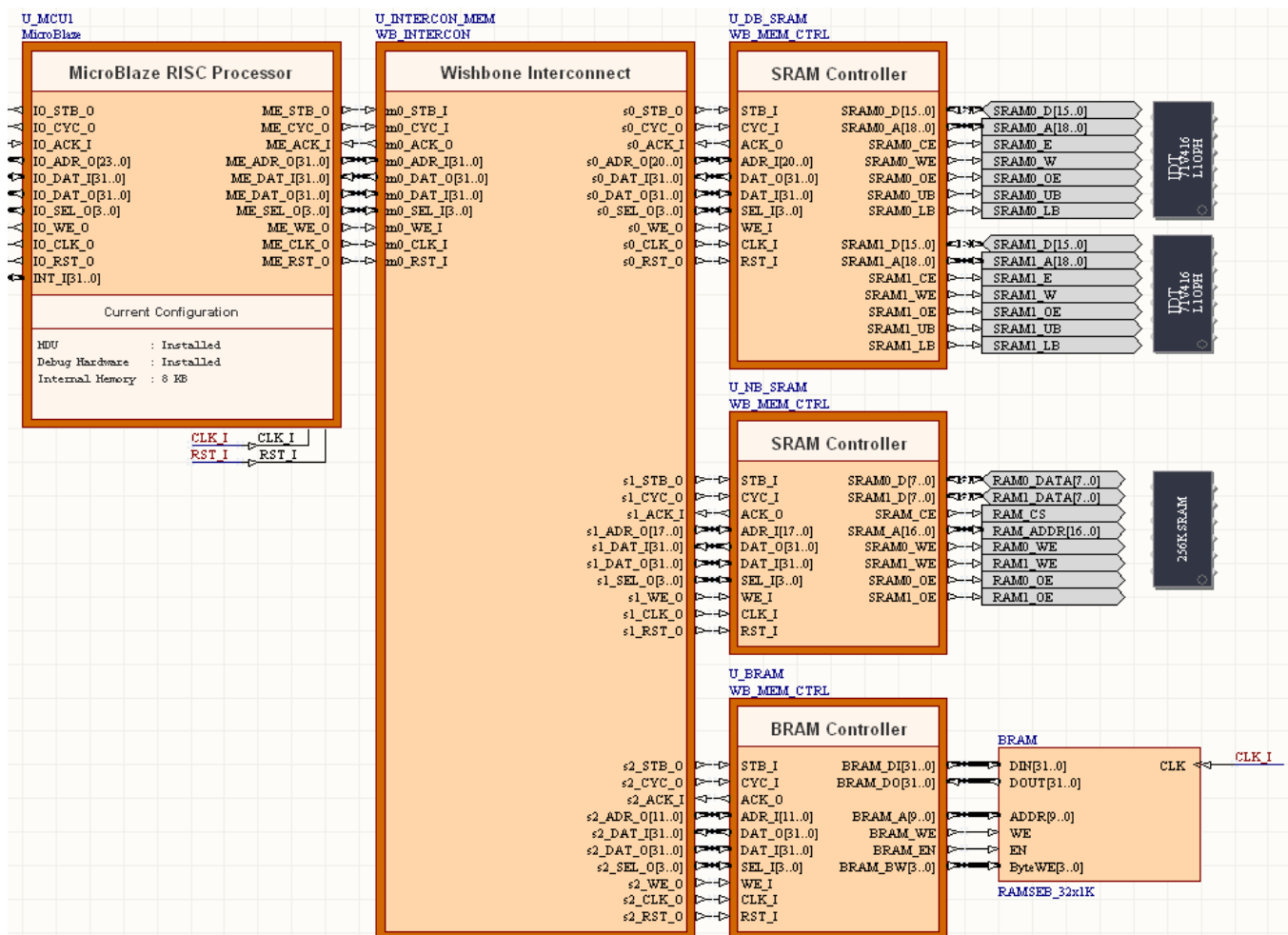


Figure 7. Multiplexing a 32-bit processor's External Memory interface using a Wishbone Interconnect device.

Connecting Multiple Peripheral I/O Devices

Typically in a design, the processor will need to interface to multiple Wishbone-compliant peripherals. Each of these peripherals may contain any number of internal registers with which to write to/read from. It is not possible to communicate directly, and simultaneously, with each of these slave devices. A means of multiplexing must be used, allowing the processor to talk to any number of slaves over the one interface.

Again, this can be readily achieved through the use of a Wishbone Interconnect device. Simply connect the Wishbone Master interface of the WB_INTERCON directly to the processor's Peripheral I/O Interface and then add and configure interfaces to each slave peripheral device as required.

In the example circuit of Figure 8, the Wishbone Interconnect peripheral enables a single 32-bit processor (TSK3000A) to communicate with three Wishbone-compliant peripheral devices (a 1X8 Parallel Port Unit, a Serial Port Unit and a PS2 Controller).

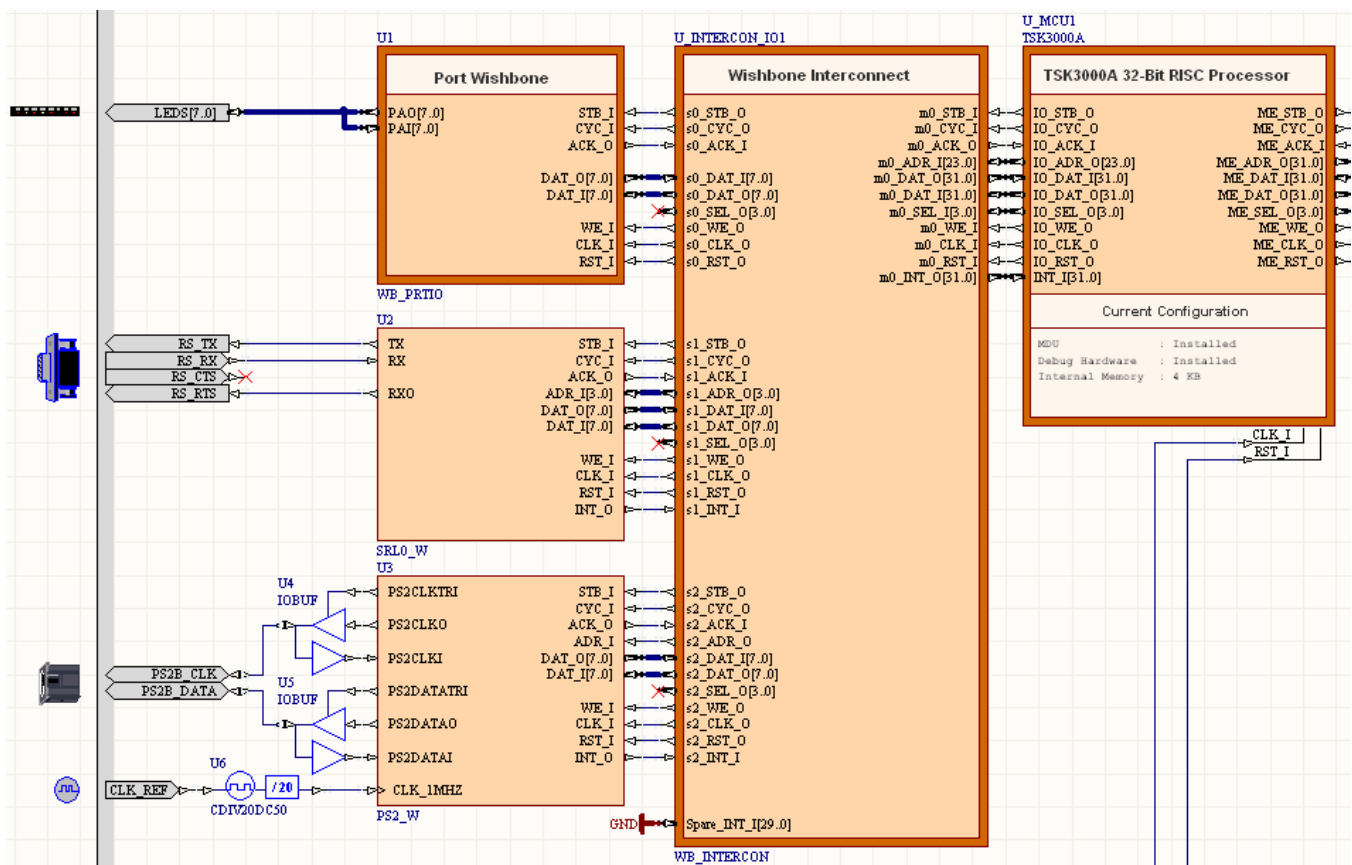


Figure 8. Multiplexing the TSK3000A's peripheral I/O interface using a Wishbone Interconnect device

Dual-Mastering

Some designs may require shared access to one or more slave memory or peripheral devices. This can be achieved by using both a configurable Wishbone Dual Master device (WB_DUALMASTER) and a configurable Wishbone Interconnect device (WB_INTERCON).

This makes it possible to connect two processor masters to a whole bank of slave memory or peripheral devices. The devices would be mapped into the respective processor address spaces at identical locations. Figure 9 shows an example of using both a Wishbone Dual Master device and a Wishbone Interconnect device to allow two TSK3000A processors to access a variety of physical slave memory devices.

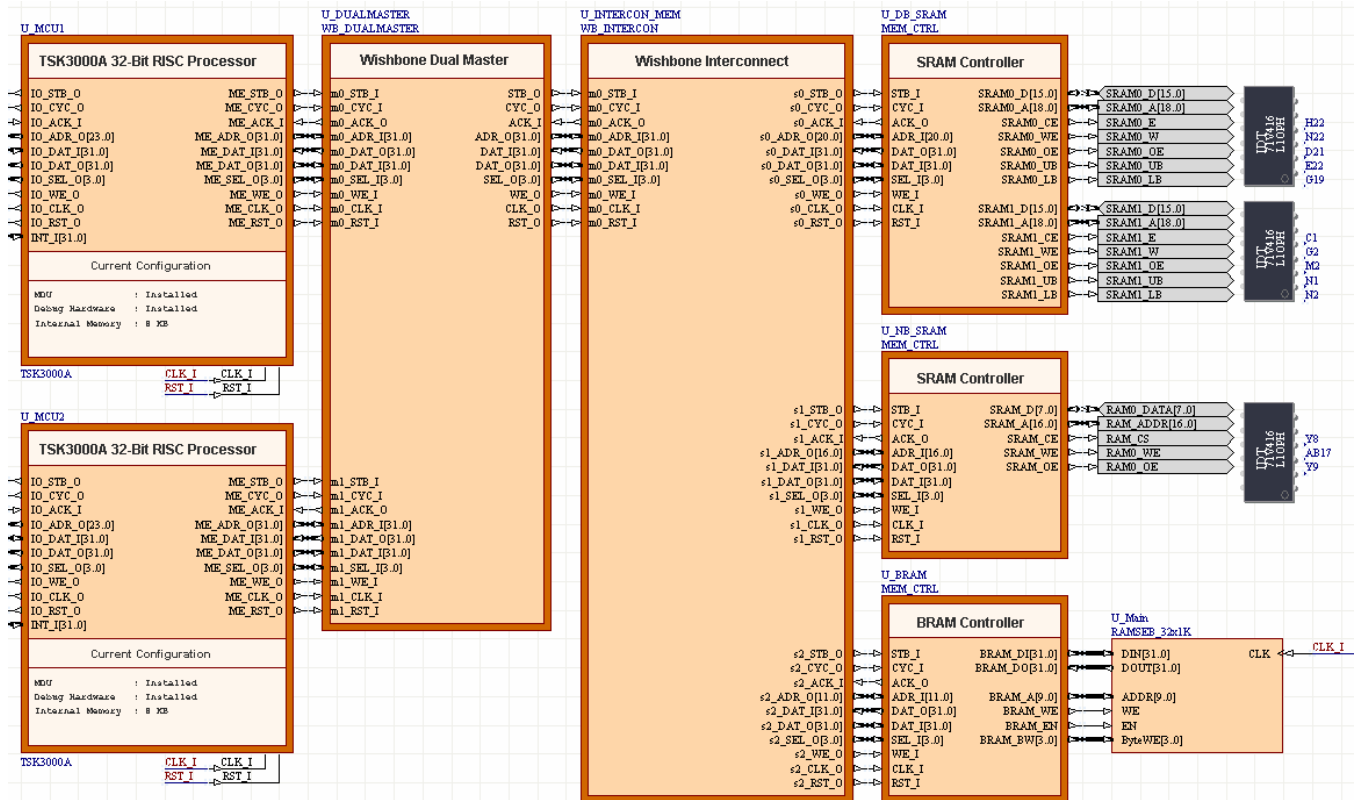


Figure 9. Sharing multiple slave memory devices between two processors

Figure 10 shows an example of using both a Wishbone Dual Master device and a Wishbone Interconnect device to allow two TSK3000A processors to access three different slave peripheral devices (two parallel port units and an Ethernet Media Access Controller).

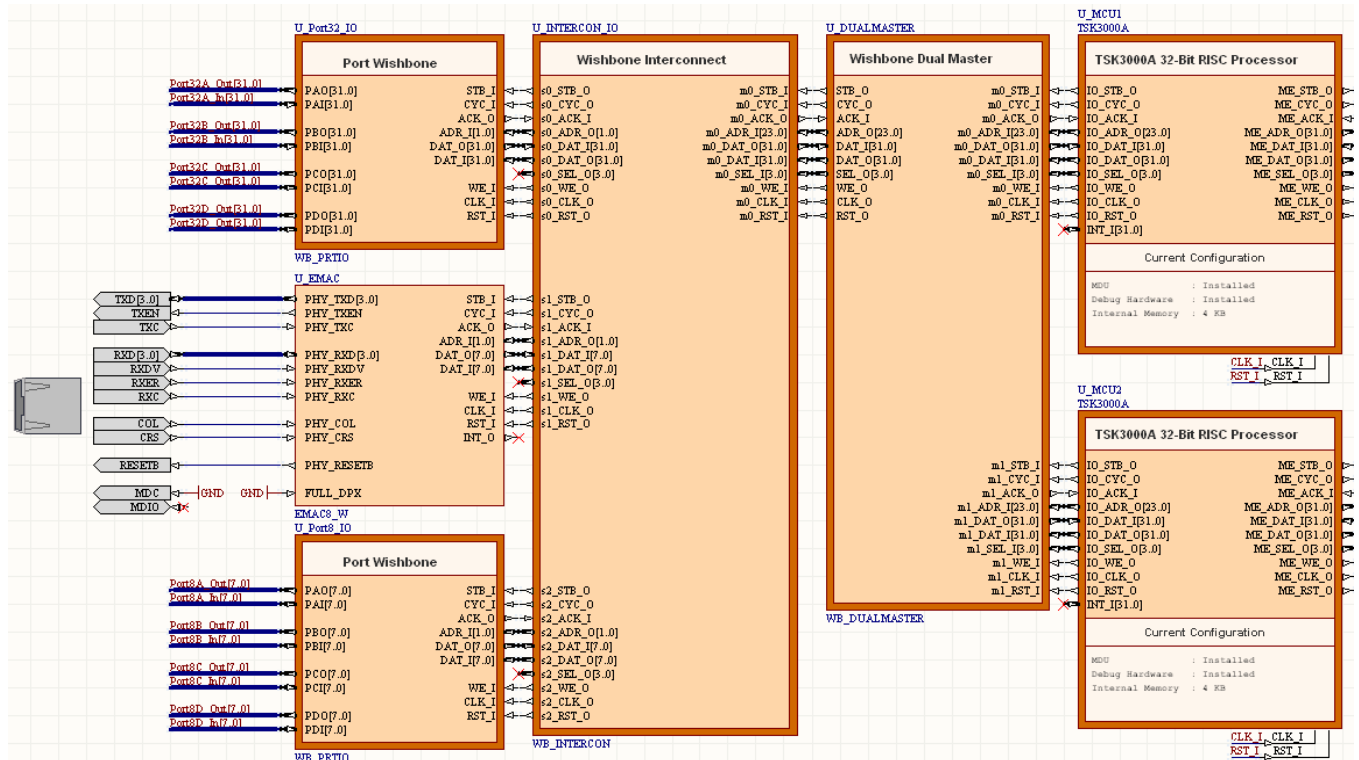


Figure 10. Sharing multiple slave peripheral devices between two processors.

For further information on the Wishbone Dual Master peripheral, refer to the [WB_DUALMASTER Configurable Wishbone Dual Master](#) core reference.

Sharing Peripheral Devices

Although the Wishbone Dual Master device can be used to share peripheral devices between two processors, it cannot provide marshalling for interrupts from the peripherals through to those processors. If the peripheral devices being shared do not generate interrupts, or they are not being used, then use of a Wishbone Dual Master – in series with a Wishbone Interconnect – is fine. In Figure 10 (previous), the port units do not generate interrupts and the interrupt from the EMAC is not being used.

If you need to share peripherals between processors, then you should consider using a Wishbone Multi-Master device. It can be configured to have between two and eight masters but, more importantly, it can pass interrupts from a connected Wishbone Interconnect through to all connected 32-bit processors. This makes it ideal for use on the peripheral side, when multiple 32-bit processors require shared access to a block of peripheral devices, and one or more of those devices generate interrupts.

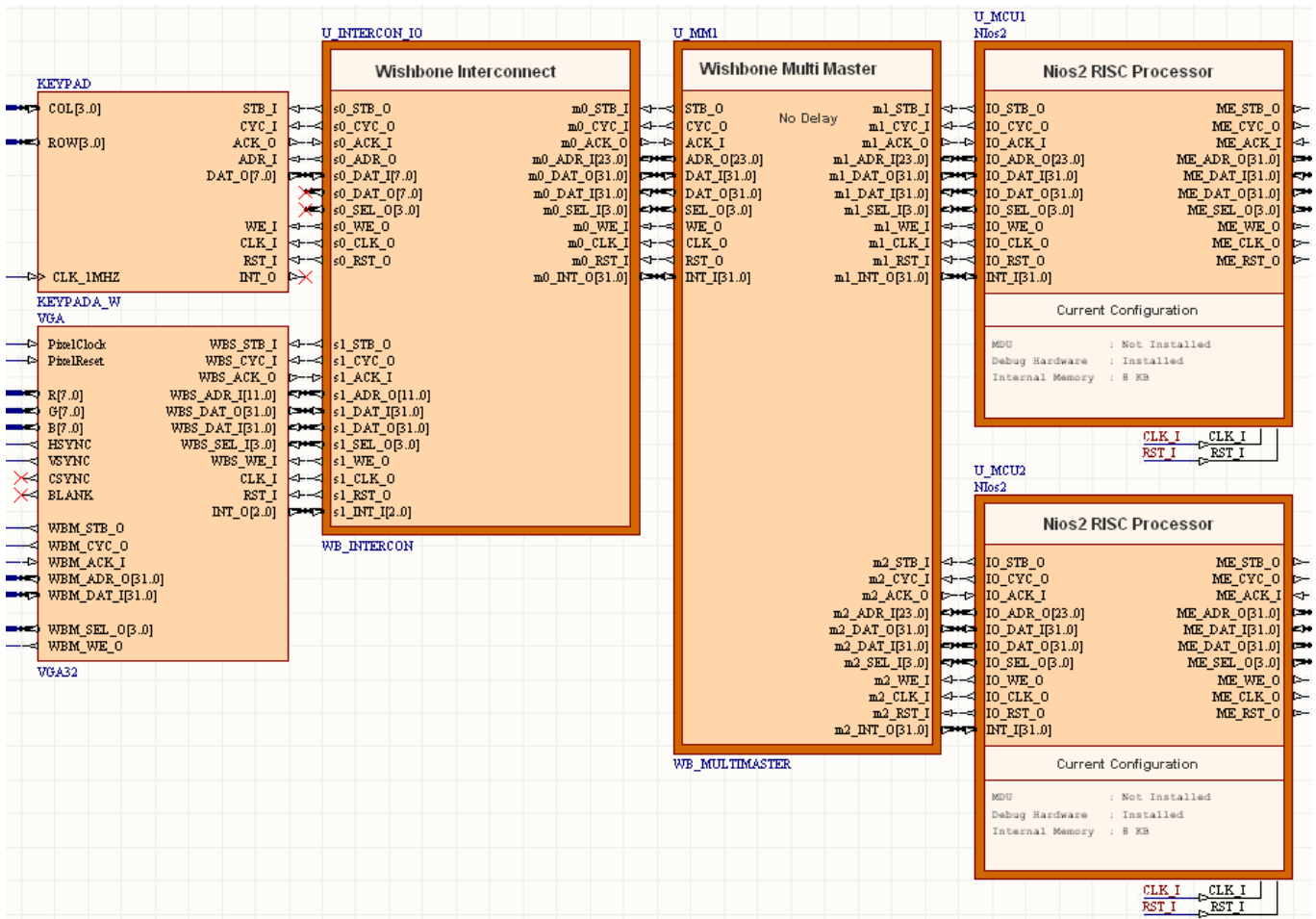


Figure 11. Sharing peripheral devices between processors using a Wishbone Multi-Master.

For further information on the Wishbone Multi-Master peripheral, refer to the [WB_MULTIMASTER Configurable Wishbone Multi-Master](#) core reference.

Revision History

Date	Version No.	Revision
10-Dec-2004	1.0	New release
23-Jun-2005	1.1	Updated for Altium Designer SP4
12-Dec-2005	1.2	Path references updated for Altium Designer 6
20-Sep-2006	1.3	Updated for Altium Designer 6.6.
12-Mar-2008	2.0	Updated for Altium Designer Summer 08

Software, hardware, documentation and related materials:

Copyright © 2008 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, Altium Designer, Board Insight, Design Explorer, DXP, LiveDesign, NanoBoard, NanoTalk, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.