WISHBONE COMPATIBLE

Combining WISHBONE interface signals

Application note

Author: Richard Herveille
Rherveille@OpenCores.org

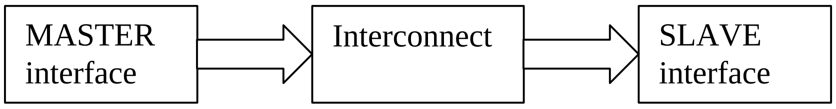
Rev. 0.2
April 18, 2001

Revision History

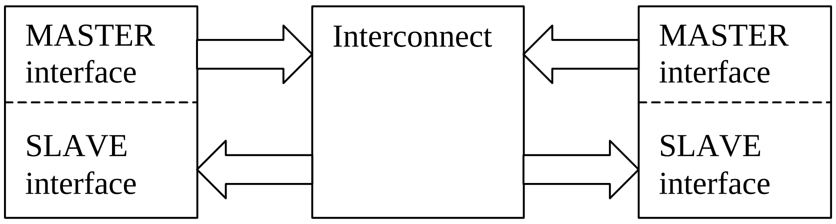
Rev.	Date	Author	Description
0.1	17/04/01	Richard Herveille	First Draft
0.2	18/04/01	Richard Herveille	Changed section 2.2 Added VHDL code and timing diagrams

1. Introduction

The WISHBONE System-on-Chip interconnect defines two types of interfaces, called MASTER and SLAVE. MASTER interfaces are cores that are capable of generating bus cycles, SLAVE interfaces are cores that are capable of receiving bus cycles. Connections between the MASTER and SLAVE interfaces can be established in a number of ways, from as simple as a point-to-point connection to as complex as traffic matrices with bus arbitration. But how complex or simple the interconnect may be, there is always a MASTER transferring data to/from a SLAVE.



There are situations where data has to be exchanged between cores that are both MASTER and SLAVE interfaces as shown in the figure below. In this case the theoretical bandwidth is double the amount of the figure above, at the expense of increased complexity caused by the additional interfaces and interconnect-signals.



In large designs these additional interconnect signals could cause routing problems. If the increased bandwidth is not needed than a substantial reduction in the number of interconnect signals can be achieved by combining common signals between MASTER and SLAVE interfaces.

2. Combinable signals

All WISHBONE signals can be combined between the MASTER and SLAVE interfaces, each at its own expense. The WISHBONE signals can be divided into three groups, common signals, data signals, and bus cycle signals.

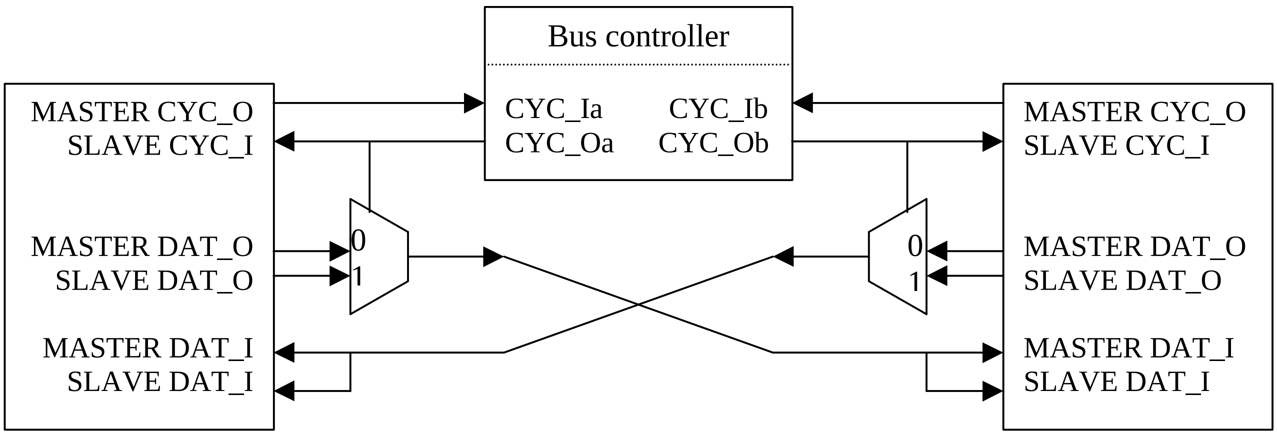
1) Common signals:
   - CLK_I
   - RST_I
   - TAG_I/O
2) Data signals
   - DAT_I/O
3) Bus Cycle signals
   - ACK_I/O
   - ADR_I/O
   - CYC_I/O
   - ERR_I/O
   - SEL_I/O
   - RTY_I/O
   - STB_I/O
   - WE_I/O

2.1 Common Signals

Combining the first group is obvious, since they are common to both MASTER and SLAVE interfaces, no additional logic is required.

2.2 Data signals

Combining the data signals decreases the number of interconnect signals substantially. In case of a 32bit data bus it requires 64 signals less. The [DAT_I] signal array of the MASTER and SLAVE interfaces can be combined without any additional logic, because both are inputs there is not driver issue to resolve. The [DAT_O] signal array of the MASTER and SLAVE interfaces can be combined at the expense of little additional logic; either a bus-multiplexor or internal tri-state buffers. The multiplexor or tri-state buffers are controlled by a statemachine that checks both MASTER interfaces' [CYC_O] signals. When a MASTER asserts ('1') its [CYC_O] signal, its MASTER [DAT_O] signal array should drive the bus, whereas the other core should drive the bus with its SLAVE [DAT_O] signals array. When both MASTER interfaces assert their [CYC_O] signal one should be selected as a MASTER, whereas the other should be selected as a SLAVE. How this selection is made is system dependent. Many solutions are possible, for example selection by priority or a round-robin scheme.

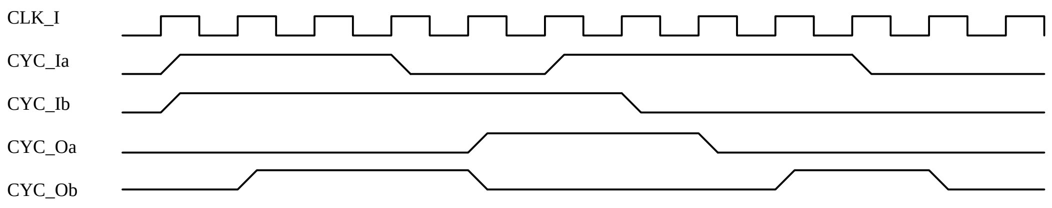


The figure above shows how to connect the cores. The Bus Controller controls which core is selected as a MASTER and which core is selected as a SLAVE interface. The SLAVE interfaces' [CYC_I] signals control the multiplexors states. When the [CYC_I] signals is asserted ('1'), the core is selected as a SLAVE interface and the SLAVE [DAT_O] signals array should drive the bus. When the [CYC_I] signal is negated ('0') the core is selected as a MASTER and the MASTER [DAT_O] signals array may drive the bus.

Listed below is the code for a simple priority based bus controlled, including the bus multiplexors.

```
library ieee;
use ieee.std_logic_1164.all;

entity bus_controller is
        port(
                CLK_I           : in std_logic;                          -- WISHBONE clock input

                CYC_Ia          : in std_logic;                          -- core A MASTER CYC_O output
                CYC_Oa          : buffer std_logic;                      -- core A SLAVE CYC_I input
                MDAT_Oa         : in std_logic_vector(31 downto 0);      -- core A MASTER DAT_O array
                SDAT_Oa         : in std_logic_vector(31 downto 0);      -- core A SLAVE DAT_O array
                DAT_Oa          : out std_logic_vector(31 downto 0);     -- core A combined DAT_O array

                CYC_Ib          : in std_logic;                          -- core B MASTER CYC_O output
                CYC_Ob          : buffer std_logic;                      -- core B SLAVE CYC_I input
                MDAT_Ob         : in std_logic_vector(31 downto 0);      -- core B MASTER DAT_O array
                SDAT_Ob         : in std_logic_vector(31 downto 0);      -- core B SLAVE DAT_O array
                DAT_Ob          : out std_logic_vector(31 downto 0);     -- core B combined DAT_O array
        );
end entity bus_controller;

architecture dataflow of bus_controller is
begin
        -- generate CYC_O signals
        gen_cyco: process(CLK_I)
        begin
                if (CLK_I'event and CLK_I = '1') then
                        CYC_Oa <= CYC_Ib and (not CYC_Ia or CYC_Oa);
                        CYC_Ob <= CYC_Ia and not CYC_Oa;
                end if;
        end process gen_cyco;
        -- assign combined DAT_O signals
        DAT_Oa <= SDAT_Oa when (CYC_Oa = ‘1’) else MDAT_Oa;
        DAT_Ob <= SDAT_Ob when (CYC_Ob = ‘1’) else MDAT_Ob;
end architecture dataflow;
```

The figure below shows some timing diagrams displaying the MASTER/SLAVE selection of the cores. Note that when both MASTER interfaces assert (‘1’) their [CYC_O] signal at the same time, coreB is selected as the SLAVE interface ([CYC_Ob] is asserted), i.e. coreA has the highest priority. The SLAVE interface remains selected until the MASTER interface negates (‘0’) its [CYC_O] signal.



2.3 Bus Cycle signals

By inserting tri-state buffers almost all bus cycle signals can be combined. But this is only useful when doing a point-to-point connection, because it would interfere with the system architecture to much otherwise. In the case of a point-to-point connected the question should be raised if it is worth the overhead. Perhaps it is better to redesign one core into a SLAVE only and thus automatically reduce the amount of required interconnect signals.

3 Conclusion

WISHBONE signals on modules implementing both MASTER and SLAVE interfaces can be combined to reduce the amount of required interconnect signals. The [CLK_I], [RST_I] and [TAG_I/O] signals can be combined without any additional logic. The [DAT_I] and [DAT_O] signal arrays can be shared between the MASTER and SLAVE interfaces. Sharing the [DAT_I] signal array costs no additional logic, sharing the [DAT_O] signals array costs only minor additional logic. Thus providing an easy, low cost solution to possible routing problems. Other signals could also be shared, but either the impact on the system architecture is to big or it is worth considering to redesign cores into SLAVE only modules.