# INTRODUCTION

The following document introduces wishbone interface, which is commonly used by IP cores available at opencores.com. Since the wishbone interface is not supported on neither Altera nor Xilinx platforms (at the time of writing this document) an adapter is required. As for the current project, the Altera Cyclone V SoC is device of choice, the wishbone interface is discussed in relation to avalon interface. At the moment, the use of wishbone interface is limited to the use of I2C\_master IP core (available at https://opencores.org/project,i2c or in source\_code folder), only the following signals are required to be incorporated by the adapter.

-- wishbone signals

wb\_clk\_i : in std\_logic; -- master clock input

wb\_rst\_i : in std\_logic := '0'; -- synchronous active high reset

~~arst\_i : in std\_logic := not ARST\_LVL; -- asynchronous reset (NOT WB INTERFACE)~~

wb\_adr\_i : in std\_logic\_vector(2 downto 0); -- lower address bits

wb\_dat\_i : in std\_logic\_vector(7 downto 0); -- Databus input

wb\_dat\_o : out std\_logic\_vector(7 downto 0); -- Databus output

wb\_we\_i : in std\_logic; -- Write enable input

wb\_stb\_i : in std\_logic; -- Strobe signals / core select signal

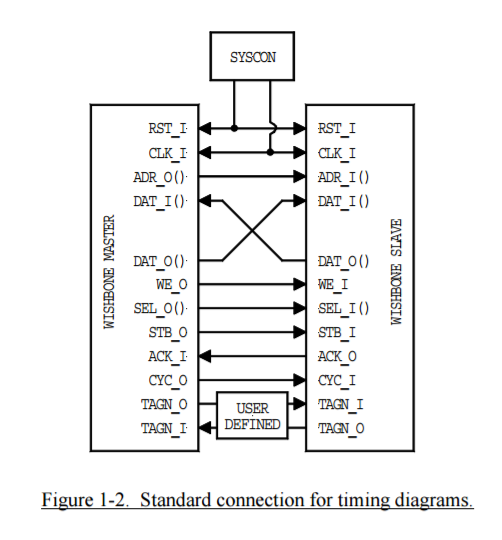
wb\_cyc\_i : in std\_logic; -- Valid bus cycle input

wb\_ack\_o : out std\_logic; -- Bus cycle acknowledge output

wb\_inta\_o : out std\_logic; -- interrupt request output signal

*snippet of VHDL code from source\_code/i2c/vhdl/i2c\_master\_top.vhd*

The a regular wishbone interface uses the signals as shown in the following figure:



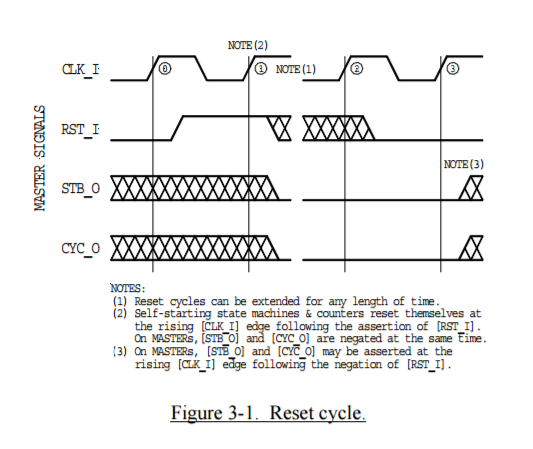
# WISHBONE INTERFACE

**CLK\_I**

The clock input [CLK\_I] coordinates all activities for the internal logic within the WISHBONE interconnect. All WISHBONE output signals are registered at the rising edge of [CLK\_I]. All WISHBONE input signals are stable before the rising edge of [CLK\_I].

**RST\_I**

The reset input [RST\_I] forces the WISHBONE interface to restart. Furthermore, all internal self-starting state machines will be forced into an initial state. This signal only resets the WISHBONE interface. It is not required to reset other parts of an IP core (although it may be used that way).

****

**ADR\_I()**

The address input array [ADR\_I()] is used to pass a binary address. The higher array boundary is specific to the address width of the core, and the lower array boundary is determined by the data port size. For example the array size on a 32-bit data port with BYTE granularity is [ADR\_O(n..2)]. In some cases (such as FIFO interfaces) the array may not be present on the interface.

**DAT\_I()**

The data input array [DAT\_I()] is used to pass binary data. The array boundaries are determined by the port size, with a maximum port size of 64-bits (e.g. [DAT\_I(63..0)]). Also see the [DAT\_O()] and [SEL\_O()] signal descriptions.

**DAT\_O()**

The data output array [DAT\_O()] is used to pass binary data. The array boundaries are determined by the port size, with a maximum port size of 64-bits (e.g. [DAT\_I(63..0)]). Also see the [DAT\_I()] and [SEL\_O()] signal descriptions.

**WE\_I**

The write enable input [WE\_I] indicates whether the current local bus cycle is a READ or WRITE cycle. The signal is negated during READ cycles, and is asserted during WRITE cycles.

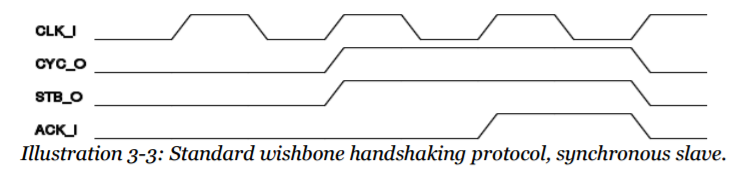
**STB\_I**

The strobe input [STB\_I], when asserted, indicates that the SLAVE is selected. A SLAVE shall respond to other WISHBONE signals only when this [STB\_I] is asserted (except for the [RST\_I] signal which should always be responded to). The SLAVE asserts either the [ACK\_O], [ERR\_O] or [RTY\_O] signals in response to every assertion of the [STB\_I] signal.

**CYC\_I**

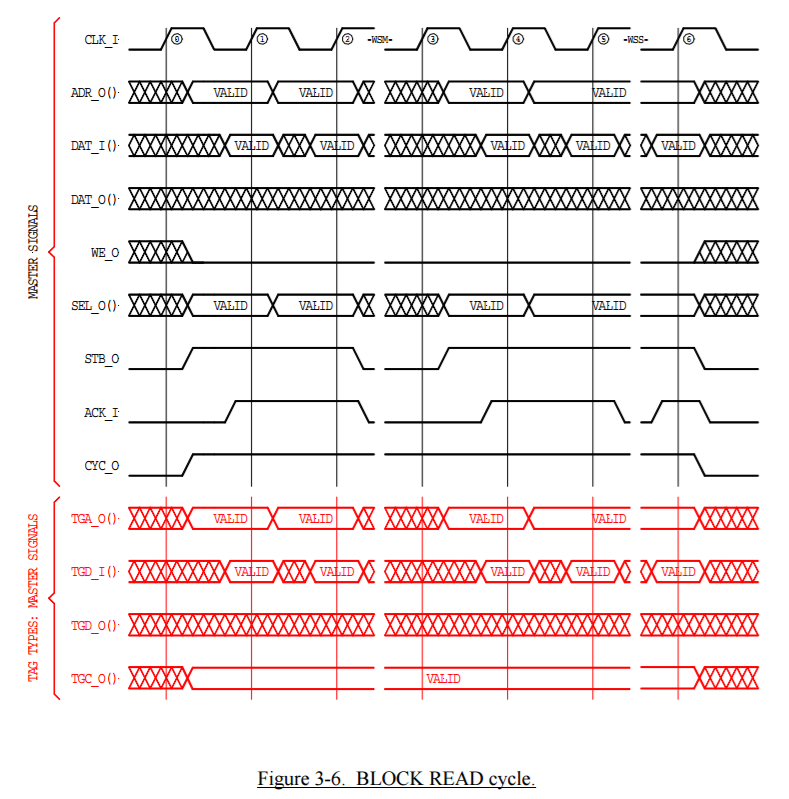
The cycle input [CYC\_I], when asserted, indicates that a valid bus cycle is in progress. The signal is asserted for the duration of all bus cycles. For example, during a BLOCK transfer cycle there can be multiple data transfers. The [CYC\_I] signal is asserted during the first data transfer, and remains asserted until the last data transfer.

**ACK\_O**

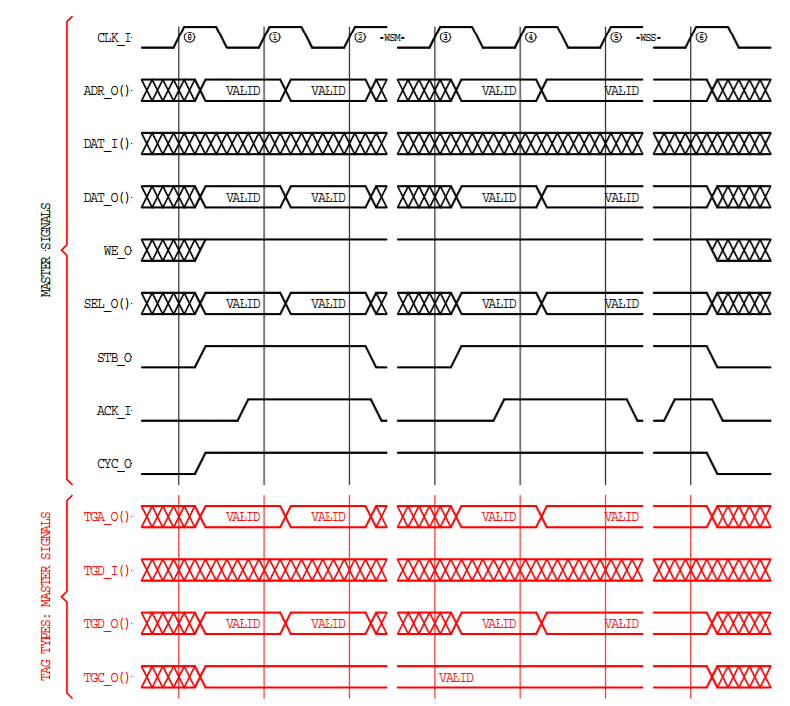
The acknowledge output [ACK\_O], when asserted, indicates the termination of a normal bus cycle. Also see the [ERR\_O] and [RTY\_O] signal descriptions. (*see* ***STB\_I*** *signal*)

## WISHBONE COMMUNICATION

Read operation



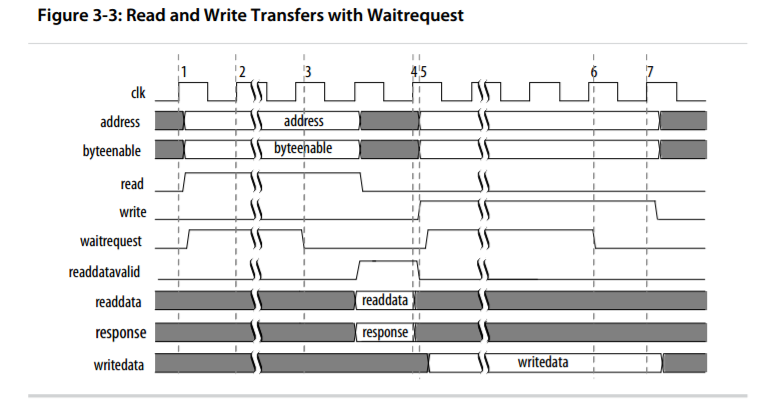
Write operation



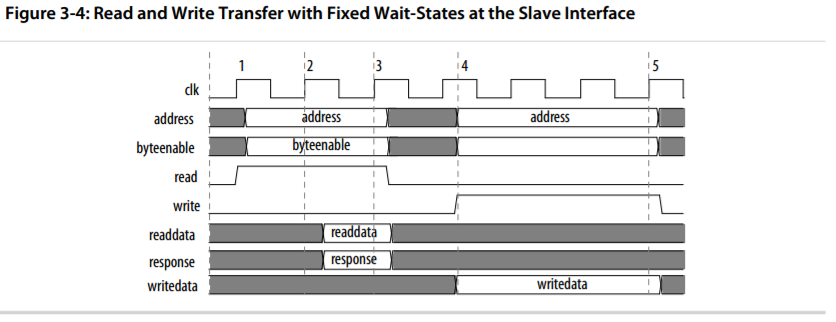
# AVALON INTERFACE

The following text briefly introduces Avalon interface. Due to the intended adapter application only the necessary signals are introduced.

An example of communication with *waitrequest* signal is introduced below:



An example of communication without *waitrequest* signal is introduced below. This implies the slave is always able to servis the communication on the following cycle after the *address*, *read* (or *write*) are asserted.



Utilized Avalon interface signals

clk -- clock signal

rst -- synchronous signal, active high

address -- address of memory-mapped interface

readdata -- data bus signal, input to master

writedata -- data bus signal, output from master

read -- begin read transaction

write -- begin write transaction

waitrequest -- slave postponing the current transaction (optional)

irq -- interrup

*waitrequest* is asserted by slave, when it is not able to process the master control signals. The master is not prevented by waitrequest from starting the communication, but may continue only when waitrequest is low. The waitrequest should be forced low on reset to prevent system lockup.

*byteenable* corresponds to writedata bus and readdata bus, and is used to select the bytes which are being written to or read from ... this concept is not useful for current application and therefore is not utilized

*readdatavalid* shows whether there are valid data asserted to readdata bus ... this is not important for the application as well, therefore it is omitted

*response* shows the status of current transaction, whether it is successful or an error occurred ... the current application doesnt require this behavior

*chipselect* is deprecated signal, and can be fully substituted by chipselect<=>(read OR write)

# ADAPTER LOGIC

|  |  |
| --- | --- |
| wishbone -> avalon |  |
| wishbone | avalon |
| clk | clk |
| rst | rst |
| adr | address |
| dat\_o | writedata |
| dat\_i | readdata |
| cyc | = write OR read |
| stb | = write OR read |
| we | = write AND !read |
| ack | = !waitrequest |
| inta | = irq |

|  |  |
| --- | --- |
| avalon -> wishbone |  |
| avalon | wishbone |
| clk | clk |
| rst | rst |
| address | adr |
| writedata | dat\_i |
| readdata | dat\_o |
| write | = cyc AND we |
| read | = cyc AND !we |
| waitrequest | = !ack |
| irq | = inta |