ECE 4401 (Fall 2014)
Lab 4 – Wishbone Bus

In this lab you will design a Wishbone bus interface to your 7 segment LED display and the switches. You will use the given Wishbone bus controller and design master and slave interfaces for the bus. The rest of this handout describes the bus controller.

Wishbone Bus:

Wishbone is an open SoC bus that can be used in semiconductor IP cores. The bus can be configured as point-to-point or a shared interconnection network. In this lab, we will assume shared bus architecture. The **wb_intercon** module provides the bus to which you will connect your design modules. The bus supports 4 masters and 4 slaves (this can be easily modified by changing the wb_intercon.vhd file). Arbitration between masters is done using a round-robin scheduler. Arbitration is not fair - in other words, if a master does not relinquish the bus, no other masters can get control of the bus. The entity declaration for wb_intercon is as follows:

```
entity wb_intercon is
       generic  (num_addr_bits : positive := 32; num_data_bits : positive := 32);
    port (
          ACK_I_M:    out std_logic_vector(3 downto 0);
          ACK_O_S:    in std_logic_vector(3 downto 0);
          ADR_O_M0:   in std_logic_vector( num_addr_bits-1 downto 0 );
          ADR_O_M1:   in std_logic_vector( num_addr_bits-1 downto 0 );
          ADR_O_M2:   in std_logic_vector( num_addr_bits-1 downto 0 );
          ADR_O_M3:   in std_logic_vector( num_addr_bits-1 downto 0 );
          ADR_I_S:    out std_logic_vector( num_addr_bits-1 downto 0 );
          CYC_O_M:    in std_logic_vector(3 downto 0);
          DAT_O_M0:   in std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_M1:   in std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_M2:   in std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_M3:   in std_logic_vector( num_data_bits-1 downto 0 );
          DWR:        out std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_S0:   in std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_S1:   in std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_S2:   in std_logic_vector( num_data_bits-1 downto 0 );
          DAT_O_S3:   in std_logic_vector( num_data_bits-1 downto 0 );
          DRD:        out std_logic_vector( num_data_bits-1 downto 0 );
          STB_I_S:    out std_logic_vector(3 downto 0);
          STB_O_M:    in std_logic_vector(3 downto 0);
          WE_O_M:     in std_logic_vector(3 downto 0);
          WE:         out std_logic;
          CLK:        in std_logic;
          RST:        in std_logic
        );
end entity wb_intercon;
```

Master modules will have entity declarations of the following form:

```
entity master is
 Port ( clk_i : in std_logic;
        rst_i : in std_logic;
        adr_o : out std_logic_vector(31 downto 0);
        dat_i : in std_logic_vector(31 downto 0);
        dat_o : out std_logic_vector(31 downto 0);
        ack_i : in std_logic;
        cyc_o : out std_logic;
        stb_o : out std_logic;
        we_o  : out std_logic);
end master;
```

The slave modules will have entity declarations of the following form:

```
entity wb_slave is
 Port ( clk_i : in std_logic;
         rst_i : in std_logic;
        adr_i : in std_logic_vector(31 downto 0);
         dat_i : in std_logic_vector(31 downto 0);
         dat_o : out std_logic_vector(31 downto 0);
         ack_o : out std_logic;
         stb_i : in std_logic;
         we_i  : in std_logic);
end wb_slave;
```

Note that additional signals to connect to the FPGA pins can be added to the master or slave if necessary. Each master or slave module is connected to the wb_intercon module. The DWR signal from the wb_intercon module connects to the DAT_I signal on the slaves and the DRD signal from the wb_intercon module connects to the DAT_I signal on the master.

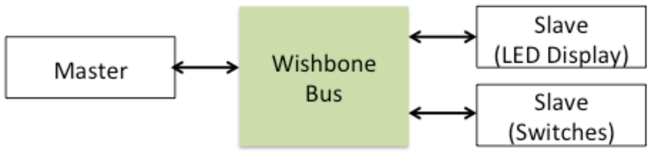
<u>Bus Cycle Description:</u>

When a master wants to read from a slave, it places the address of the slave on the ADR_O lines. The slave is addressed using the top two bits of the address. Thus address 0x00000000 refers to slave 0, address 0x40000000 to slave 1, 0x80000000 to slave 2, and 0xC0000000. In addition to placing the address, the master will assert its CYC_O signal to indicate arbitration should start if needed, and STB_O signal to indicate that it is starting a data cycle. The master will also set its WE_O to "0" to indicate a read and "1" to indicate a write request. The slave will know that it is being addressed when its STB_I signal becomes active. Since the WE_I line will be "0", it is a read and the slave will put the data on the DAT_O lines and assert the ACK_O signal. Upon receipt of the ACK_I signal, the master signals the end of data cycle by de-asserting the STB_O signal. The bus can be released to another master by de-asserting the CYC_O signal.

A write cycle is very similar. The master places the address for the slave on the ADR_O signals and asserts its CYC_O signal to indicate arbitration should start if needed and its STB_O line to indicate that it is starting a data cycle. The master also sets WE_O to "1" to indicate a write. The slave will know that it is being addressed when its STB_I signal becomes active. Since the WE_I signal will be "1", it is a write and the slave will latch the data on its DAT_I signals and assert the ACK_O signal when the write is complete. Upon receipt of the ACK_I signal, the master signals the end of the data cycle by de-asserting the STB_O signal. The bus is released for another master by de-asserting the CYC_O signal.

Goals For This Week:

Get started with the Wishbone bus, by designing two simple slave interfaces and one simple master module, as shown in figure below.



Design Wishbone bus interfaces for the switches and for the 7-segment LED display. Whenever the master module performs a read from the switch module, it should return the data on the switches in the lower eight bits of the DAT_O signal. Writing to the switch module is not supported. For this part you will update the **wb_swts.vhd** file.

Likewise, whenever the master module writes to the 7-segment LED module, the lower 16 bits of DAT_I must be displayed on the four 7-segment LEDs. In this slave module, reads are not supported. For this part you will update the **wb_segled.vhd** file.

Finally, you will need to design the master module functionality that reads from the switch slave module and writes the output to the 7-segment LED display slave module. For this part you will update the **reader.vhd** file.

Notes:

- The library modules you need for this lab are,
    - clock_divider
    - wb_intercon
    - wb_arbiter
    - word2leds (which uses hex2led and char_led_control)

Signal	**Width**	**Description**
CLK	1	System clock
RST	1	System reset
ACK_O_S	4	When asserted, indicates the termination of a normal bus cycle
ADR_O_M0	32	Used to pass binary address – Master 0
ADR_O_M1	32	Used to pass binary address – Master 1
ADR_O_M2	32	Used to pass binary address – Master 2
ADR_O_M3	32	Used to pass binary address – Master 3
CYC_O_M	4	The cycle output, when asserted, indicates that a valid bus cycle is in progress. The signal is asserted for the duration of all bus cycles
DAT_O_M0	32	Used to pass binary data – Master 0
DAT_O_M1	32	Used to pass binary data – Master 1
DAT_O_M2	32	Used to pass binary data – Master 2
DAT_O_M3	32	Used to pass binary data – Master 3
DAT_O_S0	32	Used to pass binary data – Slave 0
DAT_O_S1	32	Used to pass binary data – Slave 1
DAT_O_S2	32	Used to pass binary data – Slave 2
DAT_O_S3	32	Used to pass binary data – Slave 3
STB_O_M	4	It indicates a valid data transfer cycle. It is used to qualify various other signals on the interface. The SLAVE asserts either the [ACK_I], [ERR_I] or [RTY_I] signals in response to every assertion of the [STB_O] signal
WE_O_M	4	It 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
WE	1	It 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
ACK_I_M	4	When asserted, indicates the termination of a normal bus cycle
ADR_I_S	32	Used to pass binary address – All SLAVES
DWR	32	Connects to data_in of SLAVES
DRD	32	Connects to data_in of MASTER
STB_I_S	4	When asserted, indicates that the SLAVE is selected. A SLAVE shall respond to other WISHBONE signals only when this [STB_I] is asserted
IRQ_O	1	Used to interrupt another module (master)
IRQ_I	1	Used to receive an interrupt from another module (slave)