VHDL File and Signal Naming Conventions

Version 0.1

1.0 Introduction

This document is an attempt at standardizing the way files and ports are named so that our source files can be easily read and understood.

2.0 Naming Conventions

2.1 Source Files

Source filenames should be the same as the entity name due to a limitation in Quartus II. The ATC Coding Style Guide (version 0.3, Dec. 2003) also requests that each file should contain only one entity – architecture pair.

2.2 Miscellaneous Files

Package filenames should have a "_pack" appended to the end of the name.

Testbench filenames should have a "tb_" appended to the beginning of the name.

2.3 Ports

A system of suffixes can be used to clearly indicate the function and type of a port:

Port Functions:

Load / Set	*_ld
Enable	*_en
Control Output (ie. from an FSM)	*_ctrl

Port Type:

Data Inputs	*_i
Data Outputs	*_o
Data (Bidirectional)	*_bi
Register Inputs	*_d
Register Outputs	*_q

If suffixes are combined, use port function first, then port type:

Examples:	buffer_en_o	Buffer enable output
	regdata_q	Data output from register

At the top-level for any given card, the above conventions should not apply. In this case, the port names should be the same as those in the design schematic diagrams.

Furthermore, ports connected to the system clock should be named “clk”, and ports connected to the system reset should be named “a_rst” if it is asynchronous, or “s_rst” if it is synchronous (no suffixes).

The “*” in the above listing are signal names that are chosen to describe the data going through the port as accurately and concisely as possible. Names should ideally be concise to make it easier to see the signal name in waveform viewers, etc.

Port names in top-level files should append the entity name to the beginning of each signal. This is to aid in debugging, when viewing signals from multiple blocks in the waveform viewer.

Finally, an active-low port can be indicated by appending “n_” to the beginning of the port name.

2.4 Internal Signals

The above port type conventions do not apply to internal (local) signals.

Where a signal is used to connect two components together (an interconnect signal), the signal name should be the source port name (including suffixes).

The exception is when you are port mapping slices of a vector into the component, in which case it may be better to give the overall vector a descriptive name rather than declare a separate signal for each slice and then concatenate them.

3.0 Formatting Guidelines

Signals in association lists (ie. a port map or generic map) should be vertically aligned and use named association to make them easier to read.

Signals in entity declarations should also be vertically aligned.

The ATC Coding Style Guide also requests that signal declarations appear one per line.

4.0 Sample Usage

```
entity test is
port(  clk : in std_logic;
       rst : in std_logic;
       serial_i : in std_logic;
       parallel_o : out std_logic_vector(3 downto 0)
);
end test;


architecture behave of test is

component test_fsm
port(  clk : in std_logic;
       rst : in std_logic;
       demux_sel_ctrl_o : out std_logic;
       buf_en_ctrl_o : out std_logic;
       buf_ld_ctrl_o : out std_logic
);
end component;

component reg
port(  clk : in std_logic;
       rst : in std_logic;
       regdata_d : in std_logic_vector(3 downto 0);
       regdata_q : out std_logic_vector(3 downto 0);
       reg_en_i : in std_logic;
       reg_ld_i : in std_logic
);
end component;

component demux
port(  sel_i : in std_logic_vector(1 downto 0);
       data0_o : out std_logic;
       data1_o : out std_logic;
       data2_o : out std_logic;
       data3_o : out std_logic;
       data_i : in std_logic
);
end component;

signal demux_sel_ctrl_o : std_logic;
signal buf_en_ctrl_o : std_logic;
signal buf_ld_ctrl_o : std_logic;
signal demux_data_o : std_logic_vector(3 downto 0);

begin
        u0: test_fsm
                port map(   clk => clk,
                            rst => rst,
                            demux_sel_ctrl_o => demux_sel_ctrl_o,
                            buf_en_ctrl_o => buf_en_ctrl_o,
                            buf_ld_ctrl_o => buf_ld_ctrl_o);
```

```
        u1: demux
                port map(   sel_i => demux_sel_ctrl_o,
                            data0_o => demux_data_o(0),
                            data1_o => demux_data_o(1),
                            data2_o => demux_data_o(2),
                            data3_o => demux_data_o(3),
                            data_i => serial_i);

        u2: reg
                port map(   clk => clk,
                            rst => rst,
                            regdata_d => demux_data_o,
                            regdata_q => parallel_o,
                            reg_en_i => buf_en_ctrl_o,
                            reg_ld_i => buf_ld_ctrl_o);

end behave;
```