BlazeRouter

Design Document

Krikor Hovasapian

Kareem Matariyeh

University of Nevada, Las Vegas

12.22.2010

**Revisions**

**Rev 0.1** 12.22.2010 – <KVH> Created initial document and framework.

**Coding Conventions Used**

The BlazeRouter design has used the following conventions for programming within the VHDL language. Please abide by these rules when adding, removing, or altering code. It’s imperative that continuation on this work, over the course of its existence, be done in a unified manner. All programmers must follow this guideline prior to submission into the BlazeRouter public repository.

**Your Identity**

Within the code, if you change a line of code, utilize a three letter initial that represents:

<FIRST NAME> <MIDDLE INITIAL> <LAST NAME>

Therefore, if your name is John Chauncey Holmes, then you mark your code with the following moniker:

**<JCH>**

Example:

-- <JCH> changed the value here from 8 to 16

**VHDL Keywords**

**Rule 1: *All keywords are written in lowercase*.**

The ISE editor highlights these keywords with the color “Blue.” Notice some of the VHDL Keywords/Terms defined below:

**library**

**entity**

**port**

**architecture**

**use**

**component**

**inout**

**begin**

**end**

**VHDL Constants**

**Rule 2: *All constants are written in capital letters. Constants can be alphanumeric and an underscore “\_” must be used to differentiate between two or more words.***

Example:

WIDTH

NORTH\_SIZE

BUS\_8BIT

**VHDL Literals**

**Rule 3: *All literals are written in lowercase. Literals must abide by VHDL conventions when defining your literal. Literals can be alphanumeric and an underscore “\_” must be used to differentiate between two or more words.***

Example:

north\_in

north\_out

**VHDL Data Types**

**Rule 4: *All data types are written in lower case letters.***

The ISE editor highlights with the color “Blue.” Notice some of the VHDL Keywords/Terms defined below:

**integer**

**array**

**bit\_vector**

**bit**

**VHDL Packages**

**Rule 5: *All package names declared in .vhd files are written in lower case letters.***

Example:

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

**Rule 6: *All user defined package names are written in lower case letters.***

Example:

use work.std\_logic\_unsigned.all;

**VHDL Comments**

We can’t enforce a rule for commenting, but it’s recommended that programmers use common sense when providing information about code that can’t explain itself. Don’t overdo it! Make sure what the code can’t say by itself is provided with an ample amount of comments so that others can understand what you were doing. A one or two line brief goes a long way in making complex implementations, legible.

**VHDL Module Metadata**

**Rule 7: *All metadata provided by the ISE template must be completed by the original programmer prior to coding. Additional programmers must include their names as well and the date they started. Revisions must be updated with each check-in to the public repository. The metadata is ONLY additive. Do not remove prior history.***

Metadata is very important and the Xilinx ISE software provides a generic template with each module that gets created (see below). We cannot stress the importance of filling out this information. This will help identify you to the community that is taking part in the development of this project, and it’s a measure of accountability. This way, if something goes awry in your code, questions come up during the lifetime of the code, there is a trail that leads us back to the source.

----------------------------------------------------------------------------------

-- Company: University of Nevada, Las Vegas

-- Engineer: Krikor Hovasapian (ECE Graduate Student)

-- Kareem Matariyeh (ECE Graduate Student)

-- Create Date: 16:16:07 10/06/2010

-- Design Name: BlazeRouter

-- Module Name: BlazeRouter - Behavioral

-- Project Name: BlazeRouter

-- Target Devices: xc4vsx35-10ff668

-- Tool versions: Using ISE 10.1.03

-- Description:

-- BlazeRouter is a hierarchical router that contains instances of

-- various VHDL modules responsible for the entire makeup of the

-- design. The idea is to connect the available ports (N,S,E,W) to

-- adjacent modules (i.e Processors) to realize an NoC.

-- Dependencies:

-- (See individual files for more details)

-- Arbiter.vhd - Contains the Routing and Arbitration Unit

-- Buffer.vhd - Contains a single 8 bit width buffer

-- LinkController.vhd - Contains the Link Controller

-- SwitchUnit.vhd - Contains the Switch mechanism

-- SimplePackages.vhd - All packages for BlazeRouter

-- Revision:

-- Revision 0.01 - File Created

-- Revision 0.02 - Added additional modules (KVH)

-- Additional Comments:

--

----------------------------------------------------------------------------------

**Router Design**



**VHDL Breakdown**

**Core Files:**

**BlazeRouter.vhd**

This is the top-level module that instantiates all the components necessary to make the internals of the BlazeRouter come to life.

**Arbiter.vhd**

The Arbiter is responsible for monitoring the status of each buffer within the router to see if any new data has arrived in a Round Robin with Priority scheme. When data arrives, a copy of the packet within the buffer is taken, analyzed by one of the routing algorithms and a result is forwarded to the RNA\_RESULT port instructing the switch on configuration.

**Buffer.vhd**

The Buffer is a First-In First-Out (FIFO) buffer that stores packets or flits of data from adjacent routers.

**LinkController.vhd**

The LinkController establishes and synchronizes with adjacent routers by asserting or rejecting incoming transfers through the (N,S,E,W) ports.

**SwitchUnit.vhd**

The SwitchUnit is responsible for establishing the connections from input to output ports and transferring data from input buffers to their respective output buffer as determined by the Routing and Arbitration unit, which will post a result for the switch unit to follow through the RNA\_RESULT port.

**Packages:**

**BlazePackage.vhd**

This is the default package for all constants, supplemental data types, and functions to be used universally throughout the project. Place reusable code in a package within this file.