5-Port Router For Network-On-Chip

This markdown file provides an overview of a 5-Port router design implemented in Verilog for NoC (Network-On–Chip) or NoP (Network-On–Packet). The router is responsible for handling communication between processing elements in the network.

Router Architecture

The 5-Port router consists of the following main components:

1. **Input Ports:** There are 5 input ports of width 32 to receive flits from 5 directions, neighbouring and local (*North_in, South_in, East_in, West_in, Local_in*) processing elements. Each input has a 32 bit buffer to store the input. Also there a five acknowledgement input from adjacent router (*bf_inp_north, bf_inp_south, bf_inp_west, bf_inp_east, bf_inp_local*) give information about whether it can sent flit to next router or not.

2. **Output Port:** Just like the input, there are 5 32 wide output to sent flits to its location (*North_out, South_out, East_out, West_out, Local_out*) and also 5 acknowledgement output too(*bf_out_north, bf_out_south, bf_out_west, bf_out_east, bf_out_local*) .

3. **Routing Logic:** The routing logic will determine the output port to which an incoming flit should be assigned. This router follows the YX routing algorithm. When a flit enters a router, it checks the Y index or the destination column first, if it matches then the X index.

4. **Control Logic:** When multiple input wants the same output port, there should be a prioritisation mechanism to select which one should go first. Here it is a predefined set of priority chart. Inside this router there will be a mode 5 counter, and for its each count priority for each input varies.Whenever one output is taken by an input a local flag for example *north_taken* will be set to high to avoid collision. After each clock all flags will be reset.

Count 0 N > S > E > W > L
Count 1 S > E > W > L > N
Count 2 E > W > L > N > S
Count 3 W > L > N > S > E
Count 4 L > N > S > E > W

Priority

5. **Switch Allocation:** A flit contains destination address in its first m bits, where m is log n to the base 2 and n equal to the size of noc. After location is determined, input is assigned to required output only if desired output is not taken and buffer input is not active.

Verilog Implementation

The router is implemented in verilog as a single module, with a 3 stage pipeline

1. **Ports:** This module consists of 5 32bit input and output port 5 buffer signal input and output port
2. **Parameters:** The parameters or variables include in this file are *BUS_WIDTH* define the size of input, *LOC_X* is the X index of router, *LOC_Y* is the Y index of router (*LOC_Y,LOC_X* form the address of router*), NOC_SIZE* is the size of noc in NxN format, *ADDR_SIZE* is the width of the address. It depends on the *NOC_SIZE*. *NORTH, SOUTH, WEST, EAST* and *LOCAL* are local parameters used to show which output this input should go.
3. **Pipeline:** This is a 3 stage pipeline system consisting of the first stage **Buffer writing**, receiving the input only if buffer acknowledgement is not active otherwise to buffer the last input. The next stage **Route computing**, from the destination address output route is computed in this stage. Last stage is **Switch Allocation** will assign the input to the computed route if the buffer is not active and route is not taken in that cycle. But to avoid the 3 clock delay of the system, we are using dual clock synchronisation.

Testbench Writing

The major things that should be remembered during writing a testbench for this routers are

- Since we are using a dual clock, both clocks should be 25% duty cycled and 180 degree out of phase signals.
- Before giving any input, a stable reset signal must be applied to *rst* port minimum for a complete clock cycle
- Initialise the buffer inputs to low to avoid unknown (X) stuck in buffer
- Better to initialise all input to 32 bit zero while the time of deactivating reset, packet missing may occur
- Traffic at output depends on the address of the router when you are giving random inputs.

Extending to a network

Using a python script which takes size of noc and number of simulation cycles as input will generate a verilog file for the NxN network of the router

and 2 testbench files for this network. The first one will provide continuous input irrespective of the number of simulation cycles and the second one only provides m number of input where m is user input.

References

- Advance computer architecture NPTEL course IIT Guwahati
- bakhshalipour/NoC-Verilog: A verilog implementation for Network-on-Chip (github.com)