Lab 3 Flow control

I. Purpose

In this Lab, you will learn how to modify the source code to change the flow control algorithm of our tool. Hope you will enjoy using our tool!

II. Introduction

● NoC Structure Hierarchy

In Access Noxim, the main program flow is in main.cpp. At first, we will construct the whole NoC. Tiles, which contain a router and a processing element, in NoC will also be set. The structure of our tool is in fig. 1. After main structure be set, the simulation starts. Process Elements generates packets and injects into the network by different traffic patterns. Routers route the packets in the networks based on the routing function, and select the best output channel by selection function. Access Noxim simulate the behavior of router accurately, and focus on the network layer simulation.

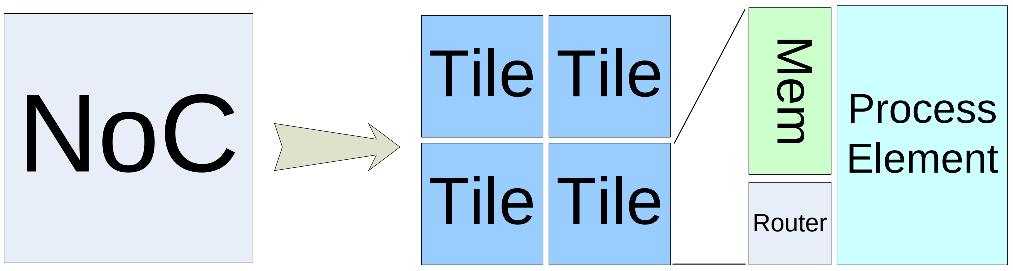

Figure 1 NoC Structure Hierarchy

The Lateral link between the tiles is declared in NoximNoC.cpp, while the vertical link is declare in NoximVlink.cpp. NoximVlink provides two kinds of vertical link: mesh-based and vertical crossbar. In mesh-based vertical link, the packet need to go through each node in vertical direction, while the vertical crossbar only take one-hop count to transmit the packet in vertical direction. If you're interesting in the vertical link, you can find more information in [1].

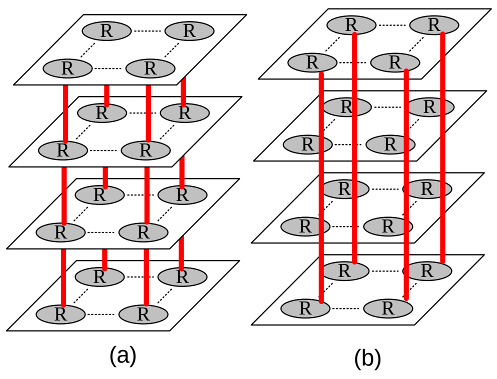

Figure 2 Vertical Link (a) mesh-based (b) vertical crossbar

● System C module

Maybe you've already learn some hardware description language, like Verilog. It's will let you quickly get familiar with the SystemC. Here we use a simple adder to show how we construct an easy module. In this example we use *sc_in* and *sc_out* to declare the input and output port. Whenever the clock signal appears, use the *sc_in_clk* to declare it. SystemC use *SC_CTOR* as the constructor of the module, you can declare the function is sensitive to positive of clock edge. In this example, the function *add()* will be execute only at the positive edge of clock signal. Thus we can simulate the behavior of the hardware. More tutorial is available in [2].

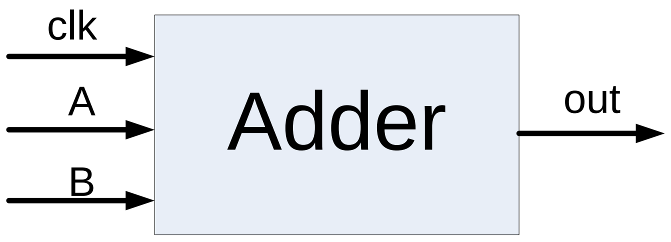

Figure 3 A simple adder

```
// All systemc code should include systemc.h file
#include "systemc.h"
SC_MODULE("adder_name") {
   // module body
   sc_in_clk     clock ;    // Clock input of the design
   sc_in<int>    A;         // Input A
   sc_in<int>    B;         // Input B
   sc_out<int>   out;       // Output out

   void add(){ out = A + B;};
    // Constructor for the counter
 SC_CTOR("adder_name") {
   SC_METHOD(add);
   sensitive << clock.pos();
 } // End of Constructor
}
```

III. Procedure

● Virtual Cut Through

1. Edit Flit struct
   a. Open NoximMain.h
   b. Add an int variable “packet_size” in flit struct

```
struct NoximFlit {
    int    packet_size;
```

2. Edit the TxProcess() of NoximRouter.cpp
   a. Open NoximRouter.h
   b. Observe the ports of Router
   c. Input port“free_slots_neighbor” is free buffer info. of neighbor

```
    sc_out < NoximNoP_data > NoP_data_out[DIRECTIONS];
    sc_in  < NoximNoP_data > NoP_data_in [DIRECTIONS];
```

   d. Open NoximRouter.cpp
   e. In TxProcess(), before flit reserve the channel, we should check if next buffer has enough space. Thus, edit the if-else loop as below.

```
if (  reservation_table.isAvailable(o) &&
     (free_slots_neighbor[o].read() >= flit.packet_size ||
       o == DIRECTION_LOCAL ) )
```

3. Edit Nextflit() in NoximProcessElement.cpp
   a. Open NoximProcessElement.cpp
   b. Edit Nextflit(), add the code below

```
    flit.packet_size = packet.size
```

4. Save those files, and recompile tool.

```
% cd <INSTALL DIRECTORY>/bin
% make clean;make
```

5. Print lantency-Injection rate diagram to show the performance difference between wormhole and virtual cut through. Thus we can verify our result is correct.

IV. Problems

● Flow control Comparison

Please compare the flow control mechanism between store-and-forward, wormhole and virtual cut through.

● Store-and-forward flow control

After you follow the procedure in this lab, please try to implement the store-and-forward flow control mechanism and verify it.

● Packet Size and Buffer Size

While we use different flow control algorithm, system performance may affect by packet size and buffer size. Please use tore-and-forward, wormhole and virtual cut through flow control in the environment describe below. 0.01

- ❖ Traffic: random, transpose1
- ❖ Routing: west-first
- ❖ Selection: random
- ❖ A:Same packet size(Packet size=4),
  different buffer size(Buffer size = 4, 6, 8, 10, 12)
- ❖ B:Same buffer size(Buffer size = 8),
  different packet size(Packet size=2, 4, 6, 8)

V. References

[1] J. Kim et al., "A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures," in Proc. Intl. Symp. Comput. Architecture (ISCA'07), June 2007, pp.138-149.

[2] SystemC Tutorial, available in http://www.asic-world.com/systemc/index.html,March 2012.