

Course Project
Part 2
Due Date: 6th December
Design of routing algorithms for Networks on Chip

1 Design Details

In the first part of the course project you have designed the combination of the FIFO buffers and the arbiter of the wormhole router, i.e., you have the part within the red circle of Figure 1. So at the output of the input MUX you have the flits coming from the channel, which has won the arbitration process. The next part of your work will be to determine the output port the winning flit has to follow. In other words you have to determine the destination of the incoming flit. It depends on the architecture of the Network on Chip (NoC) and the adopted routing mechanism.

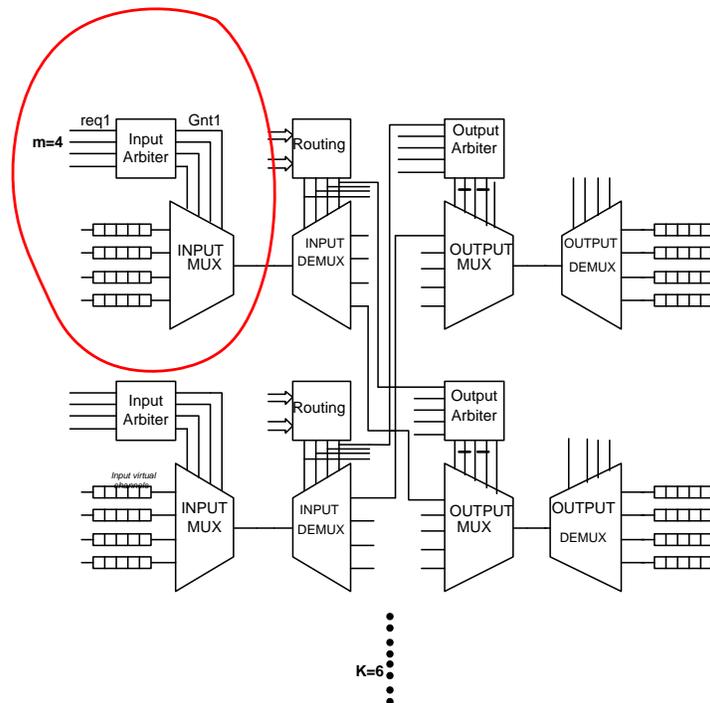


Figure 1: Block Diagram of a router

In the first part, consider a system with 16 cores interconnected through the MESH architecture as shown in Figure 2.

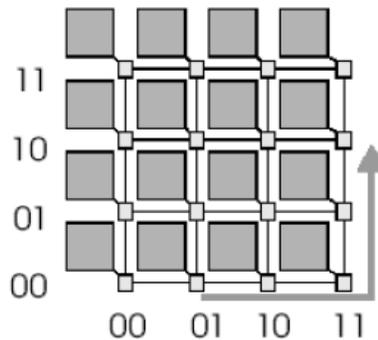


Figure 2: MESH-based NoC

As seen in the figure each wormhole router (switch) is either connected to two, three, or four neighboring routers (switches) and a single processing block. The flits are coming from the functional block (data in the FIFO). One of the possible flit structures is shown in Figure 3.

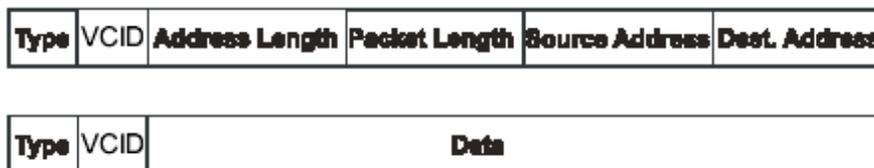


Figure 3 a) Header flit; b) Data and Tail flit

The header flit contains the source and destination addresses in addition to other fields. In reality you need an input logic controller to extract the source and destination addresses from the header and feed that to the routing block. But to make it simple assume that the source and destination addresses have been extracted already and is available as inputs to the routing block. Depending on the adopted routing algorithm, you need to determine the destination output port. There should be four possibilities: North, South, East and West.

First you need to determine the number of bits needed in the source and destination addresses. After that you have to implement the X-Y routing algorithm described in class. The algorithm is described here in the following flow diagram:

X-Y Routing Algorithm for 2D Mesh:

Inputs: Coordination of current node ($X_{\text{current}}, Y_{\text{current}}$), coordination of destination node ($X_{\text{dest}}, Y_{\text{dest}}$) and fault status of selected channel

Output: Selected output channel

Algorithm details:

$X_{\text{offset}} = X_{\text{dest}} - X_{\text{current}};$

$Y_{\text{offset}} = Y_{\text{dest}} - Y_{\text{current}};$

if $X_{\text{offset}} > 0$ then

 Channel: = X^+ ;

endif

if $X_{\text{offset}} < 0$ then

 Channel: = X^- ;

endif

if $X_{\text{offset}} = 0$ and $Y_{\text{offset}} > 0$ then

 Channel: = Y^+ ;

endif

if $X_{\text{offset}} = 0$ and $Y_{\text{offset}} < 0$ then

 Channel: = Y^- ;

endif

if $X_{\text{offset}} = 0$ and $Y_{\text{offset}} = 0$ then

 Channel: = local channel;

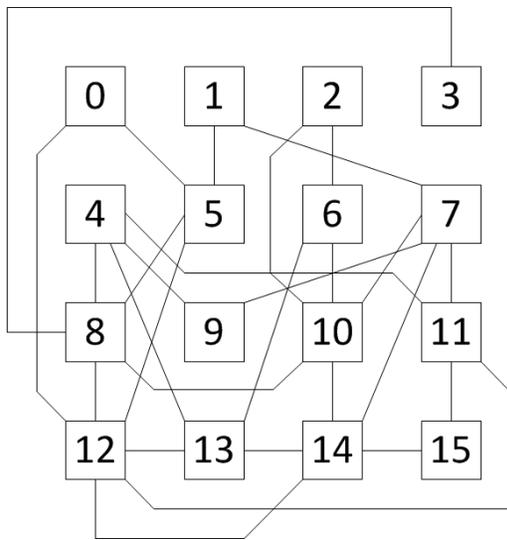
endif

Implement the X-Y routing algorithm and depending on the source and destination addresses determine the output port taken by the incoming flit. You need to set up your test bench in such a way so that you can demonstrate all the four possibilities for the destination port.

Next, assume that the same system with 16 cores is interconnected through an irregular (Small-World) architecture. The Small-World architecture is shown in Figure 4. In this architecture, the links are established following a power-law based connectivity. You must follow and implement the TRAIN routing algorithm described in [1]. Choose one of the valid trees shown in Table 1. For graduate students, please implement the TRAIN routing algorithm following the MROOTS model discussed in [2], and using at least 2 of the 4 trees shown in Table 2. Further discussion of the TRAIN algorithm is shown in the course_project_2 PowerPoint presentation. Please feel free to implement the TRAIN routing algorithm following your own method, (LUT-based, comparators, etc...). You need to set up the test bench in such a way so that you can show the routing for any pair of source and destination cores.

[1] H. Chi and C. Tang, "A deadlock-free routing scheme for interconnection networks with irregular topology," Proc. of ICPADS, pp. 88-95.

[2] O. Lysne, et al., "Layered routing in irregular networks," IEEE Trans. on Parallel and Distributed Systems, vol. 17, no. 1, 2006, pp. 51-65.



Switch #	Tree 1 Address	Tree 2 Address	Tree 3 Address	Tree 4 Address
0	510	310	111	1121
1	310	110	100	2110
2	410	210	310	0000
3	100	231	331	2210
4	200	410	210	1110
5	300	320	110	2220
6	420	220	320	1000
7	430	100	000	2100
8	000	230	330	2200
9	210	120	200	2120
10	400	200	300	2000
11	220	130	400	2130
12	500	300	410	1120
13	230	400	510	1100
14	440	000	500	1130
15	221	500	420	1131

Figure 4: SW architecture with 16 cores

Table 1: Valid TRAIN Routing Trees

2 Report

You are required to submit a 3-4 page report clearly showing how you have implemented each block. In your report, please compare the two architecture/routing methodologies implemented. Think about the advantages and disadvantages of each in terms of ideas discussed throughout the semester (layout, area overhead, power, latency, design complexity, test complexity). Find the average and maximum number of ports for both architectures. How do you feel each architecture would scale (i.e. do you foresee any issues with a 512-core design)? This discussion is mostly qualitative. Synthesized worst-case routing blocks and port numbers are the only values that should be discussed. You need to submit the VHDL/Verilog code also. If you work in a group of two then submit one report per group.

Each report should start with the following declaration signed by each student

“The submitted report is my own work. I/we have not copied the program from any other sources. If found guilty of copying then I/we will receive a grade of “zero” for this lab.”

3 Due Date

You need to demonstrate and submit your design by 6th December. There will be no extension to this due date in any circumstances. The group that creates the lowest-power implementation of the worst-case SW TRAIN routing block will receive an opportunity for something special on the final.