Architecture and Design of Core Network Interface for Distributed Routing in OASIS NoC

Ryuya Okada  s1160048  Supervised by Prof. Abderazek Ben Abdallah

Abstract

The main functions of a Network Interface (NI) are flitization, i.e., conversion of a packet into flits, and de-flitization, i.e., conversion of flits into a packet. They help data transmission from source core to destination one by adding some control information to packets. There are 2 types of NIs such as source routing NI and distributed routing NI. The former has a path information table for the complete route information. In the latter, a packet header is compact. We proposed an architecture and a design of a Core Network Interface (NI) for distributed routing on an Altera FPGA board. The designed NI occupies less than 1% of area utilization of Cyclone II FPGA. In addition, the data transmission delay from core to router and from router to core path is equal to $3 + 4(N - 1)$ clock cycles / packet ($N =$ Number of flits).

1 Introduction

Network on Chip (NoC) is a new approach for communication among different cores in a System on Chip (SoC). The basic idea of NoC is that cores are connected through a packet switched communication that is similar to the way computers are connected to the Internet. NoC is becoming an attractive option for solving SoC problems. It is a scalable architectural platform with huge potential to handle growing complexity and can provide easy re-configurability. NoC transmits packets instead of messages. Dedicated address lines like in bus-based communication systems are no longer necessary since the destination address of a packet is embedded in the packet. Data transfer of NoC can be conducted in parallel if the network provides more than one data transfer channel between a sender and a receiver. Thus, unlike bus-based communication SoC, NoC presents theoretical infinite scalability, facile IP core reusing, and higher level of parallelism [1].

Network on Chip has been focused more on issues related to router design, communication infrastructure, low power and fault tolerance. Comparatively less focus has been put on the design of a Network Interface.

It is desirable that the NI architecture is small scale and low power, that is it doesn't affect the performance and hardware complexity of NoC. Routing methods are used to determine the route followed by the message in the network on chip. Communication performance of a NoC greatly depends on the implemented routing methods. The routing methods are generally classified into distributed and source routing. In distributed routing the routing functions are implemented in each router of the network. The packet header is very compact. It carries the destination address and some control bits. Each router contains information about the neighbor routers. When the packet arrives at the input port of the router, the route path is selected either by looking up the routing tables or executing the routing function in HW. OASIS uses a so called look-ahead routing technique. On the other hand, in source routing the information about the route from source to destination is embedded in the packet header at the source node. The source makes all routing decisions before the packet injected into the network. The routing tables are placed inside the NI. The tables are filled with routing information. The sender NI selects route path from the tables and places this information in packet header.

HIRN and OASIS/OASIS-SPL NoC that are proposed in our laboratory have no Network Interface. So we cannot evaluate them with real benchmarks such as large programs. Their current evaluation is not accurate. In order to evaluate our NoCs, we need the NI. Thus, we decided to design a NI for distributed routing.

In this thesis, Section 2 describes the architecture of the designed NI for distributed routing in detail. Evaluation results are provided in Section 3. Finally, we conclude this thesis with future works in Section 4.

2 Distributed Routing NI Architecture

In this section, details of the architecture and design decisions for distributed routing NI are discussed.
2.1 Design Decisions for Distributed Routing NI

2.1.1 Packet Size
In order to decide the maximum buffer size, the maximum size of the packet should be fixed. So we assume that the maximum size of the packet will be 512-bits i.e., 16 32-bit flits. In distributed routing, a packet can have the minimum of 1 flit and the maximum of 16 flits.

2.1.2 Buffer Size
The NI has two different buffers for distributed routing. The sizes of them are the same and the maximum of 512-bits.

2.1.3 Communication Protocol
We used Ready to Receive based scheme as a communication protocol between core and NI and between NI and router. In that scheme, two 1-bit signals which is called RTR (Ready to Receive) and WR (Write) are used for handshaking signals.

2.1.4 Packet Buffering
Packet buffering is a process of temporarily storing packets in intermediate buffers while the packets are transferred from source core to destination one. The NI has two different buffers to store packets during their transmission from source to destination. The first one is used to store packets from the core. The second one is used to store packets from the router. In the data transmission from core to router, the NI receives all the packet from core into its buffer and it does not receive any further packet until it sends the whole received packet to the router, that is, single packet buffering. After transferring the whole packet to the router, it restarts receiving the next packet from the core. Similarly in the data transfer from router to core, the NI receives the whole packet from the router and it will not receive the next packet from the router until it sends all the received packet to the core. Whenever it transfers the whole received packet to the core, it again starts receiving the next packet from the router.

2.1.5 Packet Format
The maximum size of a packet is fixed to 512-bits. A packet can be divided into three types of flits such as HEADER, BODY and END of the packet. The packet header contains the first 32-bits of the packet. The remaining packets are payload. The end 32-bits of payload is a end of the packet. The payload between header and end of the packet is a body of the packet. The packet format is shown in Figure 1.

![Figure 1: Packet Format](image)

Packet Header Format
The size of the packet header is 32-bits. As the maximum size of NoC is 8x8, minimum 6-bits are required to represents the node address in the network. In the packet header, the first 6-bits represent the “Destination address” of the core in the network. The next 6-bits represent the “Packet size”. The packet size helps in tracking the arrival of the whole packet. The next 4-bits carry the “Packet sequence number”. The packet sequence number is used to rearrange the packet in correct order at the destination core. The next 8-bits are “Unused” for future use. The remaining 8-bits are “Payload”. The packet header format is shown in Figure 2.

![Figure 2: Packet Header Format](image)

Body and End of the Packet
The size of both body and end of the packet is 32-bits. Both include only “Payload”. Both body and end of the packet format are shown in Figure 3.

![Figure 3: Body and End of the Packet Format](image)

2.1.6 Flit Level Decision
After receiving a packet from the core, the NI converts the packet into flits. A packet can have minimum 1
flit and maximum 16 flits. The size of a flit is kept fixed and is equal to 34-bits. First 2-bits of each flit indicate “Flit Type”. A flit can be of three types and those are HEAD, BODY and END flit type. The table of flit types and each flit format are shown in Table 1 and Figure 4-6.

<table>
<thead>
<tr>
<th>Flit Type</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Flit with full Payload</td>
<td>00</td>
</tr>
<tr>
<td>Head Flit</td>
<td>01</td>
</tr>
<tr>
<td>Body Flit</td>
<td>10</td>
</tr>
<tr>
<td>End Flit</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 1: Flit Type and Codes

Figure 4: Head Flit Format

32 bits

Payload

Figure 5: Body Flit Format

32 bits

Payload

2.2 Network Interface Design for Distributed Routing

The designed NI for distributed routing is consisted of 6 internal blocks such as C2R-Buffer, Flitizer, C2R-Controller, R2C-Buffer, Deflitzer and R2C-Controller. The functionality of each block and control signals are discussed in the following subsections. The detailed internal structure of the NI is shown in Figure 7.

2.2.1 Blocks on Core to Router Path

The NI has different blocks and control signals as shown in Figure 7. In order to keep the core simple and not to be overloaded, flitization and deflitzation processes are done in the NI. The detailed functionality of each component is discussed in the following subsections.

C2R-Buffer

C2R-Buffer is a FIFO (First-In-First-Out) buffer which is connected to the input port of the NI from the core side. It has 16 locations to store a packet and the size of each location is 32-bits. A chunk of the packet is defined as a part of the packet and the size of a chunk is equal to 32-bits. Whenever it receives the “Write-Enable” signal from the C2R-Controller, it stores a packet coming from the core at the specified location. The location is specified by the “Add-Write” signal. It sends the stored chunk of the packet to flitizer block whenever it receives the “Read-Enable” signal from C2R-Controller. It sends the chunk from the address location which is specified by the “Add-Read” signal. This buffer can store the chunks of the packet coming from the core and send the stored packet to flitizer block simultaneously.

Flitizer

Flitization process starts when it receives the “Flitizer-Enable” signal from C2R-Controller. It reads the 32-bits of a packet i.e. a chunk from the C2R-Buffer. If the “Flit type” signal value which is received from C2R-Controller is either “00” or “01”, then the chunk of the...
packet which is read from the C2R-Buffer is assumed to be a packet header. If the “Flit Type” value is “00”, it means the packet contains only 1 flit. In that case, no body and end flits are present in the packet. If the “Flit Type” value is “01”, it means the packet contains more than 1 flit. In flitization process, it adds 2-bits flit type in the “Flit Type” field and 6-bits source address in the packet header i.e. from bit numbers 18 to 23, creates a 34-bits head flit. The format of the head flit was shown in Figure 4. When it receives the “Flit Type” signal as “10” or “11”, it assumes that the incoming packet from the C2R-Buffer is a body or end of the packet respectively. In these cases, it just adds the “Flit Type” to the flit at the field which is specified for flit type i.e. from bits 0 to 1, creates a 34-bits body or end flit. Both body and end flits format were shown in Figure 5 and Figure 6 respectively. After flitization process, the flits are transferred to a router.

C2R-Controller

C2R-Controller is the main block of the NI that generates all control signals for proper flow of packets from core to router. Internally C2R-Controller has different subcomponents and each is performing their specific tasks. In C2R-Controller, different control signals apart from data are entering and leaving.

2.2.2 Blocks on Router to Core Path

R2C-Buffer

R2C-Buffer is a FIFO buffer connected to the output port of the NI toward the core side. It has 16 locations to store the packet and the size of each location is 32-bits i.e. the size of a flit after deflittization. Whenever it receives the “Write Enable” signal high from R2C-Controller, it stores a flit coming from deflitter at a specified address location. The address location is specified by the “Add-Write” signal from R2C-Controller. Whenever it receives the “Read-Enable” signal, it sends the stored flit from the specified address location of R2C-Buffer to the core. The address location from where the flit to be read is specified by the “Add-Read” signal. R2C-Buffer can store the flits coming from deflitter and send the stored flits to the core at the same time.

Deflitterizer

This block performs the deflittization process. Deflittization is a process of reconverting flits into a packet. The deflittization process starts whenever Deflitterizer receives the “Deflitterizer-Enable” signal high from R2C-Controller and then it reads a 34-bits flit from a router port. It first checks the “Flit Type” bits. If it is “00” or “01”, Deflitterizer simply removes the “Flit Type” and “Destination Address” bits from the flit and shifts the “Source Address” bits to the “Destination Address” field and creates a 32-bits packet header. The created packet header exactly matches to the one that was created at the source, only “Destination Address” bits are replaced by “Source Address” bits and rest of the header bits remains the same. As soon as the deflittization process is completed, the created packet header will be sent to R2C-Buffer. When a body and end flit arrives, it just removes the “Flit Type” bits from the body and end flit and remaining 32-bits of payload are transferred to R2C-Buffer.

R2C-Controller

This block is responsible for controlling the communications from router to core. It has different subcomponents such as counters and FSM.

3 Evaluation Results

In this section, results of simulations and NI implementation on a FPGA board are provided.

3.1 Functional and Correctness Evaluation

The NI is designed in Verilog HDL, analyzed and synthesized with the Altera Quartus II, and simulated with the ModelSim-Altera. RTL and Gate Level simulation are done on the NI core to router and router to core path. Test bench files written in Verilog HDL are used for simulations which generate 16 packets including a packet header, 14 bodies of the packet and a end of the packet. The wave forms of RTL simulations are shown in Figure 8-9.

![Figure 8: Wave Form of Core to Router RTL Simulation](image-url)
chip RAM and 6 PIOs through Avalon bus. It is shown in Figure 10. The system is created with the SOPC Builder which is a part of the Quartus II. The FPGA board is DE2 one. The target device is EP2C35F672C6 of Cyclone II. On board 50 MHz clock frequency is used to run the system. Proper packet transmission between Nios II core and dummy memory through the NI and dummy router is verified. The RTL view of the core to core architecture is shown in Figure 11. From the left, dummy memory, NI for the memory, memory-side dummy router, Nios II-side dummy router, NI for Nios II core and Nios II system blocks are lined up. In order to generate packets and control signals from the Nios II system, a C program written with Nios II IDE is used. Similarly, the dummy memory sends packets to the Nios II core. The packets which the memory receives from the core and sends to the core can be seen on 7 segment displays on the DE2 board. The packets which the Nios II core receives from the memory and sends to the memory can be seen on the Nios II IDE console. One of the implementation results is shown in Figure 12.

![Figure 10: Nios II System Architecture](image)

### 3.2 Data Transmission Delay

#### 3.2.1 Core to Router Path Delay

The timing diagram for the communication from core to router was shown in Figure 8. 1 clock cycle is needed to receive a packet into C2R-Buffer from the NI input port and 2 clock cycles are required to read the packet from C2R-Buffer and flitize it. It sums up to 3 clock cycles to transfer the packet from the NI input port to output port. The time to transfer a complete packet from the NI input port to output port is:

\[
\text{Packet Delay} = 3 + 4(N - 1) \text{ clock cycles/packet}
\]

#### 3.2.2 Router to Core Path Delay

The timing diagram for the communication from router to core was shown in Figure 9. 2 clock cycles are needed to receive a flit into deflitzer from the NI input port and deflitize it and 1 clock cycle is required to read the deflitzed packet from R2C-Buffer and send it to the NI output port. It sums up to 3 clock cycles to transfer the flit from the NI input port to output port. The total time to send all flit from the NI input port to output port is:

\[
\text{Packet Delay} = 3 + 4(N - 1) \text{ clock cycles/packet}
\]

<table>
<thead>
<tr>
<th>Table 2: Hardware Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI Core to Core</td>
</tr>
<tr>
<td>Area (Total logic elements)</td>
</tr>
<tr>
<td>Speed (max)</td>
</tr>
<tr>
<td>Dynamic Power Dissipation</td>
</tr>
</tbody>
</table>
4 Conclusion and Future Work

We proposed an architecture and a design of a Core Network Interface for distributed routing. The NI is integrated with Nios II system, dummy router and dummy memory and implemented on a DE2 board. Evaluation results show that the area occupation of the designed NI is less than 1% of Cyclone II FPGA. The area utilization of the NI is small enough to not affect that of the NoC. The delay of packet transfer through the NI is $3 + 4(N - 1)$ clock cycles/packet ($N =$ Number of flits). Our HIRN and OASIS NoC do not have a NI. They cannot be evaluated with large programs. Therefore in the future, the NI will be integrated with our OASIS NoC and the NoC will be evaluated with real benchmarks. A NI for source routing will also be designed and prototyped. Then we will compare the source routing NI and the distributed routing one.

References


