Simplifying Microblaze to Hermes NoC Communication through Generic Wrapper

Andres Benavides A.¹, Byron Buitrago P.², Johnny Aguirre M.¹
¹ Electronic Engineering Department, University of Antioquia, Colombia
² Systems Engineering Department, University of Antioquia, Colombia

Abstract—In this paper an easy microprocessor to NoC connection strategy, based in a hardware wrapper design is proposed. The implemented wrapper simplifies the connection between a network on chip infrastructure and several MicroBlaze softcore processors. Proposed strategy improves the design process of a parallel computing environment. Wrapper development process, synthesis results and functionality test are showed and analyzed.

Key words: FPGA, Multicore, Hermes NoC, MicroBlaze, FSL bus, Embedded processors.

1. INTRODUCTION

Nowadays, computer applications need more than one processor to resolve complex tasks in short time. This particular fact has generated a new tendency in the design of high performance electronic systems. In an effort to improve the performance of a single processor scheme, multiprocessor architectures have been proposed. A multiprocessor (or multi-core) system takes advantage of the billion transistor era to achieve high performance by running multiple tasks simultaneously, on independent processors, decreasing applications execution time. However, parallel processing faces a lot of troubles, among which may be mentioned: Shared memory access and communication infrastructure. In a multi-core system, efficient communication among CPUs is a critical item to performance measurement. In reduced multi-core systems, a common bus is enough to connect the components. However with more than 8 cores a bus is not scalable because bus electrical load increases while its speed is reduced, and the bandwidth demand is not satisfied [1].

A scalable and efficient solution to connect on-chip components is a packet-switched on-chip network (NoC) [2]. Network-on-Chip (NoC) brings the techniques developed for macro-scale, multi-hop networks into a chip. Hermes [3], AET [4], Xpipes [5], are examples of NoC’s implementation. By means of NoC, systems communications improve by modularity support, cores reuse, and scalability increase. Those features enable a higher level of abstraction in multicore’s architectural modeling and allow heterogeneous systems building.

Another big problem in multicore architectures is related to quick prototyping capability. Traditionally, it has been only possible put under test the system once the silicon is available. In last years, softcore implementation on FPGA has emerged as a solution to rapid prototyping, due to their reduced cost, flexibility, platform independence and greater immunity to obsolescence [6]. A soft-core processor is a hardware description language (HDL) model of a specific processor (CPU) which can be customized for specific application requirements and synthesized for an ASIC or FPGA target. Examples of softcore are OpenRisc 1200 [7], LEON [8] and MicroBlaze [9]. Several architectures based in softcores can be found on internet sites as OpenCores [10] or Xilinx [11]. However typically softcore designs are limited to single processor or reduce multi-processors architectures connected by shared bus structure.

In this paper a strategy based in a hardware wrapper to simplify the connection between the Hermes NoC and MicroBlaze processor in order to facilitate the multicore architecture prototyping and design is proposed.

This paper starts with a background section in order to understand Hermes network on chip and MicroBlaze architecture. Then, wrapper design and internal architecture are explained. Wrapper implementation results, functionality test, conclusions and future work are showed and discussed at the end of this paper.

2. PRELIMINARIES AND BACKGROUND INFORMATION

2.1. NoC INFRASTRUCTURE

We have employed as communication infrastructure the HERMES NoC, developed by Moraes et al. [3]. The NoC (Figure 1) is formed by IP blocks and routers which are connected on a mesh topology. In Moraes’ NoC each IP block represents a computational element; in our case an IP block means MicroBlaze (MB) CPU. A unique address is
associated to each router on the net. The IP blocks have the same router’s address to which they are connected.

Figure 1. Router and a 3×3 HERMES NoC

All IP blocks can communicate with each other by sending packets on a rate of 500Mbps by each router. A valid packet is formed by a set of flits (1flit=8bits) according the formats illustrate in the Figure 2. Hermes NoC uses the wormhole flow-control for packet transference and a bi-dimensional routing algorithm.

Any IP block can be plugged into the network once it is equipped with the proper interface (wrapper). The wrapper adapts the IP block to router connection signals. The sections 3 explain how MicroBlaze processors were connected into the Hermes NoC.

2.2. MICROBLAZE SOFTCORE

The MicroBlaze core (Figure 5) is organized as Harvard architecture with separate bus interface units for data and instruction accesses. Each bus interface unit is further split into a Local Memory Bus (LMB) and IBM’s On-chip Peripheral Bus (OPB). Further, MicroBlaze core provides 8 input and 8 output interfaces to Fast Simplex Link (FSL) buses. The FSL buses are unidirectional, non-arbitrated, dedicated and synchronized communication channels. The FSL bus transmits data directly from the MicroBlaze core to other peripherals or processors buses in master-slave scheme without using a shared bus. MicroBlaze contains several instructions to read from the input FSLs and write to the output FSLs. Each read and write operations consume two FPGA’s clock cycles.

Figure 2. HERMES NoC packet’s formats

The router transfers packets among IP blocks by means of 4 bidirectional ports (North, South, East and West), and a local port (to connect an IP block). The Figure 3 shows physical connection between two consecutive ports. Each port has an output and input gates. Each one of them has a FIFO memory buffer to temporal information storage. In the output gate, the tx indicates that there is a flit in the data_out bus, the signal is cancelled when the ack_tx signal is received. In the input gate the rx signal indicates that there is a flit in the data_in bus, when it is taken and sent to other router (or to the IP block), the ack_rx signal is generated.

A FIFO memory buffer is used as interface between the MicroBlaze and the other peripheral. Buffer allows using different clock sources for the FSL_M_Clk and FSL_S_Clk signals. In a master to slave writing process the master checks the FSL_M_Full signal to known the FIFO state. When the FIFO is available ($FSL_M_Full=0$) the master puts the data on the FSL_M_data bus and activates the FSL_M_Write signal. On the slave side the signal FSL_S_Exists indicates it that a data should be read. The peripheral takes the data by means the FSL_S_Data bus and activates the FSL_S_Read signal as acknowledge.
The optional FSL_M_Control and FSL_S_Control signals can be used to coordinate the communication. In a slave to master writing process the roles between the MicroBlaze and the peripheral are inverted. Also, the FIFO depth can be increased to raise the performance communication.

3. FSL WRAPPER ARCHITECTURE

The designed wrapper allows to plug each processor with the NoC infrastructure [12]. The wrapper takes the signals from FSL and translates them to router’s properly signals. In this way, wrapper functionality can be interpreted how FSL to NoC and NoC to FSL communications abstraction layer. In the first case, flits arrive from FSL to be routed by the NoC. In the second case, flits arrive from NoC to be sent to the MicroBlaze.

The wrapper (Figure 5) is composed by a coprocessor, a register manager and Tx/Rx Module.
- Coprocessor: It takes the FSL signals described in the section 2, and generates asynchronous signals to write or read in the register manager.
- Register Manager: In the FSL to NoC communication process, it receives the frames sent by the coprocessor, decodes and saves them in the W-FSL register. In the NoC to FSL communication process, it takes the data from W-NoC register, encodes and sends them to coprocessor.
- Tx/Rx Module: It fixes the connection with the router’s local port. Input and output gates, described at section 2, are its main components. This module takes flits from W-FSL register and puts them on the output gate. In the other way, it takes the data from the input gate and writes them on the W-NoC register in the manager register module.

Packets’ integrity is guaranteed by NoC infrastructure. However it was necessary to implement a local protocol for wrapper in order to ensure the correct communication between MicroBlaze and router. In this local protocol, each packet is transmitted in three frames according the format shown in figure 6. The control field indicates which frame is being sent. The decode module in the register manager interpreters those frames. When one field is lost, the whole frame is rebroadcast.

<table>
<thead>
<tr>
<th>Control</th>
<th>1st Flit</th>
<th>2nd Flit</th>
<th>3rd Flit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Frame</td>
<td>0x01</td>
<td>Target Address</td>
<td>Payload Size</td>
</tr>
<tr>
<td>2nd Frame</td>
<td>0x02</td>
<td>Address [15:8]</td>
<td>Address [7:0]</td>
</tr>
<tr>
<td>3rd Frame</td>
<td>0x03</td>
<td>Data [15:8]</td>
<td>Data [7:0]</td>
</tr>
</tbody>
</table>

Figure 6. Wrapper protocol

The wrapper also allows to see NoC like an extension of the FSL bus. Therefore, NoC’s writing and reading tasks can be managed using high level functions available for the FSL since a programming language. The example 1 shows the C function to send a packet from MicroBlaze to NoC.

```c
void writeNoc(char tg, char sz, char cm, char adH, char adL, char daH, char daL) {
    auxTx = 0x01<<24 | tg<<16 | sz<<8 | cm;
    putfsl(auxTx, FSL_MASTER);
    auxTx = 0x02<<24 | adH<<16 | adL<<8 |0x02;
    putfsl(auxTx, FSL_MASTER);
    auxTx = 0x03<<24 | daH<<16 | daL<<8 |0x03;
    putfsl(auxTx, FSL_MASTER);
}
```

Example 1. FSL to NoC writing process

The example 2 shows the function to read a packet from NoC.

```c
void readNoc (char * frame) {
    getfsl(auxRx,FSL_SLAVE);
    frame->tg=(char)(auxRx)>>16); frame->sz=(char)(auxRx)>>8);
    frame->cm=(char)auxRx;
    getfsl(auxRx,FSL_SLAVE);
    frame->adH=(char)(auxRx)>>16);
    frame->adL=(char)(auxRx)>>8);
    getfsl(auxRx,FSL_SLAVE);
    frame->daH=(char)(auxRx)>>16);
    frame->daL=(char)(auxRx)>>8);
}
```

Example 2. NoC to FSL reading process

4. WRAPPER TEST

To study the wrapper functionality, it was generated an architecture with three MicroBlaze CPUs connected through the designed wrapper to Hermes NoC. A serial port
was included for debug purposes. The whole system is illustrated in the figure 7.

EDK Xilinx tool was used to generate each MicroBlaze core. The SDK Xilinx’s tool was employed to generate the software. Individual programs were written by each processor. Stop y Start commands ensures the coordination of communication processes. Finally ISE tool was used to make the connection among MicroBlaze processors, designed wrappers and NoC infrastructure. The whole system was synthesized on the Virtex 4 FX20 FPGA. The synthesis report is illustrated on the table 1.

<table>
<thead>
<tr>
<th>MODULE</th>
<th>Power(W)</th>
<th>LUTs</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dynamic</td>
<td>Quiescent</td>
<td></td>
</tr>
<tr>
<td>MB</td>
<td>0.183</td>
<td>0.256</td>
<td>2610</td>
</tr>
<tr>
<td>WR</td>
<td>0.013</td>
<td>0.219</td>
<td>71</td>
</tr>
<tr>
<td>MB+WR</td>
<td>0.189</td>
<td>0.256</td>
<td>2891</td>
</tr>
<tr>
<td>SERIAL</td>
<td>0.020</td>
<td>0.219</td>
<td>321</td>
</tr>
<tr>
<td>NOC</td>
<td>0.038</td>
<td>0.220</td>
<td>1912</td>
</tr>
<tr>
<td>WHOLE</td>
<td>0.651</td>
<td>0.324</td>
<td>7490</td>
</tr>
</tbody>
</table>

Table 1. Synthesis report.

The figure 8 shows application running results, data was taken from a serial port sniffer. It shows a token passing example, where each MicroBlaze takes a common variable, increases it and passes to the next MicroBlaze and a serial port.

The string “Print from MicroBlaze 01, i=1”, is a message sent by processor 01 in the net. “Print from MicroBlaze 10, i=2”, is a message sent by 10 in the net and “Print from MicroBlaze 11, i=3”, is a message sent by processor 11 in the net. The serial interface has 00 coordinate in the net.

5. CONCLUSIONS AND FUTURE IDEAS

In this paper a hardware wrapper to simplify the multicore architecture design and prototyping, using the Hermes NoC and the MicroBlaze softcore was introduced. The wrapper test showed the low cell units occupied the functionality and the good performance of the proposed wrapper.

FSL employment to connect the MicroBlaze with NoC allows to give to the developer a higher abstraction level, through simple software language functions calls hidden low level details.

On the other hand, the network structure ensures the scalability and enables a multicore architecture can be built in a modular way. This scheme reduces design time because it allows a considerable components reusing strategy.

The reconfigurable hardware environment allows architectural customization. This feature enables heterogeneous design, particularly in Virtex FPGA, PowerPC CPU can be employed in our multicore system using Hermes NoC due with developed wrapper, without the necessity of any change.

Future ideas cover applications design using multicore platform. Those applications involve signal processing, simultaneous multisensory acquisitions, scientific computations, server clusters, hardware accelerators among others.

6. ACKNOWLEDGMENTS

The authors would thank to Microelectronics and Control group of University of Antioquia, who provided the software and hardware tools during project realization.

REFERENCES


