A Parallel Low Latency Bus on Chip for Packet Processing MPSoC

Pei-Jun Ma, Pei-Yan Liu*, Kang Li, You-Yang Zou, Ai-Nv An, Yan-Long Wang, Yue Hao
School of microelectronics of Xidian University, Xi’an, 710071, China
* Email: mantousx@163.com

Abstract

In this paper an bus architecture combining crossbar switching with split transaction feature is presented for multiprocessor system on chip (MPSoC) in the packet processing application. The high throughput is achieved with crossbar bus topology and low bus latency is finished by split transaction buses which separate address bus from data one. Experimental results show that performance of the proposed architecture is improved up to 2.3 times than the one of the AHB buses and reduce communication latency about 45% than the later. Moreover, the bus arbiter implementation has reasonable area and timing cost which make it suitable for high performance IP-packet or base-band processing.

1. Introduction

Advances in technology now allow more and more IP modules integrated on a single chip, the design of high performance global communication architectures becomes the key to successful SoC designs. There are mainly two kinds of bus topologies: shared bus topology and crossbar bus topology. Several companies have developed their own on-chip bus architectures, such as CoreConnect, AMBA, and Willsbone[1]. However, as only one module can access the shared-bus at any time, the bus bandwidth is limited when the amount of processor modules increase. Crossbar switching bus allows more than one bus operation proceeded simultaneously[2]. A split-transaction bus can decouple requests from responses, so the shorter latency responses is finished[3].This paper presents a bus architecture named XDPB which achieve a high throughput by adopting crossbar bus topology in data bus and separate address and data lines. The distributed arbitration strategy is adopted in the architecture. The result shows the notable performance improvement.

2. Architecture of XDPB

XDPB used in multithreaded MPSoC environment is shown in figure 1. The system consists of control plane and data plane. Control plane is managed by a GPP which take charge of system configuration and downloading the program of Processing Engine(PE). The core of the data path is consists of multithreaded processors pool (PE0-PE5). They share several kinds of memory units providing different memory access. For instance, SRAM memory is used for accelerating look-up table access, SDRAM for providing data access and coprocessor for accelerating specific operation with hardware. The FIFO bus interface is responsible for receiving and transmitting the IP packets.

The buses are divided into independent command buses and data buses based on the split-transaction bus, at the same time we implement the crossbar interconnect in data buses. The remainder of this paper describes the architecture of XDPB in detail.

2.1 Implementation of split-transaction bus

Lower bus latency and higher throughput can be obtained by using a split-transaction bus that decouples requests from responses based on independent address and data lines. And this is the architecture strategy employed in XDPB bus topology.

The structure of split-transaction bus is adopted here, as shown in Figure 2b. A bus transaction is divided into two independent stages: Master request stage and Slave response stage. On the other hand, the bus includes data buses and command buses. The master
can raise a new access request without waiting for the last slave device’s response. The requests from the masters can be continuous so as to make bus latency much lower.

The command_queue is necessary for the split structure because the access requests from the different masters must be saved at slave devices. At the same time, a signal is needed to notify master that the transaction has been finished. The split structure for XDPB is shown in Figure 3, where the independent command and data bus topology is implemented respectively.

2.2 Crossbar Bus Topology
Crossbar bus allows multiple simultaneous bus accesses so that the bandwidth can be improved a lot. As shown in Figure 3b, for instance, when PE0 is accessing SDRAM, PE1 can access SRAM and PE2 can access FBI at the same time.

Supposing the working frequency of the bus is 232MHz and the bit width of data bus is 32, if parallel crossbar interconnect is adopted, the bandwidth of the system will be up to 22272Mbps (232 MHz *32bits*3), because through this kind of bus three different data blocks can be read out from SDRAM, SRAM and FBI at the same time. Whereas if shared bus interconnect was adopted, the bandwidth of the system is just 7424Mbps (232 MHz *32 bits), one third of that with crossbar architecture, because there is only one group of data can be read in the bus from SDRAM or SRAM or FBI at any time. Obviously this parallel join crossbar bus topology has a better performance than the shared bus topology based on our system.

2.3 Distributed Arbitration and Bus Optimization
On one hand, the shared bus architecture has been implemented in the command bus as is shown in Figure 3a. And at any given time only one processor can get the use right of the bus. So an arbiter is needed to give the bus use permission to a special PE. On the other hand, the crossbar bus architecture that is adopted by the data bus shown in Figure 3b allows three of processors (PEn, n=0~5) to access different memory units (SRAM, SDRAM and FBI) simultaneously. There is probability that several processors want to access the same memory unit, which will cause bus contention problem. So some data bus arbiters are needed, which can ensure that various processors’ accesses are right and in order. Basically, there are four arbiters connected to the bus of the system, which are shown in Figure 3. This is a kind of distributed arbitration strategy. There is an arbiter named arb_c in the command bus, and meanwhile we add an arbiter (arb1 or arb2 or arb3) in every memory unit. Commonly used arbitration methods include priority-based arbitration and time division multiple access (TDMA). In our system, in order to satisfy the requirements about “fairness” and “the priority” we set mixed priority which is base on the priority and Round-Robin algorithm for the command bus arbiter arb_c, and set Round-Robin arbitration rules for the memory units’ arbiters (arb1,arb2,arb3 ).

As there are a lot of data blocks to be transmitted in the SDRAM element, SDRAM data bus is required to have a very high data throughput. While, single data transmission is the main data transmission style in the SRAM and FBI element, so the throughput requirements about their data bus is not needed to be as high as SDRAM element. Thereby, we can optimize the bus by combining the data bus of SRAM with the one of FBI, as shown in Figure 4b. By doing so, we can reduce the number of data bus and solve the inherent problems of crossbar bus architecture that there are too many connection wires and large load. However, because of the shared bus between SRAM and FBI, there must be an arbiter within the combined buses to ensure that there is only one type of bus access at any time. To simplify the design of the named arb_d arbiter, we set time division multiple access (TDMA) rules for arb_d.Use right of the bus is changed periodically with the odd cycle and even cycle.

3. Verification And Analysis
XDPB bus architecture is used in XDNP system with MPSOC architecture. There are 6 multi-thread processors and a general purpose processor being integrated in this system. Through the shared-memory
controller they access the memory, and the whole frame is given in Figure 1. The system is verified at the RTL level in VCS environment and synthesized with Design Compiler. We do statistic analysis to the systems with XDPB bus architecture and AHB bus architecture separately, and get the result of access time to memory units and data throughput, as is shown in Figure 5 and Figure 6.

![Figure 5. Buses communication latency distribution](image)

(a) Latency distribution of XDPB

(b) Latency distribution of AHB

Figure 5. Buses communication latency distribution

The access time probability distribution of processor to memory unit with different bus architecture is shown in Figure 5. From the figure we can learn that the time of access memory is concentrated in the range from 15 to 30 clock cycles when XDPB architecture is adopted. While the access time range is 30 to 75 clock cycles when AHB architecture is implemented. Then we can calculate the average communication latency of each memory based on this probability distribution. As is shown in Table 1, the average access time with XDPB architecture is about 55% of that with AHB architecture.

<table>
<thead>
<tr>
<th>Bus Architecture</th>
<th>FBI (cycle)</th>
<th>SRAM (cycle)</th>
<th>SDRAM (cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDPB</td>
<td>21</td>
<td>20</td>
<td>44</td>
</tr>
<tr>
<td>AHB</td>
<td>40</td>
<td>38</td>
<td>71</td>
</tr>
</tbody>
</table>

Table 1. Average communication latency

Figure 6 shows the throughput of each memory unit’s data bus with different bus architecture. Compared to the bus with AHB architecture, the bus with XDPB architecture of each external element has a higher data throughput. By adding all the three groups of throughput, we can get that the average throughput of the system is 5.8Gbps, and 2.5Gbps for AHB architecture. So we can conclude that the average bandwidth of XDPB is 2.3 times as high as that of AHB.

![Figure 6. Throughput of data buses about slave elements](image)

4. Conclusion

This paper gives a kind of bus architecture XDPB focusing on IP packet processing in the chip multi-processor (CMP) system, through which we can achieve high throughput and low bus latency. Compared to the test result of system that adopts AHB bus architecture, our system with crossbar bus architecture has a 2.3 times improvement in throughput and 45% decrease in communication latency. And the throughput of the system can achieve up to 5.8Gbps. We verified our system at the RTL level in VCS environment and mapped the design to TSMC 0.18 um technology through 'Synopsys Design Compiler'. The delay of the arbiter is 2.97 ns, and its area is 10,262. Therefore, the arbitration can be performed in one cycle for bus speed up to 300 MHz. As a parallel bus with wide bandwidth and low latency, our bus architecture can be used in network processor and base-band processor.

Acknowledgement
Support for this work was provided by The National Natural Science Foundation of China (60506020) and Natural Science Foundation of Shanxi Province of China (SJ08-ZT13).

References