Advanced Extensible Crossbar Protocol for Connecting Multi-Cores and Shared-Memory on-Chip

Hongyu Meng
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
95 Zhongguancun East Road,100190, Beijing, China
menghongyu2014@ia.ac.cn

Donglin Wang, Zijun Liu and Yang Guo
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road,100190, Beijing, China
{donglin.wang & zijun.liu & guoyang2014}@ia.ac.cn

Abstract—**With the development of the chip industry, the bandwidth of interconnect on-chip are becoming more and more important for the performance of chip system. However, heterogeneous Intelligent Property (IPs) including cores and memory banks are being integrated in one chip which makes the hardware design of interconnect difficult. In this paper, we present Advanced Extensible Crossbar (AEC) protocol used for describing and designing the bus on-chip. It can support most of features in Advanced eXtensible Interface (AXI) protocol. Our AEC-based crossbar has been implemented with connecting 8 processors (the area of each one is 13.2 mm^2) and 16 MB Static Random Access Memory (SRAM, the area is 1.87 mm2/MB) in TSMC 28nm HPC process and the results show that it can achieve high frequency of 800 MHz after place and route (P&R) while the area requirement is 0.065 mm2. Compared with AXI-based crossbar provided by Synopsys, our AEC-based crossbar gives 1.45X frequency increase and 8X area requirement saving.**

Keywords-interconnect; crossbar; multi-cores; shared-memory;

I. INTRODUCTION

With the development of very large-scale integrated (VLSI) circuits and ultra-deep sub-micron manufacturing technologies, the integration level of chip system is getting higher and higher. Full-featured System on Chip (SoC) appears as the traditional integrated circuit with single function is no longer apply to the needs of information technology. Since SoC was proposed in the 1990s, its efficiency of research and development has been improved by the intellectual property (IP) reuse technology significantly.

Bus which is also often referred to interconnect is an important part of SoC. In early processor design, a few of master devices or only one master device was usually integrated on bus and most of the bus were designed in broadcast architecture which allows only one master at a time to access slaves, as this kind of bus can be achieved in small area requirement and power consumption. Advanced High-performance Bus (AHB) in Advanced Microcontroller Bus Architecture (AMBA) protocol proposed by ARM [1] is a typical representative of this kind of bus and it is still used for many low-power embedded devices. However, as more and more processors are mounted on the bus, the shared bus using broadcast architecture is exposed to bandwidth bottleneck. Therefore, a high-bandwidth bus using crossbar architecture comes out and it is also named as crossbar. Nowadays, most of crossbar are compliant to either Advanced eXtensible Interface (AXI) protocol or Wishbone standard. AXI protocol is also proposed by ARM [2] while Wishbone is an open-source interconnect standard maintained by the community project OpenCores. Crossbar can allow multi-master devices access different slave devices at the same time which means it can achieve high bandwidth compared with shared bus. The IBM's Blue Gene/Q processor is used this crossbar-based interconnect for transferring data [4]. However, the bus on-chip is still being connected with more and more IPs which makes latency, area and power of crossbar-based interconnect grow exponentially. Thus Network on Chip (NoC) has emerged as a solution to the high radix crossbar due to its scalable topology. Because of the low complexity of NoC-based interconnect in hardware implementation, it can achieve high frequency and is widely used in many-core processors, such as the second-generation Intel Xeon phi product family [5]. Though the NoC-based interconnect can often achieve high frequency, it is difficult to get maximum theoretical bandwidth while the radix of interconnect is high and communication in nodes are frequent. Compared with the 2D-mesh topology of NoC-based interconnect, the crossbar interconnect can get the theoretical bandwidth easily due to its non-blocking connectivity [3].

In order to improve the bandwidth between cores and memory on-chip, we present Advanced Extensible Crossbar (ACE) protocol which can achieve high transaction efficiency with low complexity in design. The signals contained in each channel of AEC protocol are able to handle the access selection (decode) and access conflict (arbiter) while we keep the function of AEC protocol signals in agreement with the signals of cores and memory as far as possible. The two key features of our AEC protocol are: 1) it can achieve high bandwidth utilization as the header signals, body signals and tail signals coexist in each channel; 2) it is extensive as the ID signal widths of master and slave are equal. We implement two subsystems containing 8 cores and 16 MB shared memory on-chip. One use AXI-based crossbar provided by Synopsys and the other use AEC-based crossbar. After synthesis in Design Compiler and place and route (P&R) in IC Compiler, our AEC-

based gives 1.45X frequency increase and 8X area requirement saving.

Following the above discussion, the remainder of this paper is organized as follows. We start by introducing related work in the next Section II. In Section III, we describe our AEC protocol, and we present the implementation of the subsystems using AXI-based crossbar and AEC-based crossbar in section IV. Finally, section V is a conclusion.

II. RELATED WORK

[6] mainly focus on the analysis of crossbar topology. They firstly present some analysis of relationship among radices, data width and area. Then they present three die Floorplans of crossbar and three designs of data path multiplexors. They also analyze two ways of grouping the multiplexor: bit slicing and port slicing. At last, they use standard cell design flow to build a high-radix crossbar for analysis and simulation.

[3] bring a crossbar modeling tool, and they focus on how crossbar area, power, and performance vary across input/output node number, data width, wire parameters, and circuit implementation. They identify a point using the modeling tool which demonstrates higher throughput with lower area and power than previous published designs.

The above two works focus on modeling and analysis the relationship between crossbar performance and hardware design parameters. The results of these works are modeled and not implemented by EDA tools.

[7] also present a high-radices crossbar named Swizzle-Switch Network (SSN) which can achieve high throughput. This work focus on designing the arbiter and present a single-cycle least-recently-granted (LRG) priority arbitration technique and an additional 4-level message-based priority arbitration for quality of service (QoS).

The design in [8] use an AXI-based crossbar with 256 bits data width and 4 master ports in STMicroelectronics Bulk CMOS-28nm Low Power technology library. The result of preliminary logic synthesis using Synopsys Design Compiler is that their crossbar can meet a frequency of 1GHz while the area is less than 0.2 mm2. However, the result is just after synthesis and the AXI-based crossbar may not simplify timing closure after P&R.

[9] and [10] present an AXI-based crossbar which shows a frequency of 728 MHz after P&R using Cadence Encounter Digital Implementation System in 28 nm FD-SOI standard cell technology. They also present Mesh-of-Trees topology which can improve the frequency of crossbar interconnect than typical topology. However, the data width of the crossbar is 32 bits and it doesn't support outstanding and out-of-order transactions due to the limitation of CPU cores.

III. AEC PROTOCOL

AEC protocol is mainly used for connecting cores and shared-memory on-chip so some signal nets contained in AXI protocol may not be used in this scenario. Meanwhile, AEC protocol include the key features of AXI protocol and SRAM transaction in order to improve its universality.

A. AEC Protocol Introduction

AEC protocol is suitable for high-bandwidth and low-latency design as it can provide high-frequency operation. It supports unaligned data transfers by using byte strobes and separation of reading and writing transfers. Issuing multiple outstanding address and out-of-order transaction completion are also supported by AEC protocol (out-of-order transaction is not supported in write channel). AEC protocol defines three independent transaction channels: read address channel, read data channel and write channel. Read address channel transfers the read control information from the master to the slave while read data channel transfers the read-back data from the slave to the master. Write channel transfers both write control information and write data from the master to the slave.

B. AEC Protocol Signal Descriptions

TABLE I shows that each channel are composed of a pair of handshake signals and payload signals. Figure 1 shows more specific signals in each channel.

TABLE I. SIGNALS IN EACH CHANNEL

Signal	***Description***
Xvalid	Channel valid. This signal indicates that the payload in this channel is valid.
Xready	Channel ready. This signal indicates that the payload in this channel can be received by the destination.
Xpayload	Channel payload information.

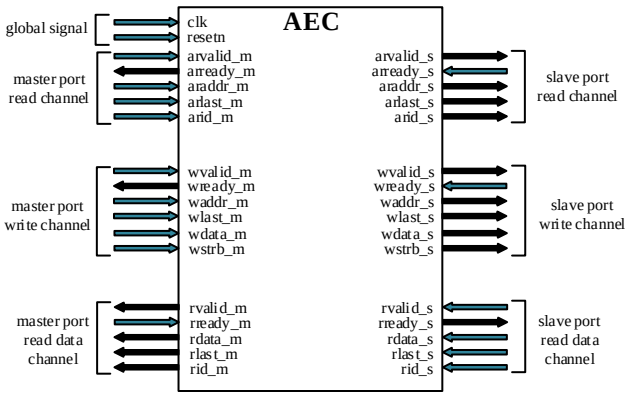

Figure 1. AEC protocol signals.

C. AEC key features

- It can achieve high bandwidth utilization without the limitation of specified number of beats in one transaction. In AEC protocol, transaction in each channel is based on packet switching, and the signal nets of header, body and tail in one packet coexist in each channel. Each beat of the packet is transferred and accepted separately by the **Xvalid** and **Xready** which means trade area for performance. We use **Xaddr** or **Xid** and internal signals in each channel as header and **Xlast** to indicate the last beat in one packet during one transaction.

- It is easy to be extended with connecting other AEC-based interconnect. In AEC protocol, we define the width of ID signals always same no matter in the master side or the slave side. The ID width of the master and the slave should be extended to *v* as in (1) which *N* indicates the number of master and *w* indicates the original ID width of the master devices. The additional ID signals of the master side should be numbered by designer, and these ID signals would be used as identifying the master device in read data channel in AEC protocol. Figure 2 shows an example that one AEC-based crossbar can be extended conveniently.

$$v = \text{ceil}\,(\log2\,(N)) + w. \quad (1)$$

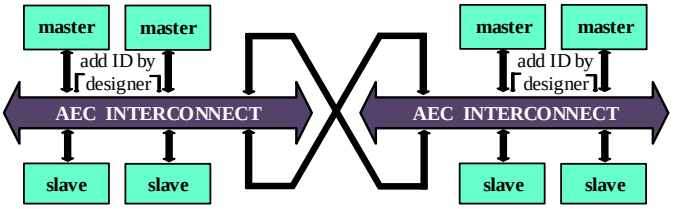

Figure 2. Extension between two AEC-based interconnects.

IV. Implementation and Results

Based on our existing APE core which stands for Algebraic Processing Engine [11], we make improvement at the interface of the core and use it as the master device. It can achieve 1.2 GHz using AXI 4.0 interface in 28 nm process with 128 bits data width while area requirement is 13.2 mm2. We use APE core which we have obtained the timing library file (**data base** file) and physical library files (in **Milkyway** folder) as macros in synthesis and back-end design.

The slave device is Static Random Access Memory (SRAM) provided by Synopsys in based TSMC 28 nm HPC process. With the help of Synopsys memory compiler, we obtain the largest single-port memory with 128 bits data width and 8192 depth as well as the **data base** file and **Milkyway** folder. However, we think the capacity (128 bits and 8192 depth mean 128KB) is still small in actual designs. So we use eight this SRAM to piece a 1 MB memory bank of which the area requirement is 1.87 mm2. Then we use one 1x2 AEC-based crossbar to connect two memory banks and use it as slave device.

We build two subsystems and each one use an 8x8 crossbars as the main interconnect to connect 8 APE cores and 16 memory banks which have been connected by 1x2 crossbars as above. The first use AXI-based crossbars provided by Synopsys and the second use AEC-based crossbars, and the AEC-based crossbars are all non-pipelined. We also complete two adapter modules between cores and AEC-based crossbar, and AEC-based crossbar and SRAM in the second subsystem as shown in Figure 3 (AXI2AEC and AEC2SRAM). We add the Asynchronous First In First Out (ASFIFO) in AXI2AEC module, and the crossbars and the memory banks are in same clock domain. In order to provide timing closure, we insert register slices at each master and each slave of the 8x8 crossbar. In addition, the Figure 3 just shows the logic from the master to the slave and from the slave to the master, we use the same logic. The implementation details are as follows.

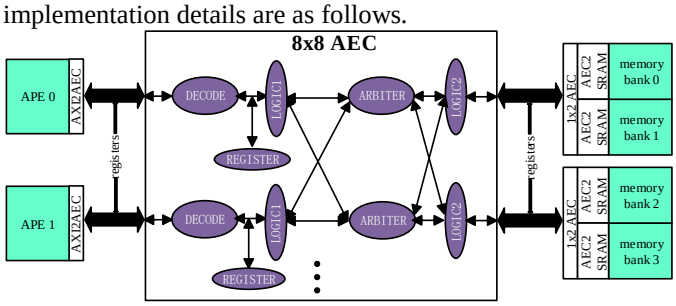

Figure 3. Subsystem architecture with AEC-based interconnect.

A. *Implementation of issuing outstanding transaction*

The write channel of AEC-based crossbar does not have the ID signal and the AEC protocol does not have write response channel. We also consider that the write data can be written in SRAM as long as the master grants in slave write channel arbiter. So in AXI2AEC module as shown in Figure 4, we enable the AXI write response channel signals according to write address channel and write data channel. As result, we don't need to add logic of issuing outstanding transaction in write transaction and the write response signals can be enabled orderly as we design.

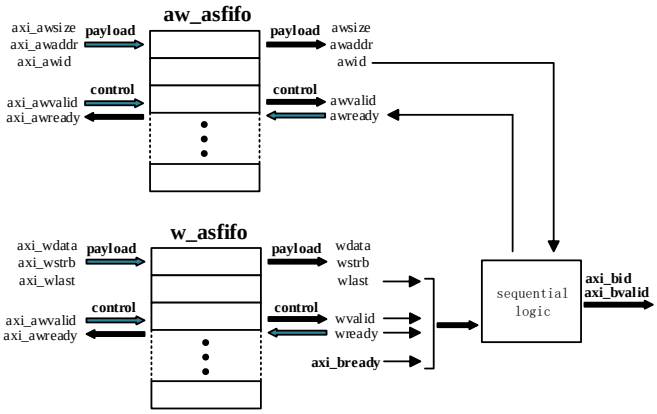

Figure 4. Implementation of issuing outstanding transaction in write channel.

In read address channel and read data channel, we add counter registers behind each master decode module. These registers are corresponding with each slave and each ID signals, and they are 2 bit-width which means that the AEC-based interconnect can support issuing 4 outstanding transactions at most when one master access the same slave in same ID. Figure 5 shows the details of implementation. When the value of ID and the slave are all same in read transaction, then the counter will add one which means one transaction. When the value of ID and the slave are all same in read data transaction, the counter will minus one. When the value of ID overflows, the **arready** of the master will be pulled down. The **ready** of the slave will be pulled down when the value of ID underflows. In addition, the **arready** will be pulled down when the master access different slaves in same ID.

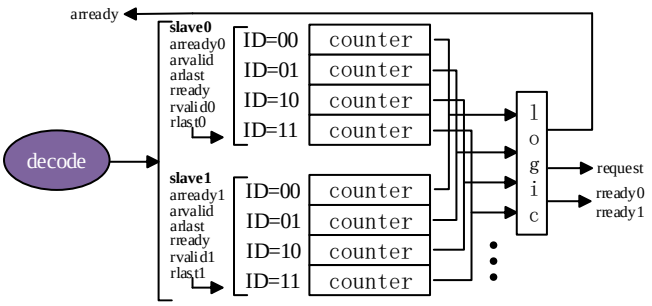

Figure 5. Implementation of issuing outstanding transaction in read address channel and read data channel.

B. Implementation of arbiter

We use round-robin arbiter in this AEC-based crossbar. In order to improve the frequency, we use one-hot code in encoding module of the arbiter which means we can use fewer logic to handle the grants than binary code. In this 8x8 crossbar design, we use 8 bits signal nets for grants and 8 bits stage registers.

C. Results

We use Design Compiler to synthesis this two subsystems and set the uncertainty of the clock to 0.2 ns. The results of our experiments and others' work are shown in the TABLE II. The AEC-based crossbar can get higher performance easily than AXI-based crossbar in power and area due to its concise protocol. As we insert register slices at the master and slave of the crossbar, it can get high frequency. From the Design Compiler timing reports, we know that the delay of decode module in AEC-based crossbar is 0.04 ns, combinational logic 1 is 0.04 ns, arbiter module is 0.16 ns, and combinational logic 2 is 0.2 ns. The output signals delay of register slices is less than 0.3 ns and the setup time of input signals requires less than 0.03 ns.

TABLE II. RESULTS OF DIFFERENT CROSSBARS

crossbar	*Power (mw)*	*Area (mm2)*	*Data width (bit)*	*Speed (GHz)*	*Node*
AXI-based in [8]	N/A	0.2	256	1	4
AXI-baesd in [9]	18.4	0.102	32	0.8	8x16
AXI-based of Synopsys	57.6	0.132	128	0.75	8x8
AEC-based	4.1	0.027	128	1	8x8

We also create placement of this two subsystems using IC Compiler in centralized crossbar architecture. Firstly, we make space for placing standard cells of the crossbar module. Then we place bank macros on the right and left of this space, and place APE macros at the top and below of this space and banks. After optimizing P&R, we check the timing reports. The maximum clock frequency of AXI-based crossbar reduces to 550 MHz while the AEC-based crossbar can also achieve frequency of 800 MHz. The area requirements increase very much due to the utilization are different in synthesis and placement. The area requirement of AXI-based crossbar is 0.52 mm2 while the AEC-based crossbar is 0.065 mm2.

V. CONCLUSIONS

In this work, we present a new bus protocol which can be used for designing interconnect between cores and shared memory on-chip. This protocol can support most of the features in AXI protocol while it contains fewer signal nets than AXI protocol. From the results of implementation of the AEC-based crossbar, we consider that it can get higher performance than AXI-based crossbar.

However, this protocol are not appropriate for supporting cache memory as it does not contain coherence signals. This protocol does not provide quality of service compared with AXI 4.0 either. Furthermore, in order to get high frequency, we simplify the design of issuing outstanding transaction module which would bring one more cycle delay when the master access different slaves in same ID.

REFERENCES

[1] AMBA T M. Specification (ahb)(rev 2.0)[J]. ARM Ltd, May, 1999, 1: 1.

[2] AXI A R M A, Specification-AXI A C E P. AXI4, and AXI4-Lite, ACE and ACE-Lite[R]. Technical report, 2011.

[3] Cakir C, Ho R, Lexau J, et al. Modeling and design of high-radix on-chip crossbar switches[C]//Proceedings of the 9th International Symposium on Networks-on-Chip. ACM, 2015: 20.

[4] Haring R, Ohmacht M, Fox T, et al. The ibm blue gene/q compute chip[J]. Ieee Micro, 2012, 32(2): 48-60.

[5] Sodani A, Gramunt R, Corbal J, et al. Knights landing: Second-generation intel xeon phi product[J]. Ieee micro, 2016, 36(2): 34-46.

[6] Passas G, Katevenis M, Pnevmatikatos D. Crossbar NoCs are scalable beyond 100 nodes[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(4): 573-585.

[7] Satpathy S, Sewell K, Manville T, et al. A 4.5 Tb/s 3.4 Tb/s/W 64× 64 switch fabric with self-updating least-recently-granted priority and quality-of-service arbitration in 45nm CMOS[C]//Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International. IEEE, 2012: 478-480.

[8] Azarkhish E, Rossi D, Loi I, et al. High performance AXI-4.0 based interconnect for extensible smart memory cubes[C]//Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium, 2015: 1317-1322.

[9] Sievers G, Daberkow J, Ax J, et al. Comparison of shared and private l1 data memories for an embedded mpsoc in 28nm fd-soi[C]//Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2015 IEEE 9th International Symposium on. IEEE, 2015: 175-181.

[10] Sievers G, Ax J, Kucza N, et al. Evaluation of interconnect fabrics for an embedded MPSoC in 28 nm FD-SOI[C]//Circuits and Systems (ISCAS), 2015 IEEE International Symposium on. IEEE, 2015: 1925-1928.

[11] Wang D, Du X, Yin L, et al. MaPU: A novel mathematical computing architecture[C]//High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on. IEEE, 2016: 457-468.