

Improving the Scalability of SAMBA Bus Architecture

Ruibing Lu, Aiqun Cao *

Synopsys, Inc.
 Mountain View, CA, 94043, USA
 {Ruibing.Lu, Aiqun.Cao}@synopsys.com

Cheng-Kok Koh

ECE, Purdue University
 West Lafayette, IN, 47907, USA
 chengkok@ecn.purdue.edu

Abstract— SAMBA bus [1] is a high performance bus architecture that can deliver multiple transactions in one bus cycle under single-winner bus arbitration. The bus architecture displays several advantages such as, high bandwidth, low latency, and low performance penalty from arbitration delay, all of which make it more scalable than traditional buses. However, its scalability may be limited by the bus access logic delay. As a module is connected to the bus through its interface unit, which is connected in series on the bus, the bus logic delay increases linearly as the bus size increases. In this paper, we propose to increase the scalability of SAMBA buses through two methods: control signal lookahead and module clustering. The control signal lookahead technique can determine the bus access control signal in advance, thereby reducing the effective delay of each interface unit. Module clustering, on the other hand, can reduce the number of interface units attached to a bus. Experimental results show that combining these two methods can effectively reduce the bus logic delay, and thus increase the scalability of SAMBA buses.

I. INTRODUCTION

Shared-buses are among the most widely used communication architectures for System-on-Chips. The main advantages of shared-bus architectures include simple topology, low cost, and extensibility. Several companies have developed their own on-chip bus architectures, such as CoreConnect [2], AMBA [3], and OCP [4]. An overview of current SoC bus architectures is available in [5].

The main problem of shared-bus communications is that the performance of buses decreases significantly when the bus size (*i.e.*, the numbers of modules on buses) increases. This is the so-called scalability problem. The low scalability of shared buses primarily comes from two reasons: bus bandwidth and arbitration delay. Because a bus can be used by only one module at any time, the available bandwidth of one module decreases significantly as the bus size increases. Bus arbitration is another bottleneck for the scalability. A module seeking for communication must first obtain the bus access grant from the bus arbiter. This may introduce a long delay, which includes the arbitration delay, the interconnect delay from a module to the arbiter, and that from the arbiter to the module. The bus clock cycle time may have to be significantly increased due to the delay introduced by the arbitration process.

In order improve bus scalability, bus architectures capable of delivering multiple transactions in one bus cycle are pro-

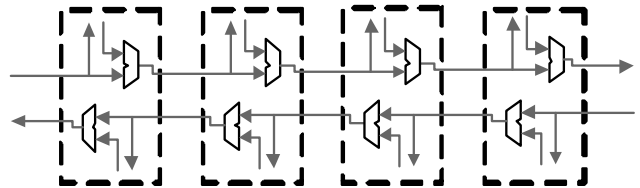


Fig. 1. Structure of SAMBA Bus from [1].

posed in [1, 6]. For buses with larger sizes, there are higher possibilities to have multiple transactions performed simultaneously. Therefore, the effective bandwidth increases when the bus size increases, thereby improving the bandwidth scalability. In [6], complex two level arbitration and tri-state buffers are required for the proposed bi-directional split-bus architecture, which may bring some difficulties for circuit design and testing.

The uni-directional SAMBA bus architecture proposed in [1] can perform multiple transactions simultaneously with any traditional single-winner arbitration scheme. The overall structure of SAMBA bus architecture is shown in Fig. 1. It consists of two sub-buses, a forward sub-bus and a backward sub-bus, with opposite signal propagation directions. For convenience, the addresses of modules from the left end to the right end are assumed to be in an increasing order. All bus structures and operations are symmetric for the two sub-buses. Therefore, the structure and operations of only the forward sub-bus will be discussed.

A module is attached to the bus through a corresponding interface unit. As shown in the Fig. 1, multiplexers are used to combine all signal sources, and each interface unit has a multiplexer on one sub-bus. Through the multiplexer, either the address/data information received from the previous unit or the pending address/data of this unit are propagated to the next interface unit.

An interface unit can access forward sub-bus for its pending communication in the following three situations:

1. This unit is the arbitration winner of the forward sub-bus.
2. (a) The address of the destination of its pending communication is lower than or equal to that of the arbitration winner; and (b) there is no bus transaction passing through this unit.
3. (a) This unit is to the right of the arbitration winner; and (b) there is no bus transaction passing through this unit.

*This work was performed while at School of Electrical Engineering, Purdue University

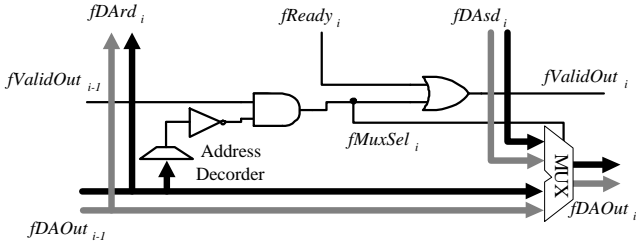


Fig. 2. Bus access logic for the forward sub-bus adapted from [1].

Bus transactions belonging to the second or third situations are called compatible transactions.

Due to the ability to perform multiple transactions simultaneously, SAMBA buses have much better bandwidth performance. Another interesting property is that the communication latency of SAMBA buses is affected only slightly by long arbitration latency. Therefore, SAMBA bus architecture is more scalable and desirable in SoC designs with large numbers of modules and long communication delay between modules and the bus arbiter. However, a problem that SAMBA bus architecture encounters is that all interface units are connected in series along the bus. Consequently, the bus clock cycle time may have to be increased significantly due to the logic delay of interface units when the bus size is large.

In this paper, we propose two compatible methods, control signal lookahead and module clustering, to further improve the scalability of SAMBA buses by reducing the bus logic delay. The lookahead technique can reduce the average delay of each interface unit, while module clustering can reduce the number of interface units on the bus. The bus logic delay can be reduced significantly when both of them are used, thereby improving the scalability of SAMBA buses.

II. CONTROL SIGNAL LOOKAHEAD

The bus access rules of SAMBA buses can be achieved through an interface unit shown in Fig. 2.

The data/address received from interface unit $(i-1)$ is propagate to the next unit through $fDAOut_i$ (the output of interface unit i onto the forward bus) if and only if the output from interface unit i is valid ($fValidOut_{i-1}$ is asserted) and the output of address decoder is negated, or

$$fMuxSel_i = fValidOut_{i-1} \wedge \overline{(fAddrOut_{i-1} = i)}.$$

Signal $fReady_i$ is an internal signal generated by the interface unit. It is asserted if (a) the data/address on $fDAsd_i$ is the valid data/address of a pending communication, and (b) the source module is the arbitration winner, a module after the arbitration winner, or a module whose transaction destination is not after the arbitration winner. A pending communication satisfying the above conditions is called a ready communication. The bus may transmit a pending communication if and only if it is a ready one and there is no valid transaction passing through the source interface unit. Signal $fValidOut_i$, which is equal to $fMuxSel_i \vee fReady_i$ informs the next interface unit whether the data/address on $fDAOut_i$ is valid.

What an interface unit shown in Fig. 2 introduces to the critical path delay of the entire bus is the delay of the path from

the address decoder to the multiplexer control, and then to the multiplexer output. As interface units are connected in series, the bus cycle time may be significantly increased for SAMBA buses with large numbers of interface units, thus decreasing bus performance and scalability.

We propose a control signal lookahead method to reduce the delay introduced by interface units. The central function of an interface unit is to decide how to control the multiplexers. Hence, if we can obtain the multiplexer control signals in advance, without waiting for the output of neighbor interface units on the address bus, the delay can be reduced.

Consider a forward interface unit i . There are two scenarios when the interface unit should negate $fMuxSel_i$, which means that the data/address $fDAOut_{i-1}$ from previous interface unit should not be passed through interface unit i onto $fDAOut_i$:

1. Data/Address on $fDAOut_{i-1}$ is not from a valid transaction, i.e., $fValidOut_{i-1}$ is negated.
2. Data/Address on $fDAOut_{i-1}$ is a valid transaction whose destination is interface unit i , i.e., $(fValidOut_{i-1} \wedge (fAddrOut_{i-1} = i))$ is asserted.

In the first scenario, the negation of $fValidOut_{i-1}$ is equivalent to $\overline{(fMuxSel_{i-1} \wedge fReady_{i-1})}$, which means that there is neither transaction passing through unit $(i-1)$ nor ready communication from unit $(i-1)$. In the second scenario, the valid transaction is from either unit $(i-1)$ (i.e. $\overline{(fMuxSel_{i-1} \wedge fReady_{i-1} \wedge (fAddrOut_{i-1} = i))}$ is asserted), or an interface unit before unit $(i-1)$ (i.e. $(fMuxSel_{i-1} \wedge (fAddrOut_{i-2} = i))$ is asserted). Therefore, we have:

$$\begin{aligned} \overline{fMuxSel_i} &= \\ & \overline{(fMuxSel_{i-1} \wedge fReady_{i-1})} \\ & \vee \overline{(fMuxSel_{i-1} \wedge fReady_{i-1} \wedge (fAddrOut_{i-1} = i))} \\ & \vee (fMuxSel_{i-1} \wedge (fAddrOut_{i-2} = i)) \\ & = \overline{(fMuxSel_{i-1} \wedge (fReady_{i-1} \vee (fAddrOut_{i-1} = i)))} \\ & \vee (fMuxSel_{i-1} \wedge (fAddrOut_{i-2} = i)). \end{aligned} \quad (1)$$

Note that in (1), all the signals required for determining the value of $fMuxSel_i$ are available in the interface unit $(i-1)$. Therefore, we can determine $fMuxSel_i$ based on signal values in previous interface unit $(i-1)$ without waiting for $fAddrOut_{i-1}$. This is called 1-stage lookahead, or single stage lookahead. Note that, in (1), the value of $\overline{(fReady_{i-1} \vee (fAddrOut_{i-1} = i))}$ is dependent only on the pending communication of interface unit $(i-1)$; hence, the delay of $fMuxSel_i$ is dominated by that of $fMuxSel_{i-1}$ and $fAddrOut_{i-2}$.

The lookahead technique can be extended for multiple stages. For a general n -stage lookahead, the transaction whose destination is interface unit i may be initiated by interface units $\{i-1, i-2, \dots, i-n\}$ or from $fDAOut_{i-(n+1)}$. Therefore, we have:

$$\begin{aligned} \overline{fMuxSel_i} &= \\ & \overline{(fMuxSel_{i-1} \wedge (fReady_{i-1} \vee (fAddrOut_{i-1} = i)))} \\ & \vee \overline{(fMuxSel_{i-1} \wedge fReady_{i-1} \wedge (fAddrOut_{i-1} = i))} \\ & \dots \\ & \vee \overline{(fMuxSel_{i-n} \wedge fReady_{i-n} \wedge (fAddrOut_{i-n} = i))} \\ & \vee (fMuxSel_{i-n} \wedge (fAddrOut_{i-(n+1)} = i)) \end{aligned} \quad (2)$$

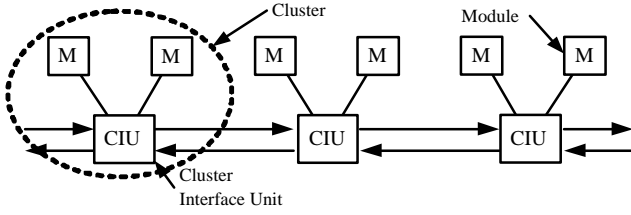


Fig. 3. A SAMBA bus with module clustering.

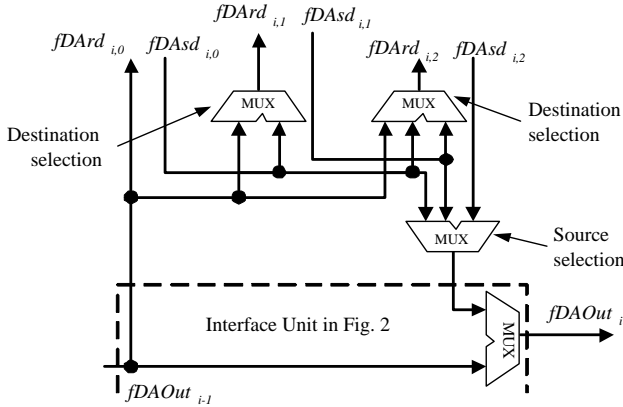


Fig. 4. The structure of the forward component of a cluster interface unit with a cluster size of 3.

In (2), we can see that the decoding of transactions to interface unit i now distributed in interface units from $(i - n)$ to $(i - 1)$. This introduces some additional logic and interconnect cost. However, considering large resource used by buses, such additional cost is minor. As we shall see from the experimental results in Section IV. The lookahead method can significantly reduce the delay from bus access logics.

III. MODULE CLUSTERING

The control signal lookahead technique can be used to decrease the delay of one interface unit. In this section, we propose a module clustering technique to reduce the bus logic delay by reducing the number of interface units on the bus.

The main objective of module clustering is to group several modules into one cluster, and all modules in one cluster shares a cluster interface unit to access the bus. Fig. 3 shows an example SAMBA bus with module clustering. Three clusters are shown in Fig. 3. And each of them has a cluster size of two, *i.e.*, there are two modules in one cluster.

A cluster interface units is composed of one forward component and one backward component, each of which is in charge of the data propagation in one direction. Each cluster interface unit performs three functions: intra-cluster communication control, selection among pending inter-cluster communication requests, and bus access control for inter-cluster communications. Fig. 4 shows the structure of the forward component of a cluster interface unit with a cluster size of 3.

The management of either intra-cluster communication or inter-cluster communications is through multiplexers as shown in Fig. 4. As in un-clustered SAMBA buses, only a ready communication may be performed, be it intra-cluster or inter-

cluster.

The cluster interface unit shown in Fig. 4 utilizes a point-to-point connection for intra-cluster communication. Through the forward component of a cluster interface unit, a module has direct connections to other modules on its right side. Therefore, the ready intra-cluster communication can always be performed as long as there is no conflict at the destination. The conflict occurs when multiple modules are trying to access the same destination module in the cluster. The selection among these multiple ready communications sent to the same destination is called *destination selection*. The destination selection is performed through a priority based method. In the forward component of a cluster interface unit, transactions from left modules have higher priority.

For a cluster, it may have multiple ready inter-cluster transactions from its modules to be put on the bus. The selection of one among these ready inter-cluster communication is called *source selection*. Again, the source selection is priority based, where left modules have higher priorities in the forward component. The source selection is dependent only on the pending communications of modules inside one cluster. Therefore, it can be performed in parallel and have little impact on bus clock cycle.

The bus access control part is performed by an interface unit, which is shown in Fig. 2. It treats the addresses of all modules in the cluster as its addresses, and takes the source selection winner as its ready communication.

Module clustering may introduce additional logic cost. However, the cost is typical minor for small cluster sizes and can be compensated by the reduction of interface units. Because point-to-point connection is used for intra-cluster communication, the logic complexity is quadratically proportional to the cluster size. Fortunately, a small cluster size can also achieve large reduction on the logic delay of SAMBA buses. With a uniform cluster size of c , the number of interface units on a clustered bus becomes $1/c$ of that on the corresponding un-clustered bus. Therefore, module clustering can significantly reduce the number of interface units on the bus, and thus reduce the logic delay along SAMBA buses.

Another important property of module clustering is that it can never reduce the possibilities of compatible transactions. From [1], SAMBA buses with more interface units have better effective bandwidth because there are more combinations to have compatible transactions. Although the number of interface units on a clustered SAMBA bus decreases, all compatible transactions on a un-clustered SAMBA bus are still compatible on the corresponding clustered SAMBA bus. In fact, module clustering brings more possibilities to have compatible transactions. Fig. 5 shows two examples of compatible transactions on clustered SAMBA buses, which are incompatible on SAMBA buses without clustering. In other words, clustered SAMBA buses can further improve the bus bandwidth, especially when there are heavy traffic among modules inside one cluster.

The module clustering can reduce the number of interface units on the bus, while the lookahead method in Section II is to reduce the delay of one interface unit. Therefore, these two methods can be combined together to further reduce bus clock cycle time.

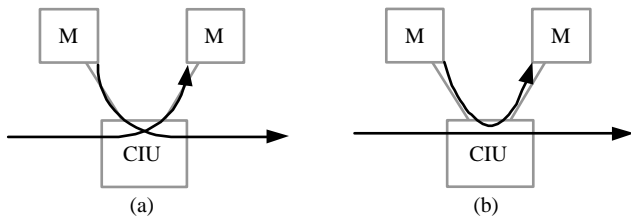


Fig. 5. Examples of compatible transactions on a clustered SAMBA bus but not incompatible on the corresponding SAMBA bus without clustering.

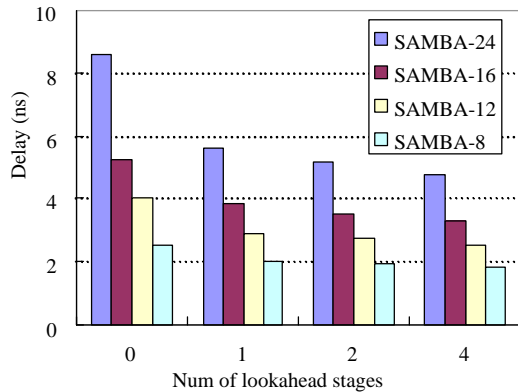


Fig. 6. Logic delay of SAMBA buses with 24, 16, 12 and 8 modules.

IV. EXPERIMENTAL RESULTS

In this section, we present our experiments and results. The first experiment is on the bus logic delay of SAMBA buses, and the effect of control signal lookahead with different lookahead stages. The second experiment studies the bandwidth and latency performance of clustered SAMBA buses in terms of bus cycles.

In the first experiment, we implement the bus architecture through Verilog and mapped the design into TSMC $0.18\mu\text{m}$ technology through Synopsys Design Compiler. We implement SAMBA buses with 24, 16, 12, and 8 interface units and with lookahead stages of 0, 1, 2 and 4. Here 0 stage lookahead means that the lookahead technique is not used.

The logic delay due to the interface units along the bus is shown in Fig. 6. First, we can see that lookahead method can effectively reduce the logic delay by 20% to 45%. Second, as expected, the logic delay along the bus increases linearly as bus sizes increase. Therefore, module clustering is effective in reducing bus logic delay as it reduce the number of interface units greatly. For example, consider a SAMBA bus with 24 interface units. Without lookahead or clustering, bus logic delay is around 8.6ns . The delay becomes near 2.9ns for a clustered SAMBA bus with a uniform clustering size of 2 and 1-stage lookahead. The reduction is more than 66%. With a uniform cluster size of 3 and 1-stage lookahead, the logic delay becomes around 2ns , which translates into a 76% reduction.

While the primary objective of module clustering is the reduction of bus logic delay, the improvement on the bus bandwidth performance is also notable. Our second experiment studies the performance of clustered SAMBA buses through

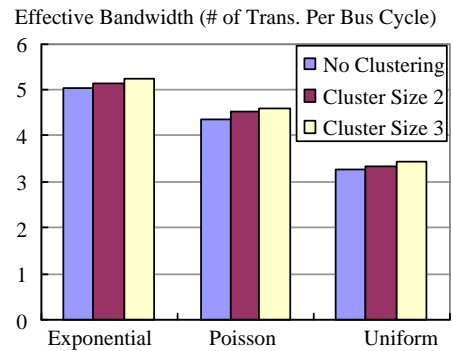


Fig. 7. Effective bandwidth of SAMBA buses with 24 modules. X axis is for three different communication distance distribution.

simulation as in [1, 7]. The simulated buses all have 24 modules, each of which can either initiate bus transactions or respond to transactions initiated by others. Bus communication traffic is generated randomly followed a method in [1, 7, 8], which is controlled by three different communication distance distributions: Uniform, Poisson, or Exponential.

The effective bandwidth, which is defined as the average number bus transactions performed in one bus cycle, is shown in Fig. 7. Although the number of interface units on the bus decreases, the performance of clustered SAMBA buses does not decrease, the performance actually increases with larger cluster sizes. Note that here the performance unit is based on the bus cycle. When the bus cycle time is considered, the performance improvement is much more significant.

V. CONCLUSION

This paper proposes two methods, control signal lookahead and module clustering, to improve the scalability of SAMBA buses. Experimental results shown that the two methods can achieve substantial performance improvement when combined together.

REFERENCES

- [1] R. Lu and C.-K. Koh. SAMBA-Bus, a high performance bus architecture for System-on-Chips. In *Proc. Int. Conf. on Computer Aided Design*, 2003.
- [2] IBM. *CoreConnect Bus Architecture*, 1999.
- [3] ARM, Limited. *AMBA Specification*, 1999.
- [4] Sonics, Inc. *Open Core Protocol Specification*, 1999.
- [5] E. Salminen, V. Lahtinen, K. Kuusilinna, and T. Hamalainen. Overview of bus-based system-on-chip interconnections. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages II-372 – II-375, 2002.
- [6] R. Lu and C.-K. Koh. A high performance bus communication architecture through bus splitting. In *Proc. Asia South Pacific Design Automation Conf.*, pages 751–755, 2004.
- [7] K. Lahiri, A. Raghunathan, and G. Lakshminarayana. LOTTERYBUS: A new high-performance communication architecture for System-on-Chip designs. In *Proc. Design Automation Conf.*, 2001.
- [8] C. Hsieh and M. Pedram. Architectural energy optimization by bus splitting. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21:408–414, April 2002.