HIBI v.2 - An Interconnection for the Network-on-Chip Era

Erno Salminen, Vesa Lahtinen, Tero Kangas, Jouni Riihimäki,
Kimmo Kuusilinna, and Timo D. Hämäläinen
Tampere University of Technology, Institute of Digital and Computer Systems
P.O. Box 553, FIN-33101, Tampere, Finland
Tel: + 358-3-3115 4540, Fax: +358-3-3115 4575
Corresponding author: Erno Salminen

Abstract-This paper presents a network-on-chip (NoC) architecture scheme targeted for complex system-on-chip designs. The proposed Heterogeneous IP Block Interconnection v.2 (HIBI) aims at maximum efficiency and energy saving per transmitted bit combined to guaranteed quality-of-service (QoS) in transfers. Other features include support for arbitrary topologies with several clock domains, flexible scalablility in signalling and run time reconfiguration of interconnection parameters. HIBI has been implemented in VHDL and SystemC and synthesized in 0.18 μm CMOS technology with area comparable to other NoC interconnection wrappers. HIBI data transfers are shown to approach the maximum theoretical performance for protocol efficiency.

Index Terms - High performance, configurable, communication, digital, design, distributed control, Vlsi

I. INTRODUCTION

Network-on-chip (NoC) can be regarded as the third wave of system-on-chip (SoC) designs after intellectual property (IP) block-based [1][2] and platform-based [3] methodologies. The third wave focuses on how to reuse not only IP blocks and platforms but also on-chip interconnections. This is a formidable challenge, since contemporary SoCs, such as MoVa [4] and Viper [5], have very heterogeneous structure. In general, SoCs include several different types of processors, memories and dozens of IP blocks that have different interface signalling. In addition, there are several clock domains that are not necessarily synchronized and some of the blocks are from time to time shut down for power saving. The overall computation is not homogeneous as in traditional multiprocessors running scientific algorithms and, therefore, a symmetric interconnection topology may be insufficient. Quality-of-service (QoS) for on-chip data transfers is a new demand that originates from real-time requirements in applications.

Acknowledgments
This research work has been supported by the Graduate School in Telecommunication System-on-Chip Integration (TELESOC) and the Nokia Foundation.

As a general principle, *orthogonalization of concerns* [3][6] facilitates partitioning the design into manageable subdesigns. For instance, separation between computation and communication helps to select an appropriate interconnection in an application independent manner. This principle forms the basis for automated design space exploration [7].

Standardization efforts include a layered approach based on IP interfaces, for example *Virtual Component Interface* (VCI) [8] and *Open Core Protocol* (OCP) [9]. A component called *wrapper* translates the IP block specific communication operations to corresponding on-chip network operations. Figure 1 illustrates the concept with a HIBI network.

Interconnections, as a research topic, have well established and long traditions originating from board level multi-processor systems and networked multicomputers. Interestingly, many topologies that were originally intended for super computers [10][11][12] are now proposed as backbones for SoCs. However, their original form may lead to unexpected problems in SoC environment. A regular and symmetric topology, such as a mesh, fits well for dataflow type of processing with static scheduling and allocation of tasks. Other types of computation or unbalanced loads may lead to significant overall inefficiency.

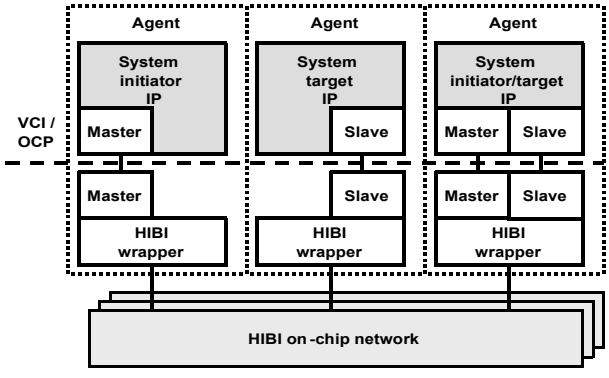

Figure 1. Utilizing VCI/OCP with HIBI network

HIBI v.2 addresses the challenge by presenting methods for guaranteed transfer capacity, fast run-time reconfiguration, and efficiency in terms of energy and performance per transmitted bit. HIBI v.2 is used to construct modular, hierarchical structures with distributed arbitration and multiple clock domains. We begin with a review of interconnection design issues and related work. Motivation, key design constraints, and advanced features of HIBI v.2 and its implementation are described in Sections 3 and 4. Low-level implementation details, such as bus signal encoding schemes, are beyond the scope of this paper. Finally, in Section 5, synthesis results and performance analysis are given with the help of a case study. Section 6 concludes the paper.

II. RELATED WORK

Traditionally, on-chip interconnections have utilized bus due to its simple and small implementation. The shared nature and signal propagation delay in deep sub-micron technologies will restrict the size of a practical single bus. AMBA [13] is a centrally arbited on-chip bus that supports two-level hierarchical structures. Silicon Backplane [14] bus uses distributed, two-level arbitration and pipelined transfers. It does not use FIFO buffers or bridging between multiple Silicon Backplane buses. Authors have carried out a study on basic features of on-chip buses [15] which indicated that many contemporary buses are based on fixed constraints, for example number of agents and the available bus widths is limited, leading to constrained scalability. We believe that it is better to keep such properties parameterizable to cover a wider range of applications and let application and implementation technology define the limits instead of the protocol. Therefore, a parameterizable and configurable interconnection architecture is needed.

Lately, a lot of research has gone into adopting packet-switched NoCs [16][17]. Proteo network is a topology-free packet-switched NoC concept [17]. The current implementation suffers from large buffering overhead. Moraes *et al.* propose a mesh-based packet-switched NoC utilizing worm-hole routing [18]. Their best-effort routing algorithm has a notable latency of ten cycles per switch. Router intoduced by Rijpkema *et al.* offers guaranteed service with time division multiplexing [19]. SPIN network uses a packet-switched fat-tree network [20]. Valtonen *et al.* consider a homogeneous cellular NoC with special attention on testability and error tolerance [21]. There are two major problems with contemporary NoCs: large area due to buffering inside communication network, and lack of providing guaranteed service.

In terms of implementation complexity, hybrid interconnections, such as hierarchical buses, lie somewhere in between the circuit- and packet-switched network. They have many properties of traditional buses but also incorporate features that are typical for packet-switched networks. Multiple bus segments can also be connected together in a circuit-switched manner, for example, such approach is used between AMBA AHB and APB [13]. Maximum number of simultaneous transfers in a hierarchical bus architecture equals the number of bus segments when segments are not circuit-switched together.

Zeferino *et al.* [22] have analyzed the switching point when mesh-based NoC becomes preferable over simple bus having no hierarchy. Using the same equations, Figure 2 shows clock cycles for total exchange for mesh-based NoC, hierarchical two-bus and four-bus networks with different degrees of locality. Locality is expressed as proportion of reachable agents. Figure 2 shows that even a two-bus network will offer comparable performance to a mesh-based NoC when at least half of the agents must be reached. If locality or the number of agents is higher, a system having more bus segments could be chosen. This suggests that the switching point will shift to a higher number of agents than presented in [22].

In hierarchical networks, the mapping of processes onto computation resources is a critical step to keep most of the communication within a small area and still ensure load balancing. A fixed topology does not allow application-specific optimization of communication topology. Optimization may be targeted toward performance by adding more communication links when needed or toward area by removing links to avoid an overkill topology. Therefore, the topology should be scalable and modified easily to support various applications efficiently.

The choice of the most suitable topology depends on the requirements of the targeted application, that is whether it requires many parallel channels or not. However, even if dedicated parallel channels would be preferable for performance reasons, implementation complexity and area are limiting factors. In addition, functional units, particularly register files [23] and memories, that utilize more than one simultaneous connections may become prohibitively complex. As a result, no fixed topology

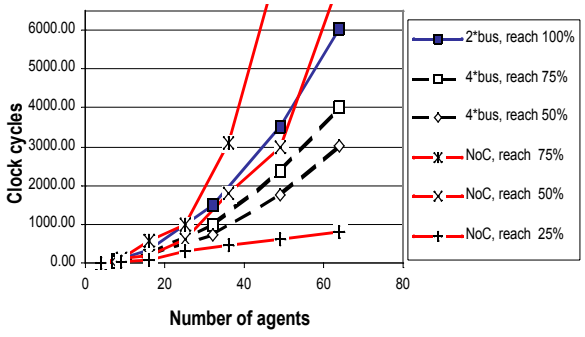

Figure 2. Clock cycles for total exchange

perfectly fits all applications and knowledge about the application is needed when choosing the topology.

III. HIBI V.2 OVERVIEW

The design objectives behind HIBI v.2 were to design a topology-independent, scalable, and still high-performance interconnection for systems consisting of heterogeneous functional units. Special attention was paid to configurability, efficiency of the protocol, and quality of service. It is based on previously presented *Heterogeneous IP Block Interconnection* v.1 (HIBI v.1) [24][25].

A. Buffering scheme

The model of computation used in HIBI design flow assumes bounded first-in-first-out (FIFO) buffers between processes. A simple FIFO interface can be adapted to more complex interfaces such as the VCI. Functional units can write data to FIFO buffers regardless of the state of the communication channel if the FIFO is large enough. Such a method is particularly suitable for posted-write operation, in which the data is sent without waiting for acknowledgement from the target.

Buffering can constitute a large proportion of the wrapper area. To avoid excess buffering, the received data should be read from the FIFO as soon as possible. Using a direct memory access (DMA) controller, data flows through the FIFO into the local memory of the functional unit. Functional unit is notified only when the whole packet has been transferred to memory. Therefore, the buffer space in HIBI wrapper is not defined by the *amount* of transferred data, but the *latency* of reading data from the wrapper.

B. Reuse and signaling

In HIBI architectures, the same wrapper may be used with every agent by changing its parameters facilitating composability and reuse. The wrapper interfaces do not depend on the number of agents because all signals are shared and there is no dedicated point-to-point signalling between wrappers. A pre-designed HIBI layout may be reused due to constant pin interfaces. Configuration can be carried out in run-time. Arbitration is distributed to wrappers meaning that they can decide the correct time to access the bus by themselves and no central arbiter is required. All wrappers implement both the master and the slave functionality, which is required in *split transactions*. In split transaction, the bus is not blocked while waiting the response from the target, but other agents may use it.

Signaling on the bus and on the IP interface are kept at minimum to make integration easier. HIBI protocol does not require any specific control signals, such as memory invalidate. If control type signalling is needed, message-passing through the data lines is utilized. If control type messaging becomes a bottleneck, it is possible to concatenate control signals to transmitted data and adapt the data bus width accordingly.

C. Quality-of-Service

In HIBI, a two-level arbitration scheme, a combination of time division multiple access (TDMA) and competition, is used. In TDMA, time is divided into repeating time frames. Inside frames, agents are provided time slots when they are guaranteed an access to the communication channel. Competition is based either on round-robin or priority. The second level mechanism is used to arbitrate the unassigned or unused time slots. Priority arbitration as a second level method attempts to guarantee a small latency for high priority agents whereas round-robin provides a fair arbitration scheme. The HIBI implementation pays special attention to minimizing the arbitration latency by removing *empty cycles* from the arbitration process by pipelining.

Basic TDMA may waste bandwidth due to unnecessary reservations. Round-robin does not have such problem, but there is more variation in transmission times since there are no repeatable time frames.

D. Run-time reconfiguration

Applications have varying requirements for communication. In addition, the requirements may vary between different phases of a single application. Therefore, HIBI allows the run-time configuration of all arbitration parameters to minimize latency by interconnection tuning. The length of reconfiguration procedure determines how often it is beneficial to change communication parameters.

Synchronizing cycle counters is a way to adapt to the varying execution times. Specifically, it prevents the time between data availability and the corresponding time slot from getting repeatedly too long and offers better guarantees for available bandwidth. In resynchronization, the cycle counters can be reset to an arbitrary clock cycle value within the time frame. One solution for triggering resynchronization process is to provide a specific monitor unit, which is aware of all the time slots. It monitors how effectively time slots are used and starts the reconfiguration if needed [26]. Furthermore in HIBI v.2, the second-level arbitration method may be changed at run-time between priority and round-robin or both of them can be disabled. When the second-level arbitration is disabled, only basic TDMA is used and a HIBI wrapper reserves the bus always for the whole allocated time-slot.

Three methods are used to decrease configuration latency. First, by making use of the bus nature, each common parameter can be broadcast to all wrappers in one operation. Second, enabling the reading of configuration values simplifies the procedure as the whole configuration does not have to be stored in the configuring device. The configuring agent can read the old parameter values to help

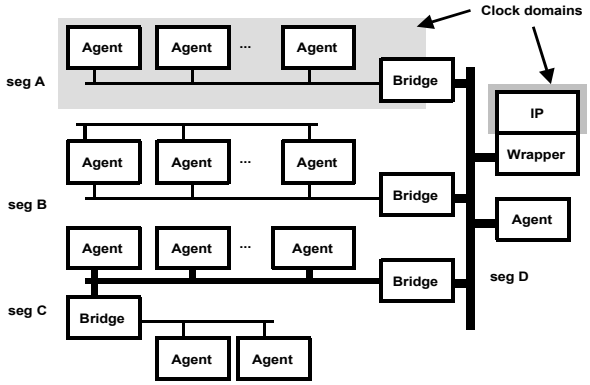

Figure 3. Hierarchical HIBI network

determine the new. Third, additional storage capacity for multiple parameter pages has been added to enable rapid change of all parameters. When a configuration page changes, all the parameters in interconnection are updated immediately. It is possible to store a specific configuration for every application (phase) in its own configuration page to enable fast configuration switching.

E. Circuit and packet-switching

Bus throughput can be scaled up by using hierarchical bus structures as shown in Section 2. Figure 3 depicts a system with hierarchical HIBI network. The key is to locate the agents that communicate frequently in the same segment to provide small latency. Although accessing distant segments results in longer latency, the overall system performance increases provided that most of the accesses are within a single segment. Applications having strong locality benefit the most from the hierarchical approach. Clustering makes the optimization easier since segments can operate with various data widths and clock frequencies. The implementation area and power consumption are lowered by making the segments as slow and narrow as possible but still meeting the application requirements.

HIBI bridges work utilizing the packet-switched principle so that bus segments are not circuit-switched together. Instead, the data is stored inside the bridge until it gets an access to the other segment. If the bridge cannot buffer all the data, the transfer is interrupted until the bridge has written some of the data to the next bus segment. Once the data is stored inside the bridge, the source segment is free for other transfers. It is possible that a bridge buffers parts from multiple transfers.

F. Data transfer operations

In HIBI v.2, all operations can be targeted either to a single agent or to a group of agents with multicast. The upper bits of the incoming address are checked against a pre-defined bit mask. Agents having the same bit mask for

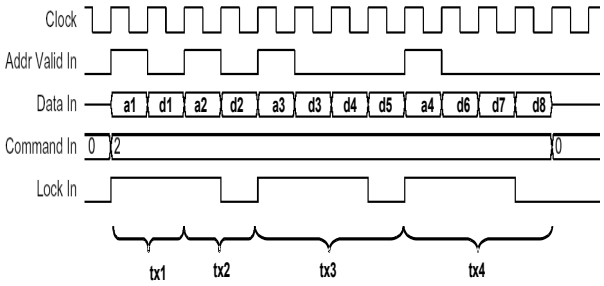

Figure 4. Example of four writes from three sources

the uppermost bits of their address ranges will accept the transfers. The multicast operation uses the two lowermost address bits to determine the comparison range. Multicast can be used for broadcasting by setting comparison range to zero.

By prioritizing data transfers, smaller latency is achieved for high priority data to support QoS. In HIBI v.2, data may be sent with two priorities. High priority data, such as control messages, bypass the normal data transfers resulting in smaller latency. This does not change the timing of bus reservations, but it selects what is transferred first. This procedure can be extended as shown in Section 4.

HIBI v.2 has multiplexed address and data lines whereas HIBI v.1 used separate address and data lines. One cycle addressing latency is negligible in long transfers. HIBI v.2 implementations can be made smaller than HIBI v.1 because signal lines are removed and less storage capacity is needed for the addresses. On the other hand, the saved area can be used for wider data transfers to increase the available bandwidth.

Figure 4 depicts four consecutive transfers made by three different agents. First, agent 1 transfers two data to different targets and then agents 2 and 3 transfer three data words. All transfers start with an address. There are no empty cycles between transfers due to efficient, pipelined arbitration.

HIBI v.1 omitted handshake signals to minimize transfer latency. This was achieved with fully distributed arbitration and with strictly non-blocking transfers. If handshaking was really needed, for example at application level, it could be carried out with message passing. To avoid overflows with all buffer sizes, one handshake signal was added to HIBI v.2. The signal interrupts the transfer when the receiving wrapper cannot accept more data and the sender retransfers the data until it is accepted. As a result, blocking models of computation can be used in system design and the depth of FIFOs is considerably smaller.

G. Energy saving

In many cases, implementation technology dependent methods have been found inadequate as the sole method for addressing the power consumption problem. Therefore, energy saving must be supported on all design abstraction levels [27] [28]. TDMA arbitration is beneficial because transfer times are known in advance and, therefore, timing shutdown and predictive power-up operations is simple. The application can actively set the power save modes through configuration memory. Due to the decentralized arbitration of the HIBI wrapper, at least the arbitration logic of all wrappers must be running all the time, but other parts can be in power save mode either keeping their state (clock gating) or having their power totally shut down. In the latter case, the contents of the buffers and all state information are lost. When some wrappers are switched off, more bandwidth is available for others to use.

IV. HIBI V.2 IMPLEMENTATION

HIBI v.2 was implemented with synthesizable VHDL and SystemC. In addition, SystemC simulation model has an embedded performance monitor which is used to analyze and tune the architecture to better fit the application.

A. Signaling

The bus signals are *Reset*, *Clock*, *Data*, *Address Valid*, *Command*, *Lock*, and *Target Full*. The number of data bits can be freely chosen. Signal Address Valid indicates when an address is transferred on the data lines. Three bits are needed for Command to present eight different network operations. Handshaking is done with the Target Full signal. All signals are shared and no point-to-point signalling is required.

There are two FIFO interfaces to IP side in both directions, one for regular data and the other for messages. Furthermore, the power control signals can be routed out of the wrapper if the IP block can utilize them. An OR network was selected for bus signal resolution.

B. Wrapper structure

HIBI v.2 wrapper is depicted in Figure 5. The modular wrapper structure may be tuned to better meet the application requirements by using different versions of the internal units or even leaving out properties that are not needed in a particular application. The main parts are buffers for transferring and receiving data and the corresponding controllers. The transfer controller includes configuration memory for storing all arbitration and addressing parameters. The receive controller includes a separate address decoding logic. All main parts are synchronous, having registered outputs. Only the FIFO

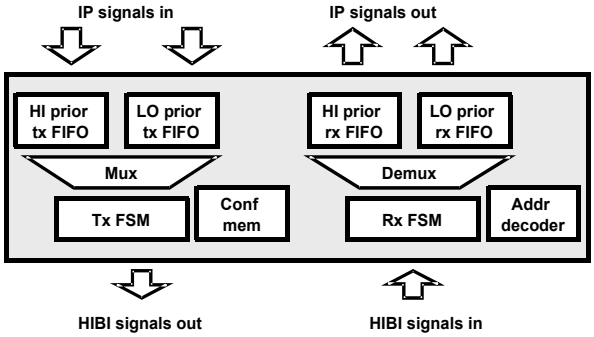

Figure 5. HIBI v2 wrapper

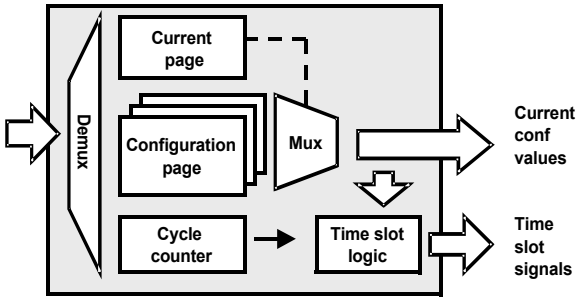

Figure 6. HIBI v.2 configuration memory

(de)multiplexer outputs and Target Full signal from receive controller are asynchronous.

Message bypassing is implemented by adding an extra FIFO along with the data FIFO. A multiplexer is placed between the transmit FIFOs and the transfer controller so that the controller sees only a single FIFO that it can access. The separate multiplexer allows this method to be expanded to support priorities in excess of two without changing the control.

The structure of configuration memory is illustrated in Figure 6. It includes multiple configuration pages for storing the parameter values, a register storing the number of currently active page, and a clock cycle counter. Current configuration values and time slot signals are fed to the transfer controller. The receive controller takes care of writing new values to the configuration memory.

Configuration values can be written to non-active pages before they are used to minimize the risk of conflict when the configuration procedure is yet to be completed. It also amortizes the configuration delay over a longer time period as other transfers can be carried out in the middle of the configuration procedure. Naturally, having multiple pages results in larger implementation area. A way to reduce area is to restrict the configuration options by hardwiring values. Still, it is possible to leave some parameters to be configured at run-time. For example, allowing clock counter synchronization and the use of multiple hardwired

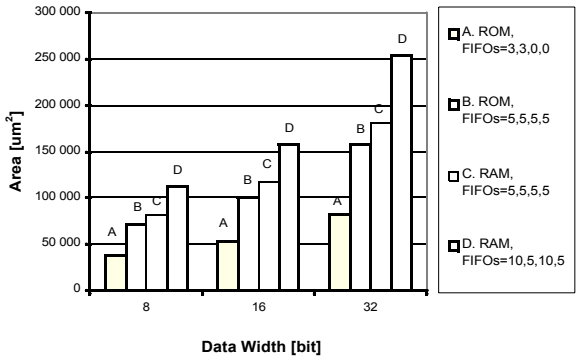

Figure 7. Hibi v.2 wrapper areas

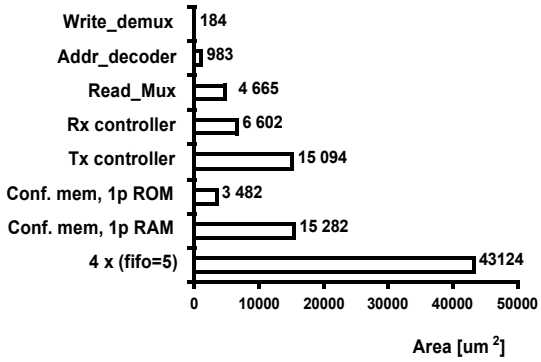

Figure 8. Sub-block areas of 8-bit HIBI v.2 wrapper

configuration pages might be a good compromise for many applications.

C. Structure of HIBI bridge

A bridge is constructed by connecting two wrappers together. If the connected bus segments use different clock frequencies, handshaking logic is needed between the two wrappers. An option is to mix channels of different widths in a hierarchical system in which the bridges are responsible for the data width adaptation. Segments having only simple peripheral devices could use a slow and narrow bus while the main processing parts have higher capacity buses.

V. HIBI V.2 PERFORMANCE

A. Synthesis results

The HIBI v.2 wrapper was synthesized using a 0.18 micron technology. The results do not include the wiring or place-and-route information. Areas for four different wrapper configurations with three different data widths are shown in Figure 7. FIFO definition "3,3,0,0" means that data buffer sizes are 3 words and message FIFOs are absent. Both A and B cases use a hardwired configuration. Version C and D allow new configuration values to be written at run-time. Furthermore, version D has two configuration pages. Version A is the smallest being roughly half the size of versions B and C and about one third of version D. The biggest wrapper (32 bit version D) occupies roughly a quarter of a square millimeter (about 20k 2-NAND gates). The maximum frequency is in the range of 200 to 300 MHz depending sizes of FIFO buffers and configuration memory used.

Currently the configuration memories and FIFO buffers are implemented with flip-flops to support soft IP core methodology, which results in rather large area. This is illustrated in Figure 8 for an eight-bit wrapper having five-word FIFO buffers and one-page configuration memories. In this case, the FIFO buffers occupy half of the wrapper area. Still, the area of FIFO buffers is smaller than in the first version of HIBI. The number of configuration pages, time slots, and the width of parameters can be parameterized according to the application to minimize the implementation area. Figure 8 confirms that hard-wiring parameter values offers a great reduction in area. Register-based configuration memory is denoted with RAM and hardwired with ROM. Table 1 shows a brief comparison to other interconnection wrappers found in literature. ARM processor is shown as an example of a processing node.

Table 1: Implementation comparison

IP, Ref	Tech	Size	Gates / mm^2	MHz
ARM 7TDMI-S, [29]	0.18	32b	- / 0.62	80-115
HIBI v2 (A)	0.18	8b	4k / 0.05	200-300
HIBI v2 (D)	0.18	32b	20k / 0.25	200-300
Noc Cell, [18]	-	32b	10k / -	-
Noc Cell, [19]	0.12	32b	- / 0.26	166/500
Noc Cell, [21]	0.35	16b	15k / 0.80	110-267
Proteo, [17]	0.18	8b	16k / 0.20	-
SPIN, [20]	0.13	32b	- / 0.24	200

B. Performance evaluation

Fair comparison of interconncetions is difficult since there are no widely accepted benchmarks or reported implementations simply do not offer enough information. The performance of HIBI v.2 protocol was tested with a test case similar to the one described by Saastamoinen *et al.* [17] (pp. 193-213) that provides at least a starting point. The application reads 10x10 byte pictures from source memory, processes the data, and writes 10x10 byte result pictures to target memory. Our test case model assumes

minimum latency of 10 cycles for processing and five cycles for memory reads. Exactly the same application model is used to test all HIBI networks. The architecture includes eight processing units, single-ported source and target memories, and either single- or dual-bus 8-bit HIBI v.1 or v.2. Round-robin was used as the arbitration method. The maximum transfer size was varied by changing the maximum allowed bus reservation time. Single- and dual-bus networks are similar to seg A and seg B in Figure 3, respectively.

The asymptotic upper bound for execution time suggests that bus and ring should produce roughly equal results. One processed picture needs at least one read request, picture reading and writing, and acknowledge from the target memory. The smaller the transfer sizes, the more requests and acknowledges are required. HIBI v.2 has a slightly lower performance than HIBI v.1 with small transfer sizes due addressing latency but, on the other hand, it has a smaller implementation. The performance of HIBI v.2 increases with buffer sizes because a smaller amount of retransfers are required. Figure 9 shows two conceptual timings for a two-bus system.

Doubling the bus width or adding another bus naturally improves the performance but the improvement is less than two-fold because requests and acknowledgements cannot utilize the increased bandwidth. Particularly adding a second bus does not necessarily improve the performance if the task pipelining is inefficient. Our simple application model does not perform any handshaking and therefore all agents start requesting source data at the same time. With a small transfer size, memory sends several slices of the picture to all requesting processors before any of them has a full picture. Therefore, processing is delayed and the same happens when agents write their results. Consequently, one of the buses is idle for a long time and execution time does not improve much from the single bus topology. This situation is illustrated in Figure 9b and with a dashed line in Figure 10.

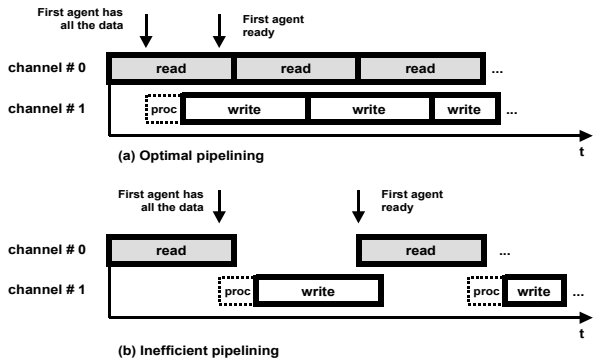

Figure 9. Case study pipelining with two-bus network

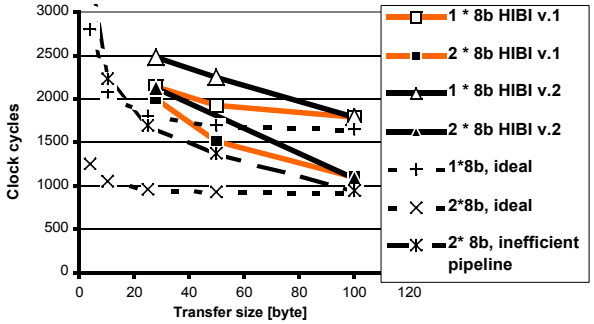

Figure 10. Execution times for 8 pictures, 8-bit HIBI networks

Figure 10 shows the measured run time of the application in clock cycles. With large transfer sizes, there is no difference in performance between HIBI v.1 and v.2. Both dual-bus HIBI networks have 15-35% smaller execution times for subsequent pictures when the time for filling the pipeline has only a minor effect. Figure 11 illustrates the estimated area for different interconnection architectures. The area of HIBI v.2 is smaller than HIBI v.1 with all transfer sizes. As noted earlier, the latency of functional unit determines the required buffer space instead of the packet size. As the packet size gets bigger, the effect of initial latency decreases and less buffer space is needed. In contrast, storing all packets inside the interconnection logic would result in area consumption to be linearly dependent on packet size. Estimated area for buffering the whole packet is shown with a dashed line in Figure, which clearly indicates why this is not a preferable method.

Comparison with results of Saastamoinen *et al.* [17] (about 2500 cycles, 4.4 mm^2 with 28B/tx) shows that even a single HIBI bus provides better performance than a Proteo ring network with considerably smaller area cost. A second bus to HIBI network improves performance nearly 40% and the area is still smaller than in Proteo.

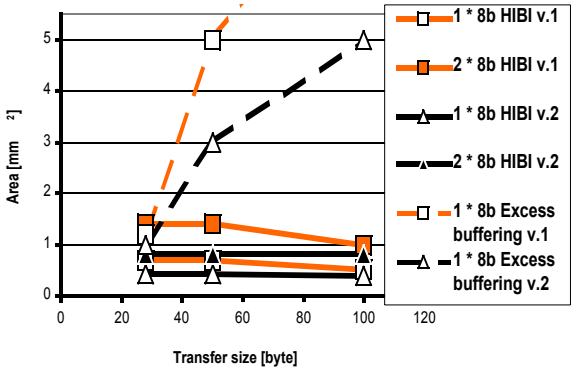

Figure 11. Effect of buffering scheme on interconncetion area, ten 8-bit wrappers

VI. CONCLUSIONS

In this paper, with our HIBI scheme, we have shown that a number of useful operations, such as architecture orthogonalization, QoS, and heterogeneous structures, can be implemented based on a non-traditional bus interconnection. Especially, the proposed arbitration and configuration mechanisms offer great potential for both execution time and energy optimization. HIBI can be regarded as NoC because it scales upwards, theoretically to arbitrary topologies, but particularly to hierarchical and fat bus structures. The data transfer capacity can, therefore, be chosen relatively freely while maintaining well known and easily programmable architectures. Furthermore, the presented synthesis results show that HIBI implementations have performance indicators and implementation areas that are acceptable for contemporary SoCs.

References

[1] H. Chang et al., *Surviving SoC Revolution*, Norwell, MA: Kluwer Academic Publishers, 1999.

[2] M. Keating, P. Bricaud, *Reuse Methodology Manual*, 2nd ed., Norwell, MA: Kluwer Academic Publishers, 1999.

[3] K. Keutzer et al., "System-Level Design: Orthogolization of Concerns and Platform-Based Design," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, issue 12, pp. 1523-1543, 2000.

[4] J.H. Park *et al.*, "MPEG-4 video codec on an ARM and AMBA," in *Proc. Workshop and Exhibition on MPEG-4*, 2001, pp. 95-98.

[5] S. Dutta *et al.*, "Viper: A Multiprocessor SoC for Advanced Set-Top Box and digital TV Systems," IEEE Design and Test of Computers, vol. 18, issue 5, Sept-Oct 2001, pp. 21-31.

[6] J.A. Rowson, A. Sangiovanni-Vincentelli, "Interface-Based Design," in *Proc*. *DAC*, 1997, pp. 178-183.

[7] D. Lyonnard et al., "Automatic Generation of Application-Specific Architectures for Heterogeneous Multiprocessor System-on-Chip," in *Proc*. *DAC*, 2001, pp. 518-523.

[8] VSI Alliance, "Virtual Component Interface Specification 2," Version 1.0, 1999.

[9] OCP-IP Alliance, "Open Core Protocol Specification," Release 1.0, Portland, OR, 2001.

[10] I.D. Scherson and A.S. Youssef, *Interconenction Networks for High-Performance Parallel Computers*, Los Alamitos, CA: IEEE Computer Society Press, 1994.

[11] A. Varma, C.S. Raghavendra, eds., *Interconenction Networks for Multiprocessors and Multicomputers Theory and Practice*, Los Alamitos, CA: IEEE Computer Society Press, 1994.

[12] J. Zalewski, ed., *Advanced Multirprocessor Bus Architectures*, Los Alamitos, CA: IEEE Computer Society Press , 1995.

[13] ARM Limited, "AMBA Specification Rev 2.0," 1999.

[14] Sonics Inc., Sonics uNetworks Technical Overview, Revision A21-1, 2000.

[15] E. Salminen et al., "Overview of Bus-based System-On-Chip Interconnections," in *Proc*. *ISCAS*, 2002, pp. II-372 - II-375.

[16] L. Benini, G. de Micheli, "Networks on chips: A New SoC Paradigm", *Computer*, vol. 35, issue 1, pp. 70-78, 2002.

[17] A. Jantsch, H. Tenhunen, eds., *Networks on Chip*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2003.

[18] F. Moraes et al., "A Low Area Overhead Packet-Switched Network on Chip: Architecture and Prototyping," in *Proc*. *IFIP VLSI-SOC*, 2003.

[19] E. Rijpkema *et al*, "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Network on Chip (Extended version)," *IEEE Proc. Computers and Digital Technique*, Spetember 2003.

[20] A. Andriahatenenaina and A. Greiner, "Micro-network for SoC: Implementation of 32-port SPIN Network," in *Proc*. *DATE*, 2003, pp. 11128-11129.

[21] T. Valtonen et al., "An Autonomous Error-tolerant Cell for scalable Network-on-Chip Architectures," in *Proc*. *Norchip*, 2001.

[22] C.A. Zeferino et. al, "A Study on Communication Issues for Sytem-on-Chip," in *Proc*. *SBCCI*, 2002, pp. 121-126.

[23] S. Rixner et al., "Register Organization for Media Processing," in *Proc*. *HPCA-6*, 2000, pp. 375-386.

[24] K. Kuusilinna et al., "Low-Latency Interconnection for IP-Block Based Multimedia Chips," in *Proc*. *PDCN*, 1998, pp. 411-416.

[25] V. Lahtinen et al., "Interconnection scheme for continuous-media systems-on-chip," *Microprocessors and Microsystems,* vol. 26, issue 3, pp. 123-138, 2002.

[26] T. Kangas et al., "System-on-Chip Communication Optimization with Bus Monitoring," in *Proc*. *DDECS*, 2002, pp. 304-309.

[27] L. Benini, G. de Micheli, *Dynamic Power Management Design Techniques and CAD Tools*, Norwell, MA: Kluwer Academic Publishers, 1998.

[28] J. Rabaey, M. Pedram, eds., *Low Power Design Methodologies*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1996.

[29] ARM Limited, "ARM7 Thumb Family Flyer," 2003.