Architectural and Physical Design Optimizations for Efficient Intra-tile Communication

A. Papanikolaou[1], F. Starzer[2], M. Miranda[1], K. De Bosschere[3], F. Catthoor[1,4]
[1]IMEC v.z.w., Kapeldreef 75, 3001 Leuven, Belgium
[2]FH Hagenberg, Hauptstr. 117, 4232 Hagenberg, Austria
[3]Universiteit Gent, Sint-Pietersnieuwstr. 25, 9000 Gent, Belgium
[4]Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Heverlee, Belgium

Abstract—**Intra-tile communication requirements for future SoC platforms are becoming ever more demanding for new processor and memory architectures. Increased bandwidth, low latency and low energy consumption are required, which the current communication architecture solutions cannot provide. In this paper we propose the use of software-controlled, light-weight segmented buses to implement the communication between the processing elements and their working memories. We show that significant energy and delay/latency gains can be expected from the use of this communication architecture.**

I. Introduction

Energy minimization is fast becoming the major optimization criterion during the design of embedded systems, in order to maximize their battery lifetime. The System-on-Chip (SoC) architecture comprising synchronous tiles or islands is an architectural template for low-energy and high-performance application domain specific system instantiations.

The communication architectures in the SoC architectural template can be divided into inter-tile and intra-tile architectures. Inter-tile architectures take care of the communication between the tiles, which is not very frequent if the application mapping is done properly and where a significant latency can be tolerated. Intra-tile communication architectures provide the means for transferring data between the components of the same tile, the local memories and processing elements for instance.

The requirements on the intra-tile communication network are more severe than the inter-tile ones. Inside the tile, the communication bandwidth is large, for the wireless and multimedia application domain our measurements indicate a bandwidth of a few tens of Gbps is needed. Additionally, latency should not exceed one, or maximally two, cycles and the energy per transfer should be very low, given the large bandwidth. Currently, past solutions are being re-used for intra-tile communication, such as point-to-point connections and crossbars, in the case of processing architectures with a centralized local memory [1].

For global energy efficiency reasons, however, new system architectures are moving toward fully software-controlled solutions. Distributed local memory organizations consisting of software-controlled scratch-pad memories instead of caches are used. New processor architectures are also moving toward software-controlled VLIW architectures, which offer the right amount of programmability and energy efficiency to fill the gap between ASICs and general purpose processors for various application domains. The TI C54x [2], for instance, has 4 local data memories, 5 local program memories and 3 load/store units. Shared buses, based internally on crossbars, are used for the communication in this architecture. They consume, however, too much energy per bit and are not scalable in number of connected components or technology node, see [3]. Furthermore, the number of memories and load/store units will further increase to fill the massively parallel datapaths of the future in order to exploit the application level parallelism.

Thus, a more scalable and energy-efficient programmable communication architecture is required to meet the demands of such platforms. It should be software-controlled and not run-time hardware based as current advocated solutions. The software control should move the exploration penalty fully to design time so that at run-time energy and delay overhead becomes negligible. Furthermore, it should be programmable in order to be flexible enough to handle several applications, but not designed under worst-case connectivity assumptions to improve energy efficiency.

On the other hand, interconnect wire technology scaling trends further deteriorate the energy efficiency of communication architectures, because the scaled wires cannot keep up with the improved performance and energy characteristics of the scaled transistors.

In this paper we describe an application domain specific, software-controlled communication network for intra-tile communication that is based on a light-weight implementation of multiple segmented buses. This architecture is scalable to future technology nodes and to more complicated systems. We also illustrate a number of optimizations that are required at the architectural and the physical design level in order to achieve energy-efficient intra-tile communication.

II. Related Work

Most of the past and current research on communication architectures has been focused on the communication between the SoC tiles. Emerging industrial standards, such as the AMBA bus [4], CoreConnect [5], STBus [6], WISHBONE [7] as well as a number of academic contributions, like NoCs [8], [9] and self-timed segmented buses [10], all target this type of communication. The architecture proposed in this paper uses a much finer-grain segmentation and much simpler control than the one proposed in [10].

In the context of intra-tile communication, however, the amount of literature is limited. As already mentioned, current industrial System-on-Chip implementations rely on textbook [11] solutions such as point-to-point connections [1], shared buses [2] and crossbars [12]. These solutions, however, are general purpose communication architectures, which cannot provide both the energy-efficiency, the programmability and the scalability that will be required by future massively parallel processing architectures [3]. Each of the existing solutions can satisfy only one of these major requirements.

Segmented buses are not a novel communication mechanism as such. They were initially developed in the context of supercomputing, in order to speed up computations on parallel architectures in the mid 90's, see [13] for instance. Chen et al [14] have illustrated the potential to use them also for communication energy optimization. They did not, however, show how such an architecture can be programmed or controlled. Their switch implementation is based on single pass transistors and it incurs a very high delay penalty, due to the high transistor resistance especially for future technology nodes. Furthermore, all the existing work on segmented buses focuses solely on architectural level optimizations, ignoring the physical aspects of the buses. In contrast, in this paper we show on a case study how to perform combined architectural and physical design optimizations during the design trajectory of segmented buses.

III. Intra-tile segmented buses

Communication between processing elements and their local working memories involves a very large bandwidth. The TI C54x architecture [2] can have a peak bandwidth of 9.6Gbps, while the Adres architecture [15] can go up to maximally 128Gbps for a clock frequency of 500MHz. Furthermore, the latency should be minimal in order to reduce stall cycles.

To meet the requirements for low energy, low latency and high bandwidth we propose a software-controlled, light-weight implementation of segmented buses. It is based on bi-directional tri-state buses, where each bus is divided into segments by introducing switches at the places where the various components are connected to it, see Figure 1. Each switch is configured at run-time to form a path between the source and the sink of the transfer. No handshaking techniques or transfer protocols are used, thus the control is light-weight. Conflicts are eliminated by providing enough parallelism and scheduling the transfers through different paths at compilation time. The number of parallel buses and the connections between components and buses are decided based on the application domain bandwidth requirements. Thus, the communication architecture can vary from a single segmented bus to a (partial) segmented crossbar. Depending on the target use of the system, this communication architecture can be fully customized to a single application or it can be flexible and programmable enough to handle the communication required by programmable processing elements in application domain specific platforms.

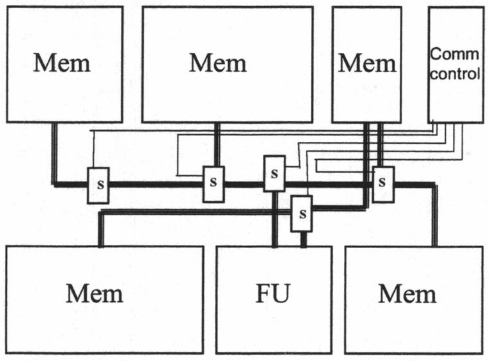

Fig. 1. The architecture of the segmented buses communication network includes a number of parallel shared communication resources, a number of switches and a mechanism that controls the switches.

Structurally, any communication architecture comprises two major parts, namely a data plane and a control plane. The data plane is the infrastructure that provides the means for the transfer of data from the source to the sink of the communication. The control plane provides the correct routing of the data. In the proposed architecture, the buses and the switches form the data plane, while the network controller and the control wires to the switches are the instantiation of the control plane.

A. Data plane

Two major issues exist in the definition of the data plane of the proposed communication architecture. The first is the number of parallel buses and the connections to the system components. The second is the implementation of the switches and where to insert them in the architecture.

We have made some assumptions about the methodology that is used to map the application residing on the embedded system. The methodology presented in [16] is used to define the number of parallel communication resources needed to satisfy the application bandwidth requirements. Based on the high-level application mapping the peak bandwidth per cycle is extracted and sufficient parallel buses are allocated. The connections of components to buses are also defined based on the synthesis of the memory organization. The major assumption in this methodology is that the application is fully characterizable at compile-time. That enables us to completely alleviate conflict situations by allocating enough parallel connections.

The implementation of the switches is based on active components instead of passive components, such as pass transistors. The reason is that passive components tend to have a very large resistance when not over-sized, leading to a large communication delay overhead. This trend becomes even worse as technology scales down. Furthermore, the active switches can also act as repeaters to drive the signal through the long interconnect wires. The switches are 3-port uni-cast or multi-cast components implemented using tri-state buffer chains, see Figure 2. Each triangle in the figure is a tri-state buffer chain that can be sized according to the corresponding wire-load and the latency requirements. The use of tri-state buffers is enabled by the fact that very long wires do not exist after segmentation and that the bus conflicts do not occur due to the design-time scheduling. This simplifies the testing, removing difficulties encountered for tri-state circuit implementations [17]. A small decoding logic is added to minimize the number of external control wires. Its area is negligible compared to the size and the number of the buffers required, 96 buffer chains are needed for a 32 bit switch while the decoding logic comprises 7 logic gates and can be shared by all the buffers.

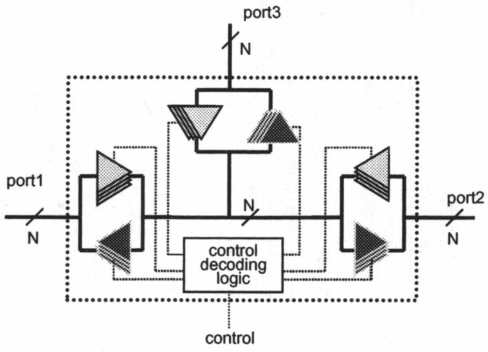

Fig. 2. The switches consist of a number of tri-state buffers that are required to buffer the output wire loads and a small control logic that decodes the control bits coming from the network controller and in turn controls the activation of the buffers.

B. Control plane

The control plane of the communication network consists of the controller and the control wires to the switches. As already mentioned, the communication conflicts are resolved at compile and synthesis time. As a result no time is needed for handshaking and run-time arbitration, so the latency of the transfer can be limited to just one cycle. The network controller only has to configure the switches to the correct setting.

The controller can be implemented in two ways, either as a dedicated hardwired controller with a memory-mapped look-up table (LUT) or via the instruction memory of the programmable components of the system in a fully software-controlled manner. In the case of an application specific hardwired controller, the address of the transferred data element in the virtual address space is decoded to identify the target component during a bus transfer. Based on the target component the controller can retrieve the control bits for the switches from the LUT. The contents of the LUT can be changed by the synthesis tool to provide some reconfigurability. In the case of a programmable platform for a target application domain and compile-time analyzable applications the compiler itself can generate the required control bits for the switches, which would be inserted in the application code as separate network configuration instructions.

We will illustrate the main concepts on an application specific design of a Digital Audio Broadcast (DAB) receiver system. Thus the memory access and transfer schedule is defined a priori, so in the illustration itself we use the hardwired network controller implementation where the LUT is initialized once.

IV. System Architecture Case Study

This target system implementation, the DAB receiver, can be regarded as a single tile in the context of Systems-on-Chip. It is a standalone tile and the only interaction with other tiles are the signal coming from the antenna (analog front-end and digital baseband processing) and the signal going to the audio output.

The system architecture of this receiver consists of three hardwired datapaths and their 9 working memories. These datapaths implement the Fourier transform, unscrambling in the time and frequency domains and error correction functionalities. The complete system architecture and connectivity is shown in Figure 3.

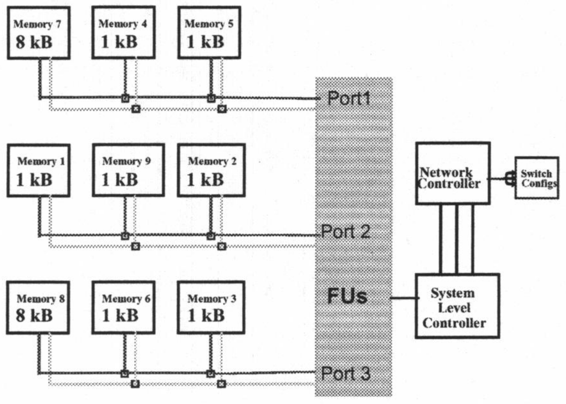

Fig. 3. The final architecture of the Digital Audio Broadcast receiver. The Functional Units are centralized and the network controller is shown. The communication architecture is clustered into three sets of segmented buses. Each switch is controlled by the network controller (connections not shown).

V. Architectural Optimizations

Two architectural optimizations have been performed in order to optimize the communication energy consumption. The first one involves the partitioning of the entire system into non-communicating clusters, so that the communication architecture can be implemented using smaller and more efficient connections. The second optimization targets the control plane energy consumption. Both are quite generally applicable and they will be described next. They will also be illustrated on our DAB case study.

A. System partitioning

One of the reasons for poor scalability of existing communication architectures is their centralized structure. Shared buses connect all the system components on a given communication resource, which spans the entire system to provide the required connectivity. One way to tackle this issue is to divide the system into clusters of components that do not communicate between them and provide local communication solutions. Even in case where communication between such clusters is required, hierarchical communication structures can provide a good alternative solution.

As already shown in Figure 3, the three datapaths are implemented in a centralized manner and they share internal registers and three ports to the memory organization. The accesses to the memories via each individual port are routed such that no memory needs to have access to more than one datapath port. This is done by analyzing the access schedule and determining which memories do not need to be accessed simultaneously. Memories that have concurrent accesses should belong to different clusters. This optimization is straightforward for application specific designs, for programmable platforms it implies additional constraints to the instruction scheduler. In the case of the DAB the memories of the system can be grouped into three clusters, where each cluster is only connected to one datapath port and no communication between clusters exists.

To connect this architecture we have built a multiple bus structure. The data and address buses that connect the memories to the datapaths are partitioned into three smaller buses, which connect only part of the architecture and can be implemented as multi-terminal net point-to-point connections since no conflicts can occur, see Figure 3. These nets connect each pin of all the communicating blocks via a single shared wire. This can bring a very significant reduction in the communication energy consumption, since the connections do not need to be routed over the entire die anymore.

A further optimization involves segmenting these connections by adding switches, we still call this new communication structure "segmented buses", even though it diverges from the original concept of [14] and [13]. The number of transfers over them is very large and segmentation significantly reduces their total switching capacitance and energy consumption.

B. Communication architecture instantiation optimization

At the communication architecture level the switches were clustered into groups in order to re-use as much as possible the control wires. A very strong correlation exists between the configurations of the neighboring data and address bus switches. The information on the address bus always travels from the datapath to the memories, while the information on the data bus can travel both ways. As a result, the address switches can be stripped down to provide only the required functionality. The control wires can be shared between adjacent address and data switches. Three control bits can configure a pair of these switches, by modifying the decoding logic inside the address switches. This optimization provides a maximum reduction in the number of required control wires and their energy consumption.

The switches used in this design are uni-cast, because the application does not require multi-casting capabilities from the bus. Thus each data bus switch has seven states, routing data from any input to any output port, and one idle state. As a result, three control bits are enough per pair of switches.

Component	Area
memories	71%
datapaths	18%
system level controller	7%
network controller	1%
switches	3%

TABLE I
AREA BREAKDOWN OF THE DAB RECEIVER USING THE PROPOSED SEGMENTED BUS ARCHITECTURE FOR COMMUNICATION. THE AREA OVERHEAD OF THE SEGMENTED BUSES IS SMALL.

For the DAB, an initial fully segmented bus architecture with three buses and connections between them to provide full system connectivity included 22 switches, thus 66 control wires. Reusing these wires among the address and data switches further reduces their number to 33.

VI. PHYSICAL DESIGN OPTIMIZATIONS

Segmenting the buses results in each individual memory having a different communication energy and delay associated to it. In order to minimize the overall communication energy consumption, the memories should be connected to the bus in an optimal order. The most active memories should be connected very close to the functional units, so that the part of the bus that is very often activated becomes minimal, see also [18]. The less active memories can tolerate longer connections.

Furthermore, this ordering should be respected during the floorplanning and placement stages. Since the memory blocks are placed manually on the floorplan, their ordering can be done in a good manner following the architectural ordering decision.

VII. OVERALL RESULTS ON DAB

In order to evaluate the energy consumption, area occupation and delay of the communication architecture, we pushed the design of the DAB application through the physical design stage. The area of the functional units is estimated via logic synthesis and the characteristics of the memories come from a 130nm memory library of an industrial partner. The characteristics of the switches were calculated based on the synthesis results for the decoding logic and analytical calculations for the buffers [19], we used ideal buffering of the interconnect wires. One iteration is required during floorplanning and global routing, because the wire-lengths impact the switch area. All the technology parameters come from the ITRS roadmap for the 130nm technology node.

The breakdown of the area occupation of the various components is illustrated in Table I. The portion of area dedicated to the communication architecture is around 4% including all the control overhead, which is negligible.

Table II contains the data transfer and storage energy consumption for the decoding of a DAB audio frame for the memory organization and the three different communication architectures, the energy of the datapaths is not included. The arbitration energy for the shared bus architecture has been neglected. The segmented bus architecture shows promising energy savings compared to the other two architectures.

The delay of the critical path in our segmented bus communication architecture was about 1.8 nsec, while the critical memory access delay in the design was about 2.5 nsec, given that the memory sizes in the design are very small. This confirms that the transfers over the bus can be performed in a single cycle, given the lack of run-time arbitration, and they can be pipelined with the memory access time.

Comm arch	Memory energy	Comm energy	Total energy	Comm energy savings	Total energy savings
single shared bus	1.16e-4	2.44e-4	3.6e-4	-	-
multi terminal P2P	1.16e-4	5.8e-5	1.74e-4	76%	52%
segmented buses	1.16e-4	2.391e-5 (data) 8.104e-6 (ctrl)	1.48e-4	87%	60%

TABLE II
BREAKDOWN OF DATA TRANSFER AND STORAGE ENERGY CONSUMPTION FOR PROCESSING ONE DAB AUDIO FRAME AT 130NM. TOTAL SAVINGS REFERS TO THE ENERGY FOR STORAGE AND TRANSFER OF DATA. THE MEASUREMENT UNIT IS JOULES.

VIII. CONCLUSIONS

Communication is becoming a significant contributor of system energy consumption. We propose the use of a light-weight, software-controlled implementation of segmented buses in order to provide the required bandwidth and latency for intra-tile communication of future SoC platforms. Energy gains exceeding 80% compared to using a shared bus and gains of around 40% compared to shared point-to-point connections have been demonstrated.

REFERENCES

[1] S. Dutta, R. Jensen, A. Rieckmann, "Viper: a multiprocessor SoC for advanced set-top box and digital TV systems", IEEE Design & Test of Computers, Sep. 2001.
[2] "TMS320VC5471 fixed-point digital signal processor: data manual", http://focus.ti.com/docs/prod/folders/print/tms320vc5471.html .
[3] A. Gangwar, M. Balakrishnan, P.R. Panda, A. Kumar, "Evaluation of bus based interconnect mechanisms in clustered VLIW architectures", Design Automation and Test in Europe, March 2005.
[4] ARM AMBA bus specification http://www.arm.com/armwww.ns4/html/AMBA?OpenDocument
[5] IBM CoreConnect bus architecture http://www-03.ibm.com/chips/products/coreconnect/
[6] STBus specifications http://www.stmcu.com/inchtml-pages-STBus_intro.html
[7] WISHBONE specifications http://www.opencores.org/browse.cgi/filter/category_soc
[8] W. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks", Design Automation Conf, June 2001.
[9] L.Benini, G. De Micheli, "Networks on chips: a new SoC paradigm", IEEE Computer, Jan. 2002.
[10] J. Plosila, T. Seceleanu, P. Liljeberg, "Implementation of a self-timed segmented bus", IEEE Design & Test of Computers, Nov. 2003.
[11] J. Duato, S. Yalamanchili, N. Lionel "Interconnection networks, an engineering approach", IEEE Computer Society, June 1997.
[12] B. Khailany, W.J. Dally, U.J. Kapasi, P.R. Mattson, J. Namkoong, J.D. Owens, B. Towles, A. Chang, S. Rixner "Imagine: media processing with streams", IEEE Micro, March 2001.
[13] Y. Li, S.Q. Zheng, "Prefix computation using a segmented bus", South-eastern Symposium on System Theory, April 1996.
[14] J.Y. Chen, W.B. Jone, J.S. Wang, H.-I. Lu, T.F. Chen, " Segmented bus design for low-power systems", IEEE Trans. on VLSI, Mar. 1999
[15] B. Mei, A. Lambrechts, J.-Y. Mignolet, D. Verkest, R. Lauwereins, "Architecture exploration for a reconfigurable architecture template", IEEE Design & Test of Computers, April 2005.
[16] T. van Meeuwen, A. Vandecappelle, A. van Zelst, F. Catthoor, D. Verkest, "System-level interconnect architecture exploration for custom memory organisations", Intl. Symposium on System Synthesis, Oct. 2001.
[17] T. Kishi, M. Ohta, T. Taniguchi, H. Kadota, "A new inter-core Built-In-Self-Test circuits for tri-state buffers in the System on a Chip", Asian Test Symposium, 2001.
[18] H.Wang, A. Papanikolaou, M. Miranda, F. Catthoor, "A global bus power optimization methodology for physical design of memory dominated systems by coupling bus segmentation and activity driven block placement", Asian South-Pacific Design Automation Conference, Jan. 2004.
[19] J.Rabaey, "Digital Integrated Circuits: A Design Perspective (2nd edition)", Prentice Hall, Englewood Cliffs NJ, 2003.