

# Energy Costs of Transporting Switch Control Bits for a Segmented Bus

Kris Heyrman, Antonis Papanikolaou, Francky Catthoor, Peter Veelaert, Wilfried Philips

*Keywords*—System-on-Chip, deep sub-micron technology, bus interconnection, segmented bus, power-aware optimization.

*Abstract*— When designing interconnection circuitry for Systems-on-Chip in deep sub-micron technology, it becomes clear [15] that the energy consumed by communication within a processing tile is gradually taking predominance over the energy consumed in the circuitry of the tile itself. Long wires are prevalent in most bus systems; therefore the segmented bus, which continuously minimizes the length of active wire, and therefore the capacitance that has to be driven, becomes attractive [20]. The segments of the bus are switched in and out of place by multiple switches, which themselves need to be controlled from the instruction subsystem of an instruction set processor.

Whatever energy is gained in the data plane could well be lost in the control plane, unless measures are taken to minimize this. The control energy for a segmented bus has two main components: there is the energy required to distribute the control bits over the tile, and the energy required to fetch the control bits from the instruction memory. In this paper we concentrate on the first part: the energy required to distribute the switch control bits. We compare this energy with the energy consumed by the segmented bus itself, and also with the energy that would be consumed by the bus, were it not segmented.

The switch control bits for a System-on-Chip need to be determined at design time by a suitable compiler. In the absence of such a tool, we have developed a method to determine the pattern of switch control bits and calculate the cost of driving the switches. We use a bus access schedule that is already power-optimized, and also a physical description of the bus structure and the modules attached. We do this for four alternative choices for the design of a digital broadcast receiver. The results indicate that the cost of driving the switches of the segmented bus do not offset the gain obtained from the segmentation, and that the prospects of driving the switches from the instruction processing subsystem are indeed promising, as long as the second part of the energy budget (fetching the control bits from the instruction memory) can be well optimized.

## I. INTRODUCTION

When designing intra-tile communication networks for Systems-on-Chips (SoCs) in the deep sub-micron area, there are serious advantages to the concept of segmented buses.

The scaling down of technology nodes, which is going on today, implies that ultimately power lost to switch the capacitance of long lines will outweigh power lost in active circuits. Thus, it becomes clear that the technique of electrically isolating those segments of a bus that are not in use, cycle-by-cycle, can deliver important savings in the data plane power budget.

If we consider intra-tile SoC bus connection systems, other than segmented buses, the obvious alternatives are: the tri-state shared bus, multiple crossbar systems, and double multiplexer systems. None of them provide the opportunity to avoid power being wasted on unused sections, and some require more segments to be switched over in voltage than the linear structure we find in a shared or segmented bus. To be sure, crossbars and multiplexers have inherently higher bandwidth than a single segmented bus. This advantage can be matched by multiple segmented buses. The extra buses are used only when required. See Figure 2.

The economy in power, of course, must be paid for in the control plane: a segmented bus system needs an additional control

Kris Heyrman is with Hogeschool Gent, Schoonmeersstraat 52, B-9000 Gent, Belgium. E-mail: kris.heyрман@hogent.be

Antonis Papanikolaou is with IMEC, Kapeldreef 75, B-3001 Leuven, Belgium. E-mail: papaniko@imec.be

Francky Catthoor is with IMEC. E-mail: francky.catthoor@imec.be

Peter Veelaert is with Hogeschool Gent. E-mail: peter.veelaert@hogent.be

Wilfried Philips is with TELIN, Ghent University, St-Pietersnieuwstraat 41, B-9000 Gent, Belgium. E-mail: wilfried.philips@ugent.be

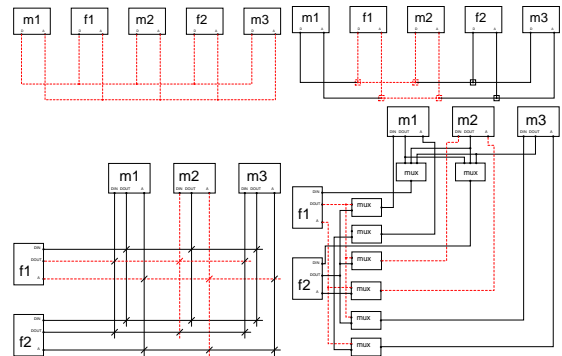


Fig. 1. Four alternatives: tri-state shared bus (top left), segmented bus (top right), a multiple crossbar (bottom left), a double multiplexer (bottom right). For e.g. a transfer from functional unit f1 to memory module m2, the dashed segments are active and must be driven.

structure, and we should ensure that the extra energy cost of this structure does not waste the gain in the data plane.

If the System-on-Chip has a programmed instruction set architecture, it seems evident that control over the switches that drive the segmentation must at some stage be generated by the program compiler. In the absence of such a compiler, we want to maximize our knowledge of the process of generating and distributing the control bits that drive the segmenting switches. More specifically, this paper aims to answer the following questions:

- Can a program that is not a compiler, but which has knowledge of the data transfer schedule that will be employed by the compiler and other system synthesizing tools, like IMEC's Atomium suite and the Memory Architect (MA)<sup>1</sup>, be used to predict the flow of switch control bits and provide us with a better knowledge of its properties?
- Is the power cost of distributing the control bits from some central generator small or large in comparison to the gain obtained from segmenting the bus?

In section II we clarify some terms. In section III we provide further background to segmented buses, followed by an overview of related work in section IV and a general discussion of the intra-tile segmented bus in section V. Section VI describes our tool, *segmented bus analysis*. Section VII contains our experiments on an application driver. In section VIII we offer our conclusions, including the answers to the questions above.

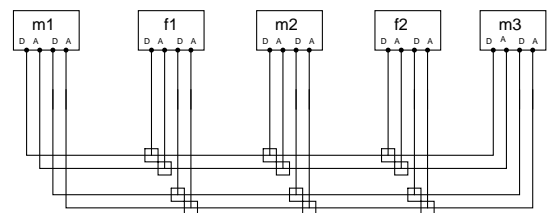


Fig. 2. Double segmented bus.

<sup>1</sup> Atomium/MA is a storage optimization and memory allocation program, part of the DTSE (Data transfer and storage exploration) tools suite.

## II. TERMINOLOGY

In the past, a “shared bus” involved the use of tri-state interconnection technology for transceiving on the bus. As a matter of fact, in the deep submicron (DSM) domain tri-state transceivers are uncommon. In industry and today’s SoC standards [1][5][16][21], the term “shared bus” has come to mean either “double multiplexer” or “crossbar”.

The term “segmented bus” is sometimes used in literature to refer to multiple buses interconnected by inter-bus bridges. Many bus standards define different buses for processor-local (in our terms, “intra-tile”) connectivity and for connection of on-chip peripherals. Our “segmented bus” is different, taking segmentation to its logical consequence: not only are intra-tile and inter-tile buses decoupled, but every individual segment of the intra-tile bus can be decoupled for reasons of saving power.

## III. BACKGROUND

Energy minimization is fast becoming the major optimization criterion in the design of embedded systems, in order to maximize battery lifetime. The SoC architecture, with synchronous tiles or islands, is a template for high-performance, low-energy systems that are specific to an application domain.

Communication architectures in the SoC template can be divided into inter-tile and intra-tile. Inter-tile architectures take care of the communication between some tiles, which is less frequent and can tolerate significant latency. Intra-tile communication architectures provide the means for transferring data between components of the same tile, for instance between local memories and processing units.

Requirements for intra-tile communication are more stringent than those for inter-tile communication. Inside the tile, communication bandwidth is large. For the wireless and multimedia application domain, measurements indicate that a bandwidth of some tens of Gigabytes per second is needed. Additionally, latency should not exceed one, at most two, cycles and energy per transfer should be very low, given the large bandwidth.

For reasons of global energy efficiency, new 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 also move to software-controlled VLIW architectures. The TI C54x [17], for instance, has 4 local data memories, 5 local program memories and 3 load/store units. In this architecture, shared buses, which internally are built from crossbars, are used for communication. However, they consume too much energy per bit and are not scalable with respect to the number of connected components or the technology node, see [9]. Furthermore, the number of memories and load/store units will increase to fill the massively parallel datapaths of the future, in order to exploit application level parallelism.

Thus, a scalable and energy-efficient communication architecture is required to meet the demands of such platforms.

## IV. RELATED WORK

Most past and current research on communication architectures has been focused on inter-tile communication. Emerging industrial standards, such as the AMBA bus [1], CoreConnect [5], STBus [16], WISHBONE [21] as well as a number of academic contributions, like NoCs [6], [3] and self-timed segmented buses [13], all target inter-tile communication. The architecture proposed in this paper is intended for intra-tile communication and uses finer-grain segmentation and simpler control.

In the area of intra-tile communication, the amount of literature is limited. As mentioned, current industrial System-

on-Chip implementations rely on textbook [7] solutions such as point-to-point connections [18], shared buses [17] and crossbars [10]. These are general purpose communication architectures, that cannot provide both the energy-efficiency and the scalability required by massively parallel processing [9].

Segmented buses are no 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, cfr, for instance, Li [11]. Chen et al [4] have illustrated their potential for communication energy optimization. They did not, however, show how such an architecture can be programmed or controlled. Their switch implementation is based on pass-transistor logic, not complementary CMOS logic, and incurs high delay penalty, due to transistor resistance, especially in the context of future technology nodes.

## V. THE INTRA-TILE SEGMENTED BUS

Communication between processing elements and their local working memories involves large bandwidth. Furthermore, in order to reduce stall cycles, latency must be minimal.

To meet 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 the bus. See Figure 3. 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. The number of parallel buses and the connections between components and buses is decided based on the application domain bandwidth requirements. Conflicts are eliminated by providing enough parallelism and scheduling the transfers through different paths at compilation time. Depending on the targeted use of the system, this communication architecture can be fully customized to a single application or it can be flexible enough to handle all communication required by programmable processing elements in application domain specific platforms.

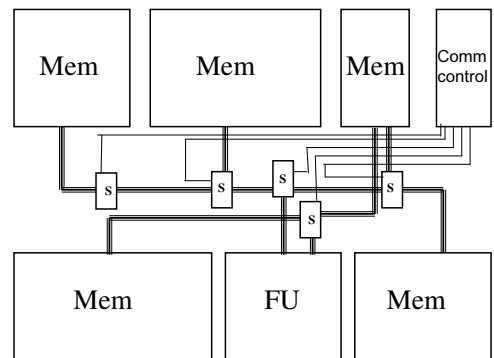


Fig. 3. The architecture of the segmented buses communication network includes a number of parallel shared communication resources, a number of switches and a mechanism that controls the switches.

Any communication architecture comprises two major parts: 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 correct routing of the data. In the proposed architecture, buses and the switches form the data plane, while the network controller and the control wires to the switches are part of the control plane.

### A. Data plane

The major design issue is the definition of the number of parallel buses and the connections to the system components.

The methodology presented in [19] is used to define the number of parallel communication resources needed to satisfy the application bandwidth requirements. Based on high-level application mapping, the peak bandwidth is extracted and a sufficient number of parallel buses is allocated. The connections of components to buses are also defined based on the synthesis of the memory organization.

We propose an implementation of the switches which is based on active, not passive, components, such as pass transistors. Passive components tend to have a large resistance when not over-sized, leading to an excessive communication delay overhead. This trend becomes worse as technology scales down. In contrast, active switches can act as repeaters to drive a signal through long interconnect wires. Switches are 3-port uni-cast or multi-cast components implemented using tri-state buffers, see Figure 4. Use of tri-state buffers is possible because very long wires do not exist after segmentation and bus conflicts do not occur due to design-time scheduling. A small decoding logic is included to minimize the number of external control wires. Its size is negligible compared to the size and the number of the buffers required.

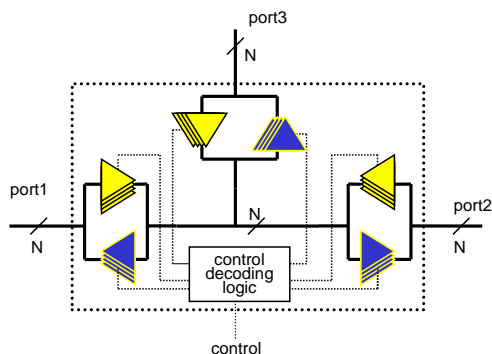


Fig. 4. The switches consist of a tri-state buffers required to buffer the output wire loads and control logic that decodes control bits coming from the network controller and in turn activates the buffers.

### B. Control plane

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

The controller can be implemented in one of 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 a hardwired controller, the address of the 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 look-up table. In the case of a programmable platform, for any target application domain and compile-time analyzable applications the compiler itself generates the required control bits for the switches, which are inserted in the application code as separate network configuration instructions.

## VI. SEGMENTED BUS ANALYSIS

We made a program called *sba* (*segmented bus analysis*) to conduct experiments on segmented buses. The approach is to use data obtained from an Atomium/MA run (together with the layout information), recover the execution schedule for each basic block, recover the memory assignment, reconstruct the access tree, and collapse it into a per-cycle node tree of concurrent accesses. Then, for each path through the bus geometry, we decide what the switch positions must be. Walking through the cycle node tree, we revisit every node, resolve each transfer to a set of control bits to be emitted, and from the dynamic behaviour of the control bits calculate the energy required to in transport the information, taking into account the physical lengths involved.

## VII. APPLICATION DRIVER: DIGITAL AUDIO BROADCAST RECEIVER

The Direct Audio Broadcast application has been used before as a driver application in the optimization of bus power [20].

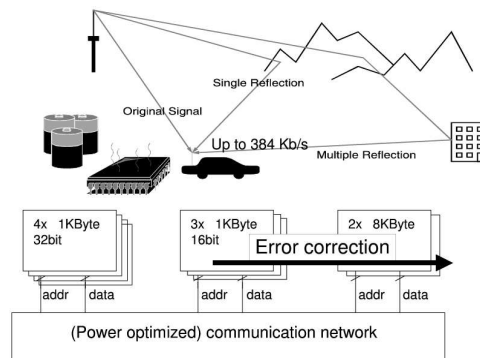


Fig. 5. The DAB receiver application.

In order to estimate the required segmented bus switch control power for different solutions in the layout of a data-intensive application, we have made 4 different physical layouts of such a receiver. The layouts were made according to the practice of power-aware floorplanning [20]. Each represents a different tradeoff between power efficiency and circuit area, resulting in a different memory chip count and consequently, different complexity of the segmented bus structure.

We will first discuss the parameters of this particular design. Then we will describe the segmented bus analysis, and finally offer some observations that the experiment allows us to make.

### A. Design Parameters

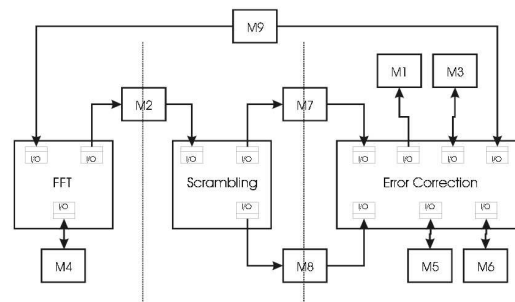


Fig. 6. The DAB receiver architecture.

**Functional Units.** The DAB receiver has three functional units: a FFT subsystem (f1 in the diagrams), a Viterbi decoder or

error correction processor (**f2**), and a deinterleaver or scrambling processor (**f3**). Most of the code of the application can be partitioned to belong to either of these three, but an appreciable part (42 out of 213 cycle node types) cannot, since some of the routines are shared by all processors. Rather than attribute them at will to any of the 3 functional units, we left these cycles out of the results of the simulation. This is an imperfection of the method that does not allow for multiprocessing well.

**Choice of Storage Bandwidth Optimization (SBO) Operating Point and Memory Assignment and Allocation (MAA).** After Data Storage and Bandwidth Exploration (DTSE) analysis of the problem, 4 different sets of SBO and MAA optimizations were chosen to set 4 alternative tasks for the layout process. The solutions are 4 floorplans that all feature 3 buses but have an unequal number of memories: respectively 4, 8, 10 and 12. They all have an 8-bit bus (**b1**), which in some solutions is extended to include some 16-bit segments, and two 32-bit buses (**b2** and **b3**). See Figure 7.

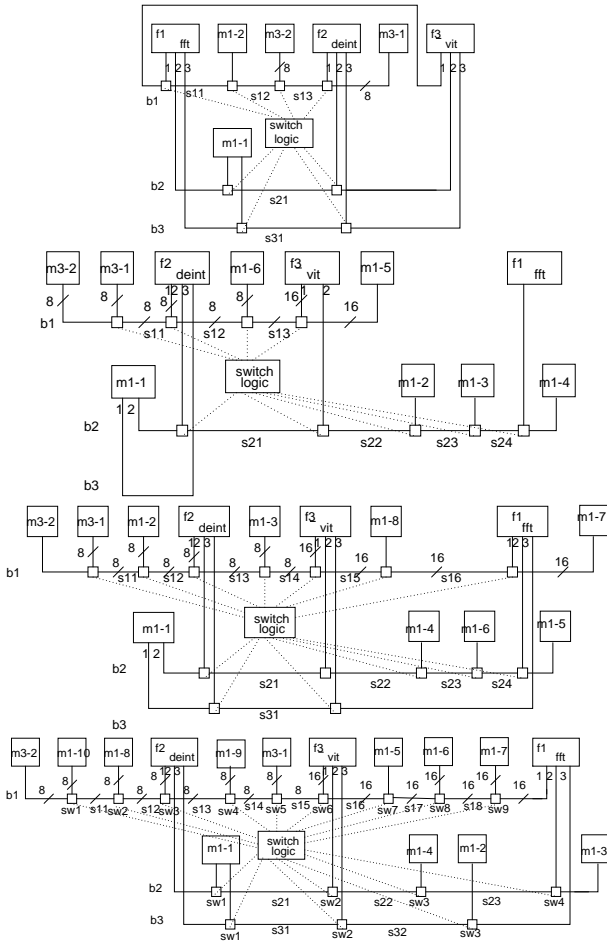


Fig. 7. Four possible solutions from design space exploration: three buses with resp. 4, 8, 10 and 12 memories; each with 3 functional units.

**Memory Architecture.** There are two memory partitions (i.e. sets of memories of different types): **m1** and **m3**. The optimization of putting more arrays in different and smaller memory modules (called **m1-1**, etc.) has been pursued to a different extent.

**Physical Layouts.** The activity-aware placement procedure (see also Figure 8) minimizes the length of segments with high activity. Figure 9 shows the four resulting layouts.

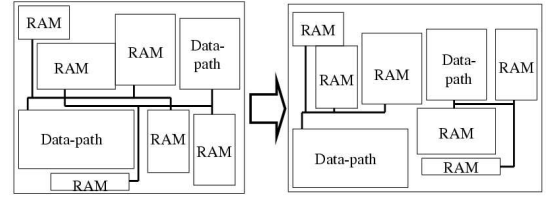


Fig. 8. Activity-aware placement.



Fig. 9. Four possible solutions for the layout: 4, 8, 10 and 12 memories.

**Segment Lengths and Control Bit Line Lengths.** For each of the chosen layouts, the lengths of segments, attachments and bit control lines were calculated. The bit control lines were assumed to originate from some central module, presumably a control bit loop buffer.

### B. Analysis

**Switch Control Patterns.** A typical activity pattern is like this one from the 12-memory layout, where FFT -processor **f1** is active and using buses **b1** and **b2**:

b1								
sw1	sw2	sw3	sw4	sw5	sw6	sw7	sw8	sw9
1100	1100	1100	1100	1100	1100	1100	1100	1000
1100	1100	1100	1100	1100	1100	1100	1100	1100
1100	1100	1100	1100	1100	1100	1100	1100	1000 1010
1100	1100	1100	1100	1100	1100	1100	1100	1100 1000
1100	1100	1100	1100	1100	1100	1100	1100	1100 1100

b2				b3		
sw1	sw2	sw3	sw4	sw1	sw2	sw3
1100	1100	1100	1100	1100	1100	1100
1100	1100	1001	0111	1100	1100	1100
1100	1100	1100	1100	1100	1100	1100
1100	1100	1100	1100	1100	1100	1100
1100	1100	1001	0111	1100	1100	1100
1100	1100	1100	1100	1100	1100	1100

Looking at the bottom part of Figure 7, we can see that only switches **b1:sw8**, **b1:sw9**, **b2:sw3** and **b2:sw4** are active, switching the right-hand side of buses **b1** and **b2**. The period is 3 cycles, in this case.

What we do find is that if some two switches change data direction, all switches in between also change data direction and show activity. The effect is counteracted by the fact that because of physical power-aware layout, very active connections will be short and the number of intermediate switches will be few.

**Energies.** Using the switch control patterns and the segment and control bit line lengths, we can calculate the energies. In Figure 10 the energies consumed on the buses by the DAB application are compared with the energies that would be consumed were the bus not segmented, for all four design choices. In the first place, the comparison corroborates the advantage of a segmented bus from the standpoint of power efficiency.

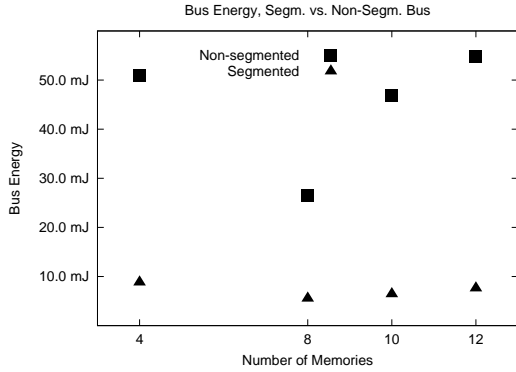


Fig. 10. Comparison of segmented vs. non-segmented energy for four DAB receiver design choices

If we compare the energy consumed in the data plane for a bus that is not segmented with the energy consumed by a segmented bus, we see that both reach a minimum which is not radically different between the solutions. This would indicate that segmentation does not impose different optimization targets for physical layout than a non-segmented solution.

In Figure 11 we compare the energy component required to transport switch control bits with the energy consumed in the data plane, for all four design choices.

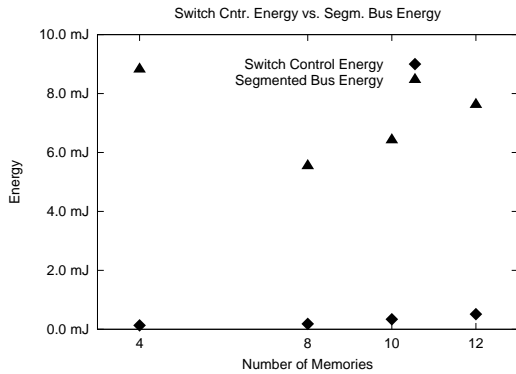


Fig. 11. Comparison of switch control energy vs. segmented bus energy for four DAB receiver design choices

If we compare the energy component required to transport switch control bits with the energy consumed in the data plane, we find that it is of a lower order of magnitude. Intuitively, this can be attributed to a good choice of the switch codes, which reduces the number of active wires; there are many more active data and address lines. Also there is only limited activity on some buses in some branches of the cycle node tree, thanks to activity-aware placement.

### C. Observations

Some observations that can be made from the case of this digital audio receiver, are:

- Power consumed in the segmented bus data plane seems to be 17-21% of the power consumed in the equivalent unsegmented bus data plane.

- The optimum layout is not different for the segmented bus than for an unsegmented one. It needs to be confirmed whether or not this is a coincidence.
- Power consumed in the segmented bus control plane, as far as the transport of the control bits is concerned, is much smaller than the gain obtained from segmenting in the first place. It is in the range of 1.5-6% of the power consumed in the data plane, when segmenting.<sup>2</sup>
- Clustering: Clustering may make sense both locally and per bus. Often the switches do not change state because the bus is not in use. At other times, there are frequent patterns on a section of a bus because only short sections are being used.
- How often does switching occur: often, for long periods, once a cycle. This follows from what SBO considers to be a cycle: a period through which accesses to external memory are scheduled. Consecutive cycles during which only internal registers are accessed, are not counted. Only if the access schedule would be completely the same for two cycles, w.r.t. sources and sinks as well as data direction, there would not be any switching activity. (Or else when the bus is simply not used.)

In table I,  $r_A$  is the ratio of energies consumed, during address and data transfers in the data plane: i.e. the ratio of the energy on a shared bus vs. the energy on a segmented bus.  $r_B$  is the ratio of the switch control energy, compared to the energy consumed during address and data transfers in the data plane, on a segmented bus.  $r_E$  is the ratio of error cycle types to all cycle types. These refer to the cycles that were left out of the calculation because they cannot be attributed to any single processor.

	DAB, 4 mem.	DAB, 8 mem.	DAB, 10 mem.	DAB, 12 mem.
Unsegm./segm. energy ratio $r_A$	17 %	21 %	14 %	14 %
Switch cntrl/segm. energy ratio $r_B$	1.5 %	3.3 %	5 %	7 %
Ratio of error cycle types $r_E$	25 %	31 %	26 %	26 %

TABLE I  
EXPERIMENTS INVOLVING COMPARISON OF FLOOR PLANS.

The data from table I are captured graphically in Figure 12.

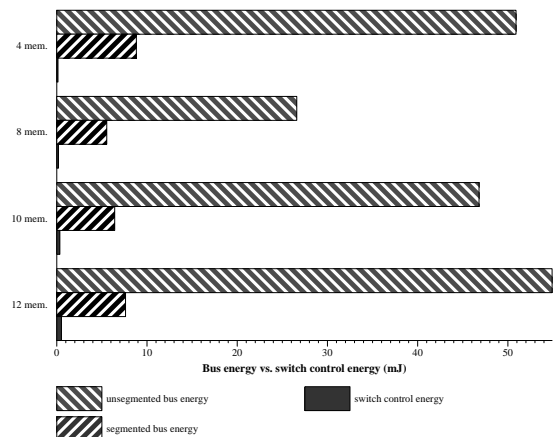


Fig. 12. Energy comparisons for DAB receiver

<sup>2</sup>If there is inaccuracy here it would come most from the denominator of the comparison: the power consumed in the data plane is dependent on the content of the data and addresses, which can only be guessed at in (in an educated way) if compilation nor full simulation is done.

## VIII. CONCLUSIONS

### A. The Answers to the Original Questions

First we return to the questions that we stated in section I.

- *Can a program that is not a compiler, but which has knowledge of the data transfer schedule that will be employed by the compiler, be used to predict the flow of switch control bits and provide us with a better knowledge of its properties? Answer: It can predict, but only for simple, "benchmark-type" problems.* The exact flow of control bits for a complex application can only be predicted by the compiler. The block-based execution schema that Atomium/MA produces, for example, does not have enough information to recreate fully the intended flow over all cycles of the application.

We can however, by judicious use of cycle-true walking, the "don't-cycle" mode of `sba` and randomized cycle-walking, obtain predictions about the control bit flow and the energy associated with it in the control plane. Thus, we can 'see' the switch control bits in operation over long portions of the control flow. Another reason why we are limited to test shorter programs is precisely the fact that we are doing cycle-true simulation, where hundreds of interpreted instructions are used to simulate a 10 ns cycle: execution times become prohibitive for long programs. Even with these limitations, `sba` has allowed us to glance at the bus control bits we had not seen before.

- *Is the power cost of distributing the control bits from some central generator comparable to the gain obtained from segmenting the bus? Answer: Depending on the opportunity for segmenting that the application offers, it can be from 1 to 10 % of the data energy on the segmented bus, which itself is typically 20 - 50 % of the data energy of the non-segmented case, and thus remains appreciably smaller than the gain obtained.* This means that, if we look at it from the viewpoint of control energy, the case for the segmented bus still stands, that we should not go for optimization of the transport component, but instead look for the ways of optimizing the second component, i.e. the energy cost of the fetching of control information.

*Until we have the complete optimized design for the control plane, we think we should not worry about the energy cost for switch control bit transportation.*

- *What will be the likely energy cost of fetching the control information for the switches from the instruction memory? Answer: This still needs to be determined, but the possibility exists that it is very low, if the switching can be controlled at run-time, even if it is scheduled at design-time.*

### B. Summary

We have proposed a segmented bus architecture for SoC designs in the deep sub-micron domain, and compared the energy required to distribute the switch control bits with the energy consumed by the segmented bus itself, and also with the energy that would be consumed by the bus, were it not segmented.

Using four alternatives in the design of a DAB receiver, carried out with a schedule optimized for low power consumption and with power-aware placement rules, we have found that the energy costs of driving the switches for a segmented bus are appreciably lower than the gain obtained from using a segmented bus in the first place.

## REFERENCES

- [1] "ARM AMBA bus specification Rev. 2.0", <http://www.arm.com/?OpenDocument>
- [2] T. Anderson and S. Agarwala, "Effective hardware-based two-way loop cache for high performance low power processors", International Conference on Computer Design, 2000.
- [3] L. Benini et al., "Networks on chips: a new SoC paradigm", IEEE Computer, Jan 2002.
- [4] Chen, J.Y. et al., "Segmented bus design for low-power systems", TVLSI, Mar 1999.
- [5] "IBM CoreConnect bus architecture", <http://www-03.ibm.com/chips/products/coreconnect/>
- [6] W. Dally et al., "Route packets, not wires: on-chip interconnection networks", DAC, 2001.
- [7] J. Duato et al., "Interconnection networks, an engineering approach", IEEE Computer Society, Jun 1997.
- [8] J. A. Fischer, P. Faraboschi, and C. Young, "Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools", Elsevier, Amsterdam, 2005.
- [9] A. Gangwar et al, "Evaluation of bus based interconnect mechanisms in clustered VLIW architectures", DATE, 2005.
- [10] B. Khailany et al, "Imagine: media processing with streams", IEEE Micro, Mar 2001.
- [11] Y. Li et al, "Prefix computation using a segmented bus", Southeastern Symposium on System Theory, Apr 1996.
- [12] B. Mei et al, "Architecture exploration for a reconfigurable architecture template", IEEE Design & Test of Computers, Apr 2005.
- [13] J. Plosila et al., "Implementation of a self-timed segmented bus", IEEE Design & Test, Nov 2003.
- [14] J. Rabaey, "Digital Integrated Circuits: A Design Perspective (2nd edition)", Prentice Hall, Englewood Cliffs NJ, 2003.
- [15] "International technology roadmap for semiconductors", 2001 edition
- [16] "STBus specifications", [http://www.stmcu.com/inchtml-pages-STBus\\_intro.html](http://www.stmcu.com/inchtml-pages-STBus_intro.html)
- [17] "TMS320VC5471 fixed-point digital signal processor data manual", <http://focus.ti.com/docs/prod/folders/print/tms320vc5471.html>
- [18] S. Dutta et al., "Viper: a multiprocessor SoC for advanced set-top box and digital TV systems", IEEE Design & Test, Sep 2001.
- [19] T. Van Meeuwen et al., "System-level interconnect architecture exploration for custom memory organisations", ISSS, 2001.
- [20] H. Wang et al., "A global bus power optimization methodology for physical design of memory dominated systems by coupling bus segmentation and activity driven block placement", ASPDAC, 2004.
- [21] "WISHBONE SoC Interconnection Architecture for Portable IP Cores - Rev B.3, Sep 7, 2002", [www.opencores.org](http://www.opencores.org)