COMB: COmbined Memory and Bus partitioning for SoC architectures.

Abstract

There has been a continued proliferation in the demand for application specific System on Chip Cores in the recent years. Meeting the power budget constraint continues to be a major challenge for the designers architecting such systems. In this work, we demonstrate that simultaneous partitioning of the bus and memory subsystem into smaller segments can be an effective mechanism for reducing the energy consumption of a SoC. We present a genetic algorithm based search mechanism to determine a system configuration that is energy-efficient and validate the effectiveness of the configuration using a cycle-accurate virtual platform for a multiprocessor SoC. Our results using various applications shows that the proposed approach gives significant energy savings and accentuates the benefits of previously proposed bus and memory partitioning schemes applied individually or in combination.

1. Introduction

There is a continued demand for small form-factor devices that incorporate an increasing number functionality on a single chip. Consequently, current system-on-chips (SoC) integrate millions of transistors and multiple cores on the same chip. Further, they use high clock speeds to meet the increasing demands of the applications. These trends have imposed a severe challenge for SoC system designers in meeting the power budget permitted for the design. Higher power consumption is detrimental in multiple ways including increased package and cooling costs and shorter life-time between battery charges.

Architectural level design decisions can have a significant influence on the power consumption characteristics of a SoC. Due to the need to support interaction between the multiple cores, the interconnect structure is a vital component in the SoC design. Further, current SoC structures support from tens to hundreds of different memory components. Consequently, power-efficient design of the memory and interconnect systems can potentially offer significant power reduction.

This paper considers the exploration of the bus and memory configuration for reducing the power consumption. We propose a new genetic algorithm based search technique for simultaneously partitioning the bus and on-chip memories for generating a power-efficient configuration based on the application characteristics. Our approach is primarily targeted towards *application-specific* SoCs [LB04, GCF00, DG04], such as those developed in wireless LAN baseband processing, xDSL modems, low-level pixel processing for HDTV, etc.

We have validated the effectiveness of the configurations generated by our search algorithm through a cycle-accurate virtual platform [ML04] of a multiprocessor SoC platform using AMBA AHB fabric that provides performance and energy results. Our results reveal that the energy consumption can be reduced on an average by 20% as compared to a baseline SoC architecture based on a single shared bus. Further, we show energy reduction can be reduced by 12% and 18% on an average as compared to using only either memory or bus partitioning schemes. We also demonstrate that the proposed algorithm helps explore new design choices that are missed by just sequentially applying the bus and memory partitioning algorithms (in either order).

The rest of this paper is organized as follows. The next section provides an overview of related work on memory partitioning and bus partitioning. It also highlights the advantages of the proposed approach. The genetic algorithm for the simultaneous bus and memory partitioning is given in Section 3. Section 4 presents our implementation details and provides the experimental results. Conclusions are provided in Section 5.

2. Related Work and Background

2.1. Bus Partitioning

Many shared bus models like the AMBA [AMBA], CoreConnect [CORE], WishBone [WISH] have been explored for connecting the various modules in SoC designs. However, shared buses do not scale well when taking into account system design with an increasing number of cores. Bus partitioning is effectively aimed at countering such scalability issues by providing both speed and power gains. Performance is achieved by allowing simultaneous communication on different bus trunks, thus decreasing congestion and latencies on each trunk. Parallelism must be exploited as much as possible by keeping heavily intercommunicating cores clustered. Energy savings can be gained due to the smaller capacitive loads to be driven. The power reduction benefits of partitioned bus architecture depend on the communication profile of the system [LAH04, SS04].

Most previous work in the area of bus partitioning ([CTH00], [WBJ99], [YZ98], [SEC04], [CZ02]) is focused upon three-state buffer interposition, or variations of this concept. All of these approaches have two major disadvantages: they require full custom design methodologies, and they focus solely on reducing power consumed by global bus wires. It is well known, however, that a significant amount of the complexity and power budget in bus designs is spent in the decoding, arbitration and multiplexer logic, which is not affected by wire segmentation. The proposed approach will instead leverage standard IP components in order to improve congestion issues, logic complexity and power consumption on each bus trunk. Additionally, this paper will show a methodology capable of exploring a huge variety of alternative topologies and will validate the

results by means of a cycle-true simulation environment with accurate power models. Similar approaches to design space exploration have been reported [SH04, GV02], but they are either focusing on performance alone, or aimed at tuning parameters such as bus bitwidth and encoding, which have a much smaller impact than bus and memory partitioning.

This work performs bus partitioning based on the AMBA AHB and makes use of bridges to communicate between different bus trunks. Fig. 1 shows a typical AMBA AHB-AHB bridge, which serves as a unidirectional link between two buses. Each of the ends of the bridge also called the bus interface units (BUI) are connected to the master and the slave buses respectively and may be operated at different frequencies. Communication across the bridges incurs performance and power penalties. Every data packet communicated across the bridge accrues latency delays due to unavoidable synchronization and communication cycles. Such overheads can be significantly avoided by judicious partitioning of the system cores and slaves over multiple buses which is a goal of our proposed algorithm.

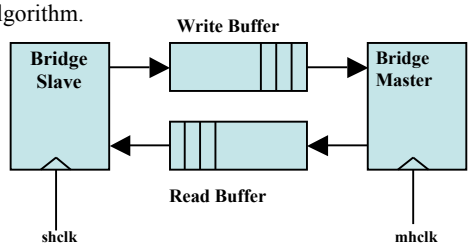

Fig. 1: AMBA AHB-AHB bridge

The bus segmentation problem comprises of solving a graph partitioning problem which, given the communication patterns (fig. 2) splits the architecture into smaller bus trunks satisfying the constraints imposed [SS04]. The main criterion behind such partitioning is to minimize the communication over the bridges, which are caused by to communication across buses. However, increasing the number of bus trunks increases the power and performance penalties, caused due to the hardware overhead in the form of additional bridges, which has to be taken care of while solving the partitioning problem.

2.2 Memory Partitioning

With the integration of tens of memory modules on a SoC, memory partitioning is an important step in reducing the architectural power consumption. The memory partitioning problem is defined as creating physical partitions of memory banks given the data access pattern defined by the application. Memory partitioning can provide both performance and power benefits. Since most applications do not tend to utilize their entire address space uniformly, memory partitioning algorithms attempt to partition the memory banks such that most accesses are confined a smaller size memory. Note that a smaller memory has a lower per-access energy cost. Figure 3 shows an example of such non-uniform access frequencies across the shared memory address space when running one of the benchmarks used in our evaluation. In this case, the small portion of memory that is frequently used can be divided into one bank and the remaining memory space that remains sparsely used maps to another bank. The key challenge for memory partitioning algorithms is to balance the benefits of reduced access energy costs against the penalty due to increased logic for bank address decoding. This tradeoff imposes limits on how small the memory partitions can be. Various approaches have been proposed to perform memory partitioning balancing these tradeoffs for reducing energy consumption [LB00, CAT98].

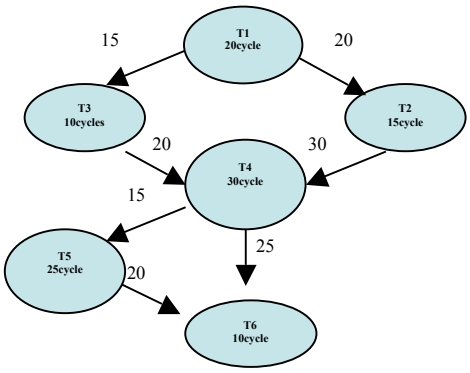

Fig. 2: Communication profile

2.3 Proposed Combined Approach

This paper aims at further strengthening the bus partitioning benefits by compounding them with simultaneous memory partitioning. Due to memory partitioning, the slave memory modules are broken into smaller chunks providing a finer grain control for assigning smaller memory chunks to different bus trunks at the bus partitioning stage. For example, the less frequently used sections of memory could be ideally partitioned and assigned to a separate bus trunk while the most accessed partition could be placed in the same trunk as the processor core which accesses it.

Exploring different bus and memory-partitioned structures simultaneously results in configurations that may have been discarded when applying the individual partitioning algorithms sequentially. For example, applying memory partitioning after bus partitioning does not provide the flexibility of assigning a new partition to a different bus trunk. However, our approach not only allows memory splitting but also simultaneously assigns it to a bus trunk in search of a power-efficient configuration. Similarly, the simultaneous partitioning approach is better than applying bus partitioning after memory partitioning. For example, increasing the number of memory partitions on a single bus also increases the control complexity because of the increased number of slaves. Consequently, the overheads may balance (or overwhelm) the benefits of accesses to smaller modules. However, if the partitions can be assigned to different bus trunks, some of these partitions may become effective solutions in saving

energy. These examples make the advantages of a combined approach clear, and the section on experimental results will quantify them.

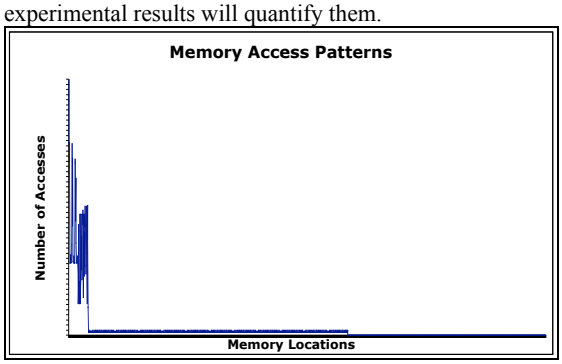

Fig. 3: Memory access patterns

3. A genetic algorithm for simultaneous partitioning

We propose a genetic algorithm (GA) [GA] search technique to explore the memory and bus partition possibilities to arrive at a power-efficient configuration. The GA takes as input the applications communication and memory behavior. A task graph (see Figure 2) that provides the duration of each task and the communication volume between the different tasks captures the communication information. In addition, the algorithm takes as input the access information of the memory elements at the granularity of banks of 128 contiguous bytes. The access information is obtained in the form of access frequency from each task in the task graph.

The first step in designing our GA involves finding a suitable representation for the chromosome to represent the bus and memory partitioning configuration employed. The proposed chromosome captures this configuration using an integer array *A[1…MEM+BANKS+PROCS],* where MEM is the number of memory elements to be allocated to a maximum of MAX banks and PROCS is the number of processors.
A chromosome depicts the following:

A[i] = t, for 0<=i < MEM, indicates that the i^{th} memory element is assigned to the memory bank t.
A[i] = x, for MEM<=i<MEM+BANKS+ PROCS, means that the i^{th} memory bank/processor is allocated to bus trunk x.

Note that we assume fixed topologies in the final proposed configuration by the genetic algorithm. In all the topologies we have any two bus trunks connected to each other by two bridges enabling two-way communication between them.

In order to avoid address-decoding complications, we prevent the assignment of two different banks that do not have a contiguous address space. Consequently, we also impose the following constraint:

$$A[i] \le A[j], \forall i < j, \forall i, j < MEM$$

Note that it may be possible to reassign the data locations or restructure the code appropriately to relax this constraint using a compiler. We expect that the power savings of the proposed partitioning scheme to be enhanced when such techniques are employed.

The GA initially generates a population of N chromosomes and continues to create new generations of this original population by modifying the chromosomes using two operators: crossover and mutation. The crossover operator produces two new offspring chromosomes by inheriting partial characteristics from two parent chromosomes. The cross over is performed over the first MEM elements of the chromosome and the rest of the chromosome separately. Figure 4 shows the basic crossover operation. The *Selection Operator* for choosing the chromosomes for mating or mutation is a random selection operator. However in case of crossover, to ensure that the resulting solution is feasible parents A and B are selected by the *selection operator* randomly but having the following characteristics:

(a) A[MEM/2] = B[MEM/2] – This ensures the contiguous partitions in the offspring. In cases where the middle element or the point of crossover has a memory element allocated to different bank, it will clearly generate the one resulting offspring having a non-contiguous memory allocation into the memory banks.
(b) Number of Bus trunks in A = Number of Bus trunks in B.

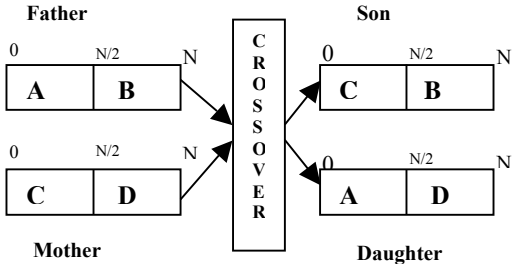

Fig. 4: Crossover operator

The mutation operator introduces new characteristics that may not already be present in the population of chromosomes. The mutation operator takes in a chromosome as an input and generates a new chromosome by randomly modifying the chromosome. However it does make sure that the newly generated solution is still a feasible solution by ensuring that the values in the array are in bounds and satisfy the primary conditions imposed for feasibility as discussed above.

A fitness function determines the quality of each chromosome (that represents a possible memory/bus partitioning configuration) based on the energy required for executing the given application. The evaluation requires the access (active and idle cycles) profiles of the different buses, bridges and memory components and the per access energy costs for each of these components.

First, we require a technique for providing a fast estimate of the number of communication cycles over

the each of the buses, namely the *intra-cluster* cycles and across the bridges, the *inter-cluster* cycles. The intra cluster cycles are simply computed based on the number of communication cycles within a bus trunk between the master and slaves attached to the bus. The inter-cluster cycles capture the latencies due to the crossing of the bridge and handshaking across the buses. Our model uses the numbers provided by AMBA Designware Datatbook [AMBA], for estimating the inter cluster latencies. We take a pessimistic approach in estimating the number of cycles, specifically the inter-cluster communication. The actual numbers of inter-cluster communication cycles are usually optimized in presence of split bus transactions, which is not captured in the estimation by the GA. Consequently, the number of cycles required for completing an application is typically over-estimated by our GA fitness function.

The per-cycle energy cost for the different components such as ARM core, caches, bus-attached memories and bridges is derived from STMicroelectronics foundry datasheets for a 0.13 μm process. The bridge energy cost was derived from an actual design and the switching capacitance was extracted for cycles in which the bridge was used and not used. Consequently, our energy model combines the number of cycles of activity in the bridge, the switching capacitance and the operating frequency, to derive the energy consumed in the bridge. In cycles, where there is no communication, there is still a small amount of power consumed in the bridges due to the switching of some gates in each of the bus interface units. The on-chip memory module power numbers are obtained directly from the datasheets for the different memory configurations. It also includes the leakage power in idle state. The bus power is modeled based on the numbers from [AB04], and scales proportional to the number of masters and slaves connected to the bus and their sizes.

The GA combines the estimated number of cycles and the per-cycle energy numbers to obtain the energy consumed by the configuration represented by a chromosome. The fitness value associated with a chromosome is therefore computed as:

Fitness = (1/Energy Consumed)

The fitness function rates a chromosome with higher energy consumption lower than the one that consumes lesser energy. The selection mechanism generates a new population from the current generation based on the fitness value assigned to each of the chromosome based on the survival of the fittest scheme [GA]. Chromosomes with a higher fitness value are more likely to survive while those with low fitness values are weeded out from the population.

4. Experimental Framework

The genetic algorithm was implemented in C language using the Genetic Algorithm Utility Library (GAUL)[GAUL]. The algorithm is operated with a crossover rate of 0.8 and a mutation rate of 0.2. The algorithm was let to run for 10000 generations. It is important to stress that the task graph input to the genetic algorithm was not created artificially (e.g. with a randomized task graph generator) or from abstract specification. Graphs and their annotation where created from functionally accurate execution traces of benchmark applications on single shared bus architecture. We obtained traces for a set of parallelized data-intensive benchmarks when executed on a shared bus system with 4 processor cores and 7 slaves simulated on the cycle-accurate virtual platform simulator. Four of the slaves act as the private memories, one as a shared memory, and the other two as semaphore and interrupt devices respectively.

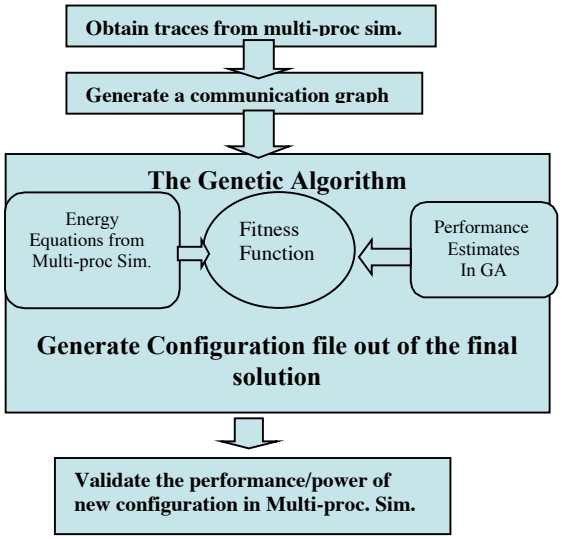

Fig. 5: Implementation Flow

The resultant topologies produced by the GA were used to configure the cycle-accurate simulator to execute the benchmarks and obtain the energy and performance numbers. While the per-cycle energy models used by the cycle-accurate virtual platform simulator are similar to that of the GA, the simulator accurately models the access behavior of the various components unlike the coarse grain estimate of the GA. The cycle estimate of the simulator itself has been validated in prior work in comparison to real designs. Consequently, we validate the energy benefits of the configurations determined by the GA using the simulator results. The implementation flow of our experimental methodology is illustrated in figure 5.

4.1 Enhancement of the Simulation platform

To validate the proposed scheme in this paper, we had to enable the instantiation of custom interconnection topologies in the existing virtual platform simulator based upon a configuration file given by the user as a boot-time input. The configuration file specifies the amount and placement of system components, thus allowing flexible design space exploration. The configuration file is in plain text format, and can therefore be easily generated by an external program. This feature is exploited in the present paper to link the

existing virtual platform developed in prior work to the genetic mapping algorithm, thus providing a fully automated way of validating the effectiveness of the configurations proposed by the GA. The configuration file also includes details about the memory mapping of slaves in the architecture. By means of this input, it is possible to choose how many physical slaves to use to implement a function. For example, a shared memory space can be implemented as a single memory bank, or by means of multiple devices contiguously mapped but attached to different buses.

4.2. Experimental Results

Table 1 shows the bus and memory system energy savings obtained by the new partitioned configuration determined by the GA as compared to the default shared bus architecture. The table provides energy savings determined by both our genetic algorithm estimation and the cycle-accurate virtual platform simulator. The percentage improvements determined by the GA and the cycle-accurate simulation results are different due to the pessimistic estimates of the GA as stated in Section 3. However, it is important to note that both the GA estimates and simulation numbers show similar trends. Note that there is an average improvement of 21% in bus and memory energy consumption in the new partitioned configuration found by our GA search algorithm as compared to the default shared bus configuration. For all our experiments we generated the configurations using a sample dataset and verified the effectiveness of the resulting solution for 30 randomly generated input sets.

Benchmark	GA estimate	Cycle-accurate simulation
DES	8.4	12.89
FFT	13.22	16.22
FILTER	11.28	14.89
LU	26.53	38.59
Image Smoother	11.04	18.4
PIL filter	8.16	14.51

Table. 1. Percentage reduction in bus and memory system energy using our GA proposed configuration as compared to using a shared bus configuration.

To show that the combined approach presented in this paper out performs the sequential approach or the individual bus partitioning and memory partitioning schemes we implemented each of the schemes using genetic algorithms. Table 2 provides the percentage improvement obtained by our approach over applying bus partitioning alone and memory partitioning alone. It is clear from these results that our approach accentuates the benefits of either approach. Finally, we compared the proposed approach to the sequential approach of implementing bus partitioning followed by memory partitioning and vice-versa. Due to the advantages of the simultaneous memory and bus partitioning, our approach provides 5% more energy savings on the average.

As an added advantage of our approach the total energy consumed by the processor cores goes down due to the reduction in the number of cycles required for the complete execution of any application. Figure 6 shows a plot of the percentage improvement in the energy consumed by the ARM cores attached to the bus using our proposed configurations as compared to single shared bus architecture. We observe an average 10% percent reduction in the energy consumed by the processor core energy.

Benchmark	BP only	MP only
DES	3.3	10.6
FFT	16.2	15.5
FILTER	28.2	14.4
LU	22.1	5.50
Image Smoother	19.64	3.31
PIL Filter	26.45	5.92

Table 2. Percentage reduction in total energy consumption of configuration determined by our approach as compared to configurations returned by Bus partitioning (BP) and Memory Partitioning (MP).

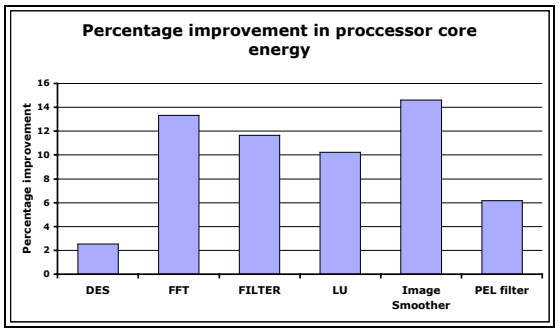

Fig. 6: Energy reduction in processor cores.

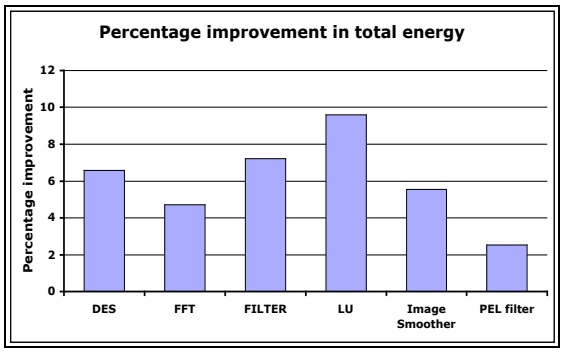

Fig. 7: Total system energy reduction.

The savings in the total energy consumption including the energy consumed by the caches is plotted in figure 7. We observe that our algorithm provides and average improvement of 7% over the single shared bus architecture.

Figure 8 shows the resultant configuration generated from our algorithm for the LU benchmark. The solution

demonstrates the effectiveness of our approach in splitting the memory in a way that complements the bus partitioning in an extremely constructive manner. The banks heavily accessed by processor 0 are placed in bus trunk 1 and the other banks are placed in bus trunk 0. The private memories are connected to the same bus trunks as their respective processors.

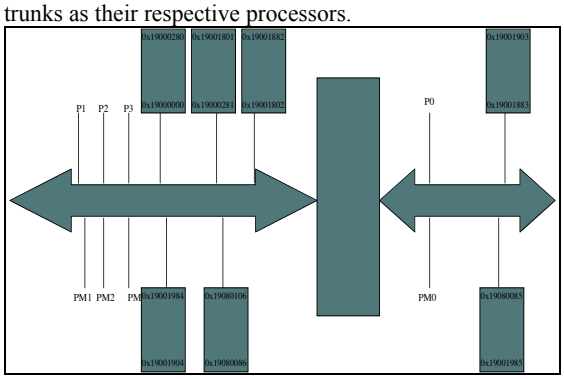

Fig. 8: Configuration provided by our algorithm for the LU benchmark. (PM denote the private memories)

5. Conclusion

This paper presents an approach to simultaneous bus and memory partitioning to reduce the energy consumption in an application specific SoC. This problem was formulated as a genetic search algorithm and the resulting partitioned configurations were validated to be effective using a cycle-accurate virtual platform simulator for a multiprocessor SoC. Our results reveal that there are significant energy savings to be gained by combining both memory and bus partitioning strategies. We also demonstrated that our approach is superior to sequential application of memory and bus partitioning algorithms. Our future work will consider the influence of voltage and frequency assignments to the different clusters in conjunction with the partitioning.

6. References

[LB04] L. Bisdounis, C. Dre, S. Blionas, D. Metafas, A. Tatsaki, F. Ieromnimon, E. Macii, P. Rouzet, R. Zafalon, L. Benini, Low-power system-on-chip architecture for wireless LANs*, Computers and Digital Techniques, IEE Proceedings, Volume: 151, Issue: 1 , 15 Jan. 2004, Pages:2 – 15*

[GCF00] Hung-Chi Fang; Chao-Tsung Huang, Yu-Wei Chang, Tu-Chih Wang, Po-Chih Tseng, Chung-Jr Lian, Liang-Gee Chen, 81MS/s JPEG2000 single-chip encoder with rate-distortion optimization, *Solid-State Circuits Conference, ISCC-2004.*

[DG04] D. Garrett, G. Woodward, L. Davis, G. Knagge, C. Nicol, A 28.8 Mb/s 4/spl times/4 MIMO 3G high-speed downlink packet access receiver with normalized least mean square equalization, *Solid-State Circuits Conference, ISSCC-2004.*

[LB00] L Benini, A Macii and M Poncino. A recursive algorithm for low-power memory partitioning. *In proceedings of the international symposium on Low power electronics and design,* 2000.

[YZ98] Yan Zhang, Wu Ye, M. J. Irwin. An Alternative Architecture for on-chip Global Interconnect: Segmented Bus Power Modeling. *3 2nd Asilomer Conference on Signals, Systems and Computers, 1998.*

[CAT98] F. Catthoor, S. Wuytack, E. De. Greef, F. Balasa, L. Nachtergaele, A. Vandecappelle, Custom memory management methodology exploration for memory optimization for embedded multimedia system design, *Kluwer,* 1998.

[CORE] CoreConnect Bus Architecture. http://www.chips.ibm.com /products/coreconnect.

[AMBA] AMBA Specification (rev2.0) and Multi layer AHB specification, Arm: http://www.arm.com, 2001.

[WISH] WishBone Specification. http://www.silicore.net/ wishbone.htm.

[WBJ99] W.B. Jone, J.S. Wang, H. Lu, I.P.HSU and J.Y.Chen. Segmented Bus design for low-power systems. *IEEE transactions on VLSI Systems,* March 1999.

[LAH04] K. Lahiri, A. Raghunathan and S. Dey. Design space exploration for optimizing on-chip communication architectures. *IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems,* June 2004.

[GV02] T. Givargis, F. Vahid, Platune: a tuning framework for system-on-a-chip platforms *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Nov. 2002.

[SH04] C. Shin, Y.T. Kim, E-Y. Chung, K-M. Choi, J.-T. Kong, S.-K, Eo, Fast exploration of parameterized bus architecture for communication-centric SoC design, *In proceedings of Design, Automation and Test in Europe 2004.*

[SS04] S. Suresh, Lin Li, N. Vijaykrishnan. Simultaneous partitioning and frequency assignment for on-chip bus architectures, *In proceedings of Design Automation and Test in Europe, 2005.*

[GAUL] GAUL: Genetic Algorithm Utility Library. http://gaul.sourceforge.net.

[AB04] Andrea Bona, Marco Caldari, Vittorio Zaccaria, Roberto Zafalon, High-Level Power Characterization of the AMBA Bus Interconnect, *SNUG, Boston 2004*

[CZ02] Chuanjun Zhang, Frank Vahid, A Power-Configurable Bus for Embedded Systems, *IEEE International Symposium on Circuits and Systems, Scottsdale (ISCAS), May 2002, pp. V-809-812.*

[SEC04] T. Seceleanu, Communication on a Segmented Bus Platform. In Proceedings of the IEEE International SOC Conference, Santa Clara CA USA, Sept. 2004. Pages 205-208.

[ML04] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, R. Zafalon, Analyzing On-Chip Communication in a MPSoC Environment, *Proceedings of the Design, Automation and Test ing Europe Conference and Exhibition 2004, Paris, France, Feb 16-20, 2004, pp. 752-757*

[CTH00] C-Ta Hsieh and M. Pedram. Architectural power optimization by bus splitting, *Design, Automation and Test in Europe,* 2000.

[GA] D.E Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, *New York: Addison Wesley, 1984.*