Power Aware Interface Synthesis for Bus-based SoC Designs

Nikolaos D. Liveris Prithviraj Banerjee
Electrical and Computer Engineering Department, Northwestern University, Evanston IL 60208
{nikos,banerjee}@ece.northwestern.edu

Abstract

In this paper we discuss the problem of interface synthesis for a system on a chip (SoC) such that the power consumption is minimized under some given latency constraints. Since the AMBA protocol has become one of the standard interfaces for SoC cores, we develop our interface synthesis methods around the AMBA protocol. We first provide an analysis of the parameters of the AMBA bus and the communication protocols and a bus power model that will be used by various transformations. Several latency improving and power minimizing transformations are presented at the bus level. Finally, a heuristic is presented which applies the above transformations in a certain order to provide minimum power under a given latency constraint. Experimental results are reported on two example benchmarks in that show that the heuristic is able to reduce power consumption on the wires by about 28% on the average from an initial design having a single layer bus architecture.

1 Introduction

In the past, the major concerns of system level designers were area, performance, and cost. Recently power consumption has become a very important issue in system-level design. On the average, a large portion of the system's power is consumed on the global on-chip busses [2]. A number of techniques [2-8] have been proposed to reduce the power consumed on the busses. Some of them focus on decreasing the switching activity of the wires of the bus and others concentrate on reducing the capacitance of the wires of the bus.

For reducing the switching activity on the wires of the bus, several encoding techniques have been proposed in the past [6], [7]. Moreover, several approaches manage to decrease switching activity by reducing the transfers on busses [4], [5]. This is achieved either by transforming the algorithm's specification or by architectural optimizations. One approach [3] for reducing the bus capacitance is to use a segmented bus instead of a single bus to decrease the capacitance of the wires.. Another approach [8] is to use a split shared-bus architecture compared to a monolithic shared-bus architecture. In addition, there are techniques [2] that take into account the capacitance of the data bus lines in conjunction with the encoding technique used for the data transfers.

In this paper we discuss the problem of interface synthesis for a system on a chip (SoC) such that the power consumption is minimized under some given latency constraints. Since the AMBA protocol [1] has become one of the standard interfaces for SoC cores, we develop our interface synthesis methods around this protocol. The AMBA specification defines three distinct busses; the Advanced High- performance Bus (AHB), the Advanced System Bus (ASB), and the Advanced Peripheral Bus (APB) [1]. In this work, we will use the AHB as a standard bus for the SoCs since the modules we will work with are high-performance system modules.

Section 2 describes the parameters of the bus and the communication protocols. Section 3 provides a bus latency constraint equality. Section 4 describes the bus power model that will be used to evaluate various bus level transformations. Several bus-level latency improving transformations are presented in Section 5, and several bus-level power improving transformations are described in Section 6. A heuristic is presented in Section 7 which applies the above transformations in a certain order to derive a minimum power solution under a given latency constraint. Experimental results are reported on two example benchmarks in Section 8 that show that the heuristic is able to reduce power consumption on the wires by about 28% on the average from an initial design having a single layer bus architecture.

2 Basic Parameters and Assumptions

2.1 Components

We assume that for each component of the system we know its type, its width and height for the specific technology used, and the number and types of its interfaces. The in-

1530-1591/04 $20.00 (c) 2004 IEEE

Symbol	Definition
λ	number of bus cycles to execute the operation in case a private bus is provided
δ	number of bus cycles after the invocation of the operation in which the operation must be completed
ρ	average rate of invocations per cycle for the operation
β	the number of beats of a burst (if the operation is a single transfer, $\beta = 1$)
B	the number of bits that contain useful data transferred during each beat
r	binary value that indicates whether the operation is a read (1) or write (0)
$btpi_{sig}$	the number of bit transitions on wire sig caused by one invocation of the operation

Table 1. Communication Operation Parameters

terfaces of each component work at the bus clock frequency, while the component may work at a different clock rate.

2.2 Communication Operations

All communication operations are initiated by the masters. Each communication operation (com-op from now on) is a read or write operation, which is requested by a specific master to be performed by a specific slave. The amount of data to be read or written is fixed for each com-op. Different com-ops may involve a different amount of data.

The main parameters of com-ops are summarized in Table 1. For the rest of this paper we assume that the com-ops do not include split, retry, lock transfers, and unspecified length burst transfers.

2.3 Interfaces

We assume that all interfaces are AHB compatible. The main parameters that characterize interfaces are summarized in Table 2. The quantities τ_i and $\lambda_{max\,i}$ characterize only master interfaces. More specifically, $\tau_{i\zeta}$ represents the toughest constraint of all operations of master *i* that will be executed via AHB layer ζ[1]. Added to that, $\lambda_{max\,i\zeta}$ is equal to $max\lambda_k$ of all com-ops invoked by *i* that have to be executed through layer ζ. For the rest of this paper we will assume that each master stalls until its requested operation is completed and that the slave interfaces are not capable to split bus transfers.

2.4 Bus Logic - Arbiter

For the arbiters a FIFO protocol is assumed because it can guarantee that no master will be led to starvation. Optimizations by changing the arbitration protocol are out of the scope of this paper.

Symbol	Definition
CR_{ij}	set of com-ops involving master i and slave j
cr_{ij}	binary value which is 0 if $CR_{ij} = \emptyset$ and 1 otherwise
$\tau_{i\zeta}$	$\min(\delta_k - \lambda_k)$ of all com-ops requested from *i* through ζ
$\lambda_{max\,i\zeta}$	$\max\lambda_k$ of all com-ops invoked by *i* through ζ

Table 2. Interface Parameters

2.5 AHB signals

The complete description of the AHB single-layer and multi-layer architecture can be found in [1].

The main parameters used for the characterization of an AHB signal are its number of lines, N, and the number of modules, γ, it connects to. In the scope of this paper we will assume that the N value for each signal is already decided and fixed.

3 Basic Latency Constraint Inequality

Since our heuristic for interface synthesis will search for a minimum power solution under latency constraints, it is important to develop a metric to constrain latencies on busses. The basic latency constraint inequality will be derived for the case of a single layer bus with M master interfaces and S slave interfaces connected to it. In order for this architecture to satisfy the latency constraints the following inequality must hold for each master *i*:

$$\tau_i > \sum_{m=1, m\neq i}^{M} \lambda_{max\,m} - (M-1) + 1 \qquad (1)$$

This inequality represents the worst-case scenario. All masters have already issued their worst in terms of latency com-ops. Since the arbiter implements a FIFO protocol, the requests of the other masters will be executed first. From the summation of the worst-case latencies $M-1$ cycles have to be subtracted. The reason is that the address and data transfer cycles are pipelined, so 1 cycle per transaction is saved. After the request from master *i* is issued, the arbiter will sample it at the beginning of the next cycle. Therefore, one more cycle is added at the right hand of the above inequality. If the above inequality holds for τ_i, it will hold for all com-ops requested by master *i* for the assumed architecture.

4 Bus power model

In this section the model that will be used for bus power estimation will be explained. The power consumed on the bus consists of two parts: (1) the power consumed internally in the bus components (arbiter, decoder, muxes), and (2) the

power consumed on the AHB wires that connect the master and the slave interfaces, and the bus components. The power consumed on each wire is given by: $P = 0.5V_{DD}^2 C f \alpha$ where V_{DD} is the voltage swing between the logic levels 1 and 0, C is the capacitance of the wire, f is the clock frequency, and α is the switching activity. In the next sections we will show how we estimate C and α for each wire.

4.1 Wire capacitance estimation

For estimating the wire capacitance we used the method described in [3]. In order to get an estimation of the length of each wire we use floorplanning algorithm found in [11] each time we want to evaluate a solution. The version, which was implemented as a C program, is the one described in Section 4.2 of that paper. The wirelength is approximated by the bounding box of the centers of the modules the wire is connected to.

4.2 Switching activity estimation

4.2.1 Data busses

Each layer of the AHB bus has two data busses; the HRDATA and the HWDATA. A com-op that will go through that layer can be a read or a write operation and will use only the corresponding bus. Since we assume that the data to be read or written are random, there is $1/2$ probability for each wire of the data bus in use to switch value. Therefore, for a data bus in use with N bit lines the beginning of com-op k will cause $N/2$ bit transitions on it. In case the k consists of more than one beats, subsequent beats will cause $B_k/2$ bit transitions. So, the bit transitions per invocation of a com-op k on the read data bus will be:

$$btpi^k_{HRDATA} = \frac{1}{2} r (N_{HRDATA} + (\beta_k - 1) B_k) \qquad (2)$$

where r is a binary value denoting whether k is a read or write operation. Similarly, the bit transitions per invocation of k for the write data bus can be derived:

$$btpi^k_{HWDATA} = \frac{1}{2} \bar{r} (N_{HWDATA} + (\beta_k - 1) B_k) \qquad (3)$$

4.2.2 Address busses

In the beginning of a com-op we assume that $N/2$ bit transitions of a N-bit address bus will occur. If the com-op is a burst then the bit transitions can be in average less for each of the sequential transfers. If the address space is split for minimum transitions on the address bus, each beat of the burst will produce approximately $1/2 \lg\beta$. However, we should take into account that a optimum split of the address space may not be feasible Therefore, the bit transitions of each sequential transfer of a burst will be more and we believe that $\lg\beta$ is a good approximation for this number. So, the number of bit transitions per invocation of a com-op k for the address bus is:

$$btpi^k_{HADDR} = \frac{1}{2} N_{HADDR} + (\beta_k - 1) \lg \beta_k \qquad (4)$$

4.2.3 Control signals

The switching activities of the majority of the control signals is low. The main reason for that is the small N value of these signals [1]. For the rest of the paper we will focus on the power consumed on the data and address busses. However, we will take into account the impact of the control signals on the placement of each module.

4.2.4 Estimated switching activity for each wire

The next step after computing the bit transitions per invocation of com-op k is to compute the bit transition per cycle, $btpc^k_{sig}$, caused by k for each wire sig. Since the ρ_k is the number of invocations per cycle for k then $btpc^k_{sig} = \rho_k btpi^k_{sig}$ Then the switching activity of each wire sig will be the bit transitions per cycle caused by all com-ops of the system that use sig. So,

$$\alpha_{sig} = \sum_{\forall k \in com-ops} btpc^k_{sig} = \sum_{\forall k \in com-ops} \rho_k btpi^k_{sig} \qquad (5)$$

The total power consumed on the wires of the bus will be then

$$P = \frac{1}{2} V dd^2 f \sum_{\forall sig} C_{sig} \alpha_{sig} \qquad (6)$$

where α_{sig} is given by equation 5 and C_{sig} is given by the method in [3] after estimating L by using floorplanning.

5 Latency improving transformations

In the event of inequality (1) being violated, several latency improving transformations can be applied. These transformations will either aim at increasing the left hand side by increasing τ or at decreasing the right hand side.

5.1 Private Slave Creation

The necessary condition to make a slave interface j private [1] to a master interface i is that

$$cr_{mj} = \begin{cases} 1 & : \quad m = i \\ 0 & : \quad m \neq i \end{cases}$$

If the above relationship holds, slave j need not be connected to any other master than i. Therefore, i should be able to access j without waiting to be granted the bus. For i a decoder and a multiplexor should be added. The decoder will control the multiplexor that will output to i the correct HRDATA and HRESP signals. Figure 2 shows an example of a transformed bus architecture with a private slave.

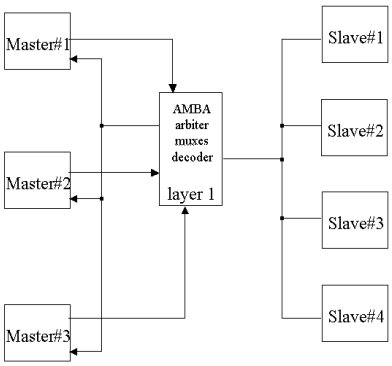

Figure 1. Single-layer architecture with 3 masters and 4 slaves

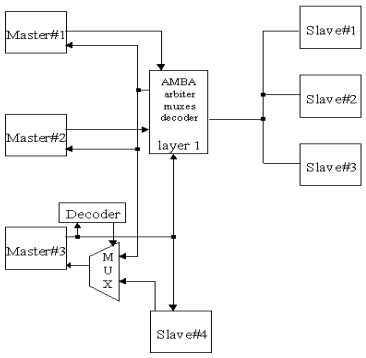

Figure 2. Making slave 4 a private slave for master 3

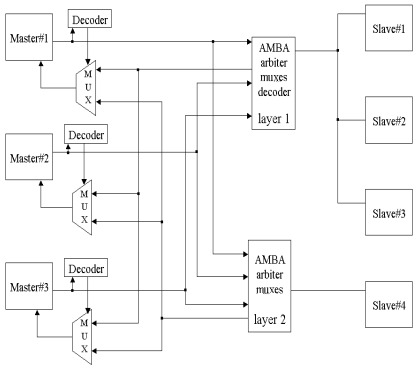

Figure 3. Isolating slave 4 in another layer

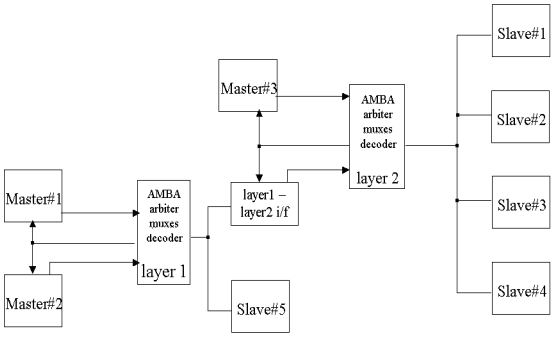

Figure 4. Grouping masters in another layer

5.2 Slave isolation

A slave can be responsible for one or more λ_{max} terms of the right hand side of inequality (1). By moving the slave to another layer the values of the λ_{max}s will be reduced and the constraint will be easier to satisfy. This transformation targets slave interfaces with large latency. An example of this transformation can be seen in Figure 3.

5.3 Grouping masters

Another way to satisfy inequality (1) is to reduce the value of the right hand side by moving a group of master interfaces to another layer as shown in Figure 4. This transformation will decrease the number of masters M of inequality 1 in the transformed layer, as the group of masters will appear as one master in that layer. Therefore, only the maximum of the λ_{max}s of the grouped masters has to be considered and not their sum.

6 Power Improving Transformations

In this section some power improving transformations will be described. These transformations aim at reducing the fanout of signals with high switching activity, so that the power dissipation on the bus wires will be decreased. The dynamic power consumed on the arbiter and the decoder is insignificant compared to the power on the wires and depends on the technology. For recent technologies it will be less. Therefore, it makes sense to concentrate our efforts to decrease the wire power consumption. However, the designer should add an overhead for each component inserted to the SoC due to a transformation. This overhead, which depends on the technology used and the kind of component added, should be used to estimate the impact of each transformation.

After each transformation the floorplanning algorithm should be run again for the new bus architecture to estimate the length of each wire.

6.1 Private Slave Creation

In this section the effect of making a slave private on the power consumption will be explained. We are going to use the following conventions: ζ is the layer from which the slave j was extracted before becoming private to master i.

Making j private can require the addition of a decoder and a multiplexor for controlling the output and input signals for i. For all the signals that are driven by the AMBA logic and are input to the slaves of ζ the fanout will be reduced by 1.

6.2 Slave isolation

This is a transformation that targets the write path of a specific slave. In the event of having a slave j that is heavily accessed especially by write operations, it makes sense to try to isolate j, so that the fanout of the wires accessed during the write operation will be reduced.

After the transformation the number of fanout nodes for each wire connecting the masters with AMBA components will have been increased by one (Figure 3). The fanout of the address bus for the read com-ops involving j will also be decreased.

6.3 Grouping masters

Whereas by isolating a slave in a separate layer we target the write path, by grouping a number of masters in a newly created layer the read path is targeted. The purpose of this transformation is to reduce the fanout of the HRDATA bus. This can be achieved by decreasing the number of masters connected to the bus.

In Figure 4 a number of masters can be grouped in one layer and appear as one master in another. By using this transformation and grouping several masters the fanout of the wires of the HRDATA bus of the second layer will be reduced.

7 Proposed Heuristic

In the previous sections several transformations and their effect on latency and power consumption were described. In this section a heuristic will be proposed to apply these transformations in order to produce a solution that will allow the system to satisfy the latency requirements and work with minimized power consumption.

The proposed heuristic starts by creating an initial solution as a single layer AHB. All master and slave interfaces of the system are connected to it. The second step is to find all potential private slaves for each master and create private connections for them. Since creating private slaves is a latency and potentially power improving transformation it is applied first.

In case inequality (1) is violated for a master latency improving transformations will be applied. First, the introduction of another layer for the slowest slave will be tried. Then the latency constraints of both layers will be checked and if they are satisfied the algorithm will stop. If the constraints are not met the algorithm will try to use the grouping of the masters transformation. The masters with the largest λ_{max} will be put together in a new layer. In case there is a latency violation in the newly created layer the transformation will be undone, else the constraints will be checked for the transformed layer. Latency improving transformations will be continued until either the constraints are met or no transformations can be done to the architecture. In the latter case our method will exit.

If a solution that satisfies the latency constraints is found, the heuristic continues by applying power improving transformations. After each transformation the requirements for the latency of the com-ops are checked again.

As a first step, the power of the current solution is computed by applying floorplanning to find the wire length and by computing the α_{sig} as shown in Section 4.2.

For each interface of the system the *btpc* for the data (HRDATA, HWDATA) and address (HADDR) busses, which are affected by it, is calculated. This is done by examining each com-op and finding *btpc* it causes to the corresponding signals in the design. Then the $btpc^{i}_{HRDATA}$ of each master i is multiplied by the number of masters M_ζ that belong to the same layer ζ as i. In the same way, the sum of $btpc^{j}_{HWDATA}$ and $btpc^{j}_{HADDR}$ for each slave j is multiplied by S_ζ, the number of slaves in the same layer. The result BT_q represents a promising gain by applying a transformation to interface q.

Then starting from the largest BT_q a transformation targeting the corresponding interface q is tried. If q is a slave then it is isolated in another layer. In case q is a master the heuristic will try to group a number of masters of that layer together in one layer. q will certainly not be a member of this set of masters. In our case we try to group all masters except q, so that we can have the largest gain in power. However, a more conservative approach would be to group together the master interfaces with the largest τ. After transforming the design the latency constraints will be checked. If they are not satisfied, the heuristic will choose to undo this transformation. Otherwise, it will go to a power estimation stage. In case the solution produces better power estimation results, it is stored, else the previous design is recovered and the algorithm proceeds with the interface with the next largest BT_q.

These steps will be repeated until all entries of the BT_q array have been checked. Every time a transformation is accepted the array is refilled and sorted. Then the power improving transformations start again with the largest element.

8 Experimental Results

We implemented the power improving heuristic as a C++ program and applied it on two different examples, namely a SoC with a hardware unit implementing the Sobel transform and a SoC with a Vector Quantization Decoder. The initial and resulting architectures can be seen in Figures 5 - 8. Moreover, the results for power consumption can be seen in Table 3.

For the Sobel case with the single-layer approach the constraints are violated and power consumption is high. The application of the proposed heuristic results in an architecture, where SRAM, LCD registers, HW registers, and APB bridge will be private slaves of the ARM, and ARM and PL110's master interface will be grouped together to appear as one master in the layer with the hardware unit and the memory controller (Figure 6).

For the vector quantization case Figure 7 shows the initial architecture for the SoC with all components connected to a single layer AHB. Figure 8 shows the output of the heuristic. After the private slave creation the constraints are still violated. In order for the algorithm to meet the latency constraints SRAM1 is isolated and connected to the ARM core and the LCD master interface. This transformation reduces also the power for the whole bus architecture. In Table 3 the results for this example are summarized.

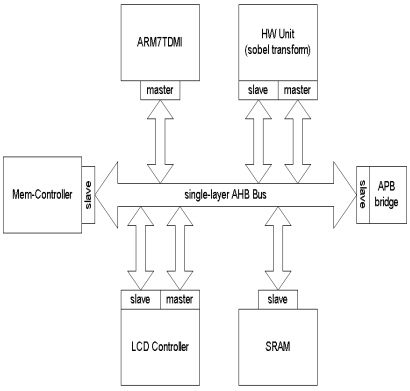

Figure 5. The initial bus architecture for Sobel case

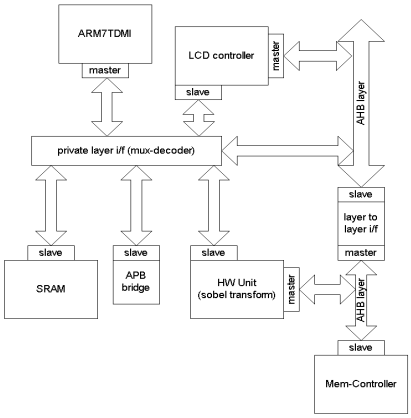

Figure 6. The resulting architecture for the Sobel case

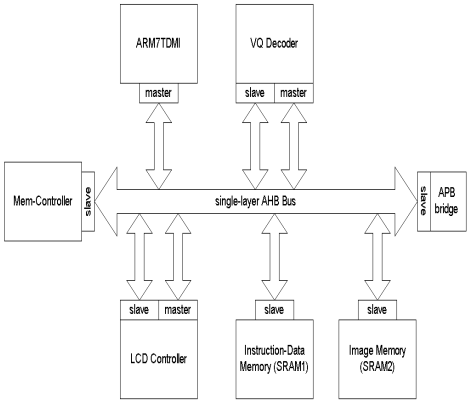

Figure 7. The initial architecture for the VeqDec case

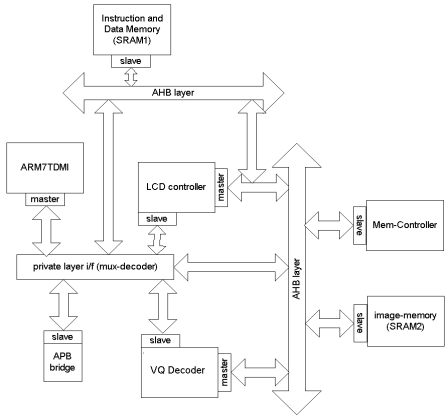

Figure 8. The resulting architecture for the VeqDec case

Sobel	Initial Design	Final Design	Power
Constraints	not-satisfied	satisfied	Reduction
Power(mW)	2.27716	1.60325	29.6%
VeqDec	**Initial Design**	**Final Design**	**Power**
Constraints	not-satisfied	satisfied	Reduction
Power(mW)	2.66482	1.94976	26.8%

Table 3. Results for the Sobel and the VeqDec

9 Conclusions

In this paper we presented a solution to the problem of interface synthesis for a system on a chip (SoC) such that the power consumption is minimized under some given latency constraints. We developed our interface synthesis methods around the AMBA protocol which has become a standard for SoC interfaces.. Several latency improving and power minimizing transformations were presented at the bus level. Finally, a heuristic was presented which applied the above transformations in a certain order to provide minimum power under a given latency constraint. Experimental results were reported on two example benchmarks in that show that the heuristic is able to reduce power consumption on the wires by about 28% on the average from an initial design having a single layer bus architecture.

In the future, we will look at several benchmark designs. We also plan to compare our proposed heuristic with some real commercial designs to see if the power on those designs can be improved. We further plan to use more accurate bus-power models than the one used in this paper since we realize that with 70 nm designs, the static leakage power will become more dominant than the dynamic power which was modeled in this paper.

References

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

[2] T. Givargis, F. Vahid; "Interface exploration for reduced power in core-based systems", ISSS 98, pp 117 -122

[3] Yan Zhang, et. al.; "An alternative architecture for on-chip global interconnect: segmented bus power modeling", Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems and Computers, 1998, pp 1062 -1065

[4] F. Catthoor, et. al. "Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design" Kluwer Academic Publishers

[5] N.D. Zervas, et. al.; "Power exploration of multimedia applications realized on embedded cores", ISCAS 99, pp.378-381

[6] M.R. Stan, W.P. Burleson; "Bus-invert coding for low-power I/O" IEEE Transactions on VLSI Systems, Volume: 3 Issue: 1 , Mar 1995 pp.49-58

[7] L. Benini, et. al.; "Power optimization of core-based systems by address bus encoding" IEEE Transactions on VLSI Systems, Volume: 6 Issue: 4 , Dec 1998 pp.554-562

[8] H. Cheng-Ta, M. Pedram; "Architectural energy optimization by bus splitting" IEEE Transactions on CAD of Integrated Circuits and Systems, Apr 2002, pp.408-414

[9] R.P. Dick, N.K. Jha; "MOCSYN: Multiobjective Core-Based Single-Chip System Synthesis", DATE 99, pp.263-270

[10] R.K. Gupta; "Co-Synthesis of Hardware and Software for Digital Embedded Systems"; Kluwer Academic Publishers

[11] H. Murata, et. al.; "Rectangle-packing-based module placement" ICCAD 95, pp. 472 -479