# Interconnect Delay Aware RTL Verilog Bus Architecture Generation for an SoC

Kyeong Keol Ryu*, Alexandru Talpasanu, Vincent J. Mooney III* and Jeffrey A. Davis
**Center for Research on Embedded Systems and Technology*
*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA*
*{kkryu, alex, mooney, jeff}@ece.gatech.edu*

## Abstract

*As feature size is scaled down to the submicron level, interconnect delay in the design of a high-speed System-on-a-Chip (SoC) becomes a major concern. This concern is especially acute for on-chip buses. In this paper we describe a methodology to generate a custom bus architecture using accurate estimations of interconnect delay. To improve bus delay accuracy, the bus Verilog register-transfer level (RTL) specification was altered based on interconnect delay estimations. Interconnect delay information is provided from an estimated chip layout. The delay estimates for the on-chip buses are used early in the design phase with a corresponding impact on system correctness and performance. As an example of interconnect delay aware bus generation, we compare three different General Global Bus Architecture (GGBA) configurations, showing that certain system blocks (the memory controllers) need to be modified based on interconnect delay estimation.*

*The three different GGBA configurations are evaluated through the simulation of an orthogonal frequency division multiplexing (OFDM) wireless transmitter application. The impact of accurate interconnect delay estimation is shown through a 35.3% reduction in execution time between a worst-case bus delay configuration (GGBA III) and an accurate interconnect delay aware GGBA configuration (GGBA II).*

## 1. Introduction

This paper presents interconnect aware bus synthesis for System-on-a-Chip (SoC). Due to the nature of SoC design, in which multiple Intellectual Property (IP) cores are placed together and connected with global busses, interconnect delay plays a significant role in system performance. This paper shows that a design which takes into account accurate global bus interconnect delay differs by a significant amount in performance from a design with worst-case interconnect delay. Namely, when accurate interconnect delay is included in the RTL specification of the buses in an SoC, the application execution time decreases when compared to a worst-case interconnect delay configuration that has a maximum estimated delays on all data paths. Specifically, in a comparison of accurate interconnect delay aware design case versus worst-case delay aware design in the context of an orthogonal frequency division multiplexing (OFDM) example, the former case has 35.3% performance improvement.

Bus architecture in a multiprocessor SoC plays an important role in system performance. Several efforts from industry provide platforms to connect intellectual property (IP) cores in an SoC, e.g., CoreConnect from IBM [2] and AMBA from ARM [3]. However, these do not integrate custom bus synthesis and interconnect delay.

Several previous works focus on bus architecture synthesis and IP integration for a multiprocessor SoC: CoWare Napkin-to-Chip (N2C) [4] and Shin *et al.* [5]. These tools, however, do not discuss interconnect delay aware bus architecture generation.

Thepayasuwan *et al.* [6] describe layout conscious bus architecture synthesis. Our work, on the other hand, utilizes interconnect delay prediction from a system floorplan to provide a more customized bus architecture that is suitable for a specific user's set of applications since the methodology generates the bus architecture based on various user input options [1].

The floorplan described in this article was created manually; however, automated floorplanning tools such as MOCSYN [15] can produce SoC floorplans which can be used to automatically extract bus interconnect lengths.

## 2. Methodology for bus generation

We discuss interconnect delay aware bus architecture generation in this section. As an example, we show a generation of an SoC bus system called General Global Bus Architecture (GGBA). First, we present how to estimate interconnect delay in a system and then describe the generation of a module that is closely related to the interconnect delay in the system operation. After that, we introduce the bus architecture generation.

### 2.1. Interconnect delay estimation

The method used to estimate bus delay was to construct an estimated floorplan for the system, extract interconnect lengths from the floorplan, and model the respective global buses using circuit simulations tools.

#### 2.1.1. General Global Bus Architecture floorplan

The construction of an estimated floorplan for the GGBA was facilitated by obtaining die area estimates for four PowerPC processing elements (PEs) used in this system. This information was available from the Motorola [8] website. Another element used in the floorplan was the memory module. The area estimate for the SRAM module in the GGBA system was found using the UMC chip-sizer [9] available on the UMC website. The UMC chip-sizer tool was also used to find approximate areas for a bus arbiter, four CPU bus interfaces (CBI), as well as a memory bus interface (MBI).

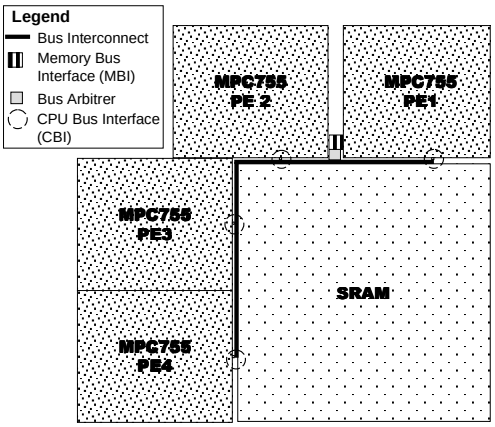



**Figure 1. GGBA estimated layout**

An estimated floorplan of the GGBA architecture is shown in Figure 1. This floorplan was manually created with designer input, but could have been automated by a core placement tool such as MOCSYN [15]. Figure 1 illustrates the floorplan of a global bus connecting the four processing elements and a single memory element. The GGBA floorplan was used to estimate PE-SRAM interconnect lengths; the results are listed in Table 1.

**Table 1. Interconnect length estimation for GGBA system**

| | SRAM Memory | |
|---|---|---|
| | Length [cm] | Delay [ns] |
| Processing Element 1 | 0.2521 | 0.2848 |
| Processing Element 2 | 0.6143 | 0.5727 |
| Processing Element 3 | 1.2753 | 2.2882 |
| Processing Element 4 | 1.9363 | 3.0472 |

### 2.1.2. Bus interconnect physical models

The bus interconnect shown in Figure 1 represents a data bus connecting the four processing elements to the memory. The bus width for this GGBA system is 64 bits. Repeaters are not used in this design because it was found that they take up significant area while offering non-substantial improvements in delay and crosstalk. A more detailed analysis is discussed in a technical report available online [11].

HSPICE simulations were performed on this bi-directional bus to calculate interconnect delay. The HSPICE wire models included resistance, capacitance and inductance values extracted from a MOSIS run [10] for the chosen TSMC 0.25µm technology as well as bus interconnect lengths from the GGBA system floorplan. A set of series connected RLC L-models was used to model each bus wire, with the total resistance [14], inductance and capacitance [12][13] being derived from the total length of the bus. The interconnect length and HSPICE delay estimations between each processing element and the memory are shown in Table 1. The method used to estimate interconnect delay was automated by the use of shell-scripting and a C program.

## 2.2. Memory bus interface (MBI) module generation

One of the effects of interconnect delay insertion in an SoC is in the memory access cycle count of each PE. In this section, we describe an interconnect delay aware memory controller (an MBI module) for a system in its operation, and we also show automatic generation of the MBI.

### 2.2.1. The operation of an MBI module

An MBI module in a system is an interface module operating as a memory controller which is located between a bus and a memory. The module generates PE control signals (e.g., *aack_bar* and *ta_bar* in PowerPC) related to memory access cycle and also generates memory control signals (e.g., *cs_bar, addr, we_bar and oe_bar*). Since moving data to or from memory in a system is deferred due to interconnect delay, suitable memory controlling in the system is required to account for the bus delay so that the system operates without failure and with maximum performance. A method to ensure suitable memory control is to extend every memory access cycle according to the length of the bus interconnect delay. For example, we control two pins of the PowerPC MPC755 for the purpose of memory cycle extension: address acknowledge (*aack_bar*) and transfer acknowledge (*ta_bar*). Here, the *aack_bar* signal terminates an address bus cycle while the *ta_bar* terminates a data bus cycle. To extend each memory cycle, we delay the control signals in a memory access cycle by inserting dummy clock cycles in the memory controller.

### 2.2.2. The generation of an MBI module

Before generating the MBI module, with regard to the estimated interconnect delay shown in Section 2.1, we calculate total delay including the time taken to move controls and data on the bus in two directions (e.g., from a PE to memory and vice versa) and the time taken to access memory in a read operation. However, since transmitting the signals for controls and data on the bus to a shared memory has the same direction to the memory in a write operation, we only consider the effect of bus interconnect delay in a read operation.

**Table 2. Estimated total delay of paths between each PE and a shared memory**

| | Estimated bus delay between a PE to SRAM [ns] | Delay in a read operation [ns] | SRAM (2Mbyte) access time [ns] | Total delay in a read operation [ns] |
|---|---|---|---|---|
| PE 1 | 0.2848 | 0.5696 | 8.00 | 8.8544 |
| PE 2 | 0.5727 | 1.1454 | 8.00 | 9.7181 |
| PE 3 | 2.2882 | 4.5764 | 8.00 | 14.8646 |
| PE 4 | 3.0472 | 6.0944 | 8.00 | 17.1416 |

*1. The access time of a shared SRAM (2Mbytes) is estimated by CACTI 3.0

Table 2 shows estimated delays for the GGBA estimated layout shown in Figure 1. The second column shows estimated interconnect delays described in Section 2.1, and the third column shows bidirectional delays for a read operation. The fourth column shows memory access time for a 2 MByte SRAM, where the access time is estimated by using CACTI 3.0, which is an integrated cache access time, cycle time, area, aspect ratio and power model [7]. Finally, the fifth column is the summation of the previous three columns, that is, total delays considered in read operations.

Table 3 shows the number of clock delay cycles that will be inserted into a memory cycle for the cases that a GGBA system has three different bus clocks, respectively. The total delays shown in Table 2 are divided by each bus clock period in order to obtain the number of clock delays shown in Table 3.

**Table 3. Number of clock delays in data paths**

| | Number of clock delays in each PE for a read operation [clock] | | |
|---|---|---|---|
| | 100 MHz (10.00ns) system clock | 200 MHz (5.00ns) system clock | 300 MHz (3.33ns) system clock |
| PE 1 | 0 (0.8854) | 1 (1.7709) | 2 (2.6589) |
| PE 2 | 0 (0.9718) | 1 (1.9436) | 2 (2.9183) |
| PE 3 | 1 (1.4865) | 2 (2.9729) | 4 (4.4638) |
| PE 4 | 1 (1.7142) | 3 (3.4283) | 5 (5.1476) |

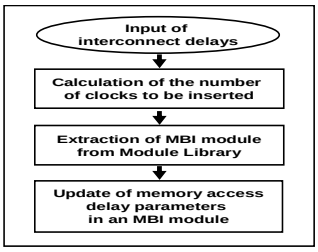



**Figure 2. Sequence of MBI module generation**

Figure 2 describes the sequence of MBI module generation, which is a module generation procedure of our bus synthesis tool (BusSyn) that will be described in Section 2.3. With the input of interconnect delay shown in Table 1, the number of clock cycles required to be inserted for a memory access cycle is calculated in the second step. Then, based on the user input options [1] that configure an SoC bus system with shared memory, an MBI

module is extracted from a Module Library that contains the respective module as a library component [1]. The module is described in Verilog HDL and has pre-defined delay parameters which model corresponding clock delays to memory access cycles. Finally, the delay parameters are updated with the number of clocks calculated in a previous state.

### 2.3. Bus system generation

The flowchart in Figure 3 shows the sequence for bus system generation in our bus synthesis tool (BusSyn) [1]. First, BusSyn takes user input options for a bus system to be generated, and based on these options, BusSyn generates the required bus access nodes (BANs) after generating required modules for the BANs [1]. The MBI module described in Section 2.2 is generated at this time based on the user options. BusSyn subsequently assembles the BANs into required bus subsystems, each of which consists of one or more BANs connected together using bus components. After that, if the bus system the user wants has more than one bus subsystem, the generated bus subsystems are integrated into a resulting bus system. Otherwise, the generated single bus subsystem becomes a bus system. Finally, BusSyn writes synthesizable Verilog HDL code for the generated bus system.

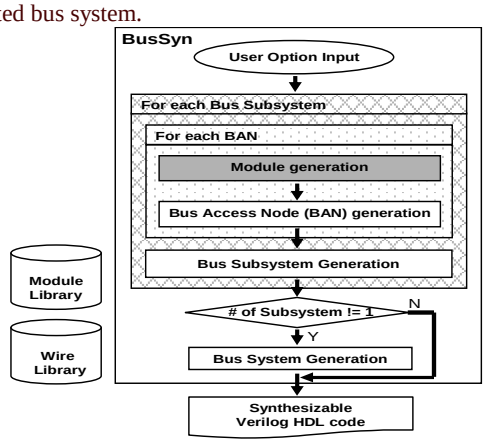



**Figure 3. Sequence of a bus system generation**

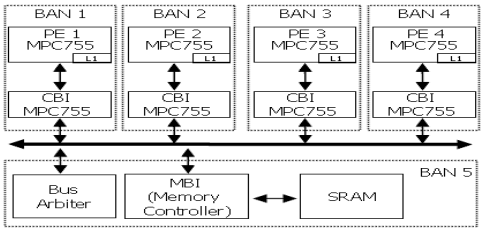



**Figure 4. A GGBA System**

Figure 4 shows an example (GGBA) of a generated bus system that consists of five BANs based on specific input to the generation sequence shown in Figure 3. Four BANs (BANs 1 to 4) have PowerPC MPC755s, and BAN 5 is composed of a global bus arbiter and a shared SRAM. The bus system has a single bus subsystem.

## 3. Application example

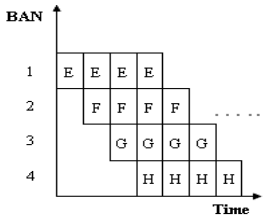
The generated bus system is evaluated to see the impact of interconnect delay in the design phase with an orthogonal frequency division multiplexing (OFDM) wireless transmitter.

Figure 5 shows OFDM function groups to be executed and the computation in each BAN; the details of the application are in [1].

| Group | Functions in an OFDM transmitter |
|---|---|
| E | *Initialization (channel parameters)** <br> *Train pulse and symbol generations** <br> Data generation and symbol mapping <br> Bit reverse for inverse FFT |
| F | Inverse FFT |
| G | Normalizing inverse FFT |
| H | Normalization <br> Insertion of guard signal <br> Data output |

*Italicized functions are executed once at first

(a) Function group in OFDM transmitter




(b) Pipelined parallel processing

**Figure 5. Function assignment and processing**

## 4. Experimental results

Based on the methodology shown in Section 2, we generate an SoC bus architecture called General Global Bus Architecture (GGBA). With the experimental setup shown in Section 4.1, we examine the impact of bus interconnect delay on an SoC design and show a comparison of results in Section 4.2.

### 4.1. Experimental setup

As shown in Figure 6, our bus synthesis tool (BusSyn) takes user input to configure a bus system and interconnect delay estimated from an approximated system floorplan, producing synthesizable Verilog HDL code for the specified custom bus system. The floorplan was created manually. The delay estimation was calculated using automated scripts and programs with bus interconnect lengths from the designed floorplan given as input. Our setup (on the right side of Figure 6) for the bus system simulation is the same as in [1].

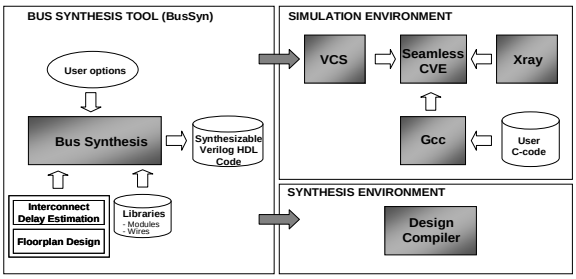



**Figure 6. Experimental Setup**

### 4.2. Comparison of results

To show the impact of interconnect delay prediction in the design phase, we show three different configurations of a GGBA system: GGBA I, GGBA II and GGBA III. These three configurations have the same bus architecture, which is shown in Figure 4; nevertheless, the configurations vary in that each have different memory controllers. The first GGBA system, GGBA I, has a memory controller working with no regard to interconnect delay on the bus between each PE and the shared memory (thus, GGBA I may fail if implemented in a real SoC; nonetheless, GGBA I represents a typical initial simulation with communication across wires occurring instantaneously). The second GGBA system, GGBA II, is generated by BusSyn based on the methodology introduced in Section 2 and has a memory controller that works with different estimated interconnect delays on the shared bus. Here, the delays are provided from an estimated chip layout as introduced in Section 2.1, and the delay values are shown in Table 2. Finally, the third system, GGBA III has a memory controller that operates with a maximum estimated delay on all connections between the PEs and the shared memory. In light of memory access, the third system is a non-

optimized system that can be designed if we only use worst-case interconnect delay information in the design phase.

Table 4 shows execution times for an OFDM packet in GGBAs I, II and III, and their percentage comparison. Here, an OFDM packet consists of 128 real and imaginary data samples and 32 guard data samples. Note that in Table 4 simulations are performed with the bus clocked at 100MHz, 200MHz and 300MHz. For GGBA II and III, both of which account for interconnect delay, the memory controller waits for an appropriate number of bus cycles based on the required delay and the bus cycle time; e.g., a 10ns delay requires only one bus cycle at 100MHz but requires three bus cycles at 300MHz.

#### 4.2.1. Comparison I

In Comparison I of Table 4, GGBA I is used as a baseline for performance degradation according to increasing interconnect delay. In the case of (a) 300MHz bus clock in Table 4, the execution time shown in Comparison I increases up to 161.0% in GGBA III against the result of GGBA I. This increase is due to the fact that GGBA III uses overall worst-case interconnect delay. Here, the performance degradation results from inserting delay clocks into memory access cycles so that the system can operate without failure. In other words, while GGBA I would fail in a real SoC, GGBA III would work fine.

**Table 4. Performance comparison**

| GGBA System | Execution Time [ns/packet] | Comparison I [increase in execution time] | Comparison II [decrease in execution time] |
|---|---|---|---|
| 1. GGBA I (0 clock delays in all data paths) | 1,218,455 | 0.0% | - |
| 2. GGBA II (2, 2, 4 and 5 clock delays in each data path from PE 1 to PE 4) | 2,057,487 | 68.9% | 35.3% |
| 3. GGBA III (5 clock delays in all data paths) | 3,180,220 | 161.0% | 0.0% |

(a) 300 MHz Bus Clock

| GGBA System | Execution Time [ns/packet] | Comparison I [increase in execution time] | Comparison II [decrease in execution time] |
|---|---|---|---|
| 1. GGBA I (0 clock delays in all data paths) | 1,825,751 | 0.0% | - |
| 2. GGBA II (1, 1, 2 and 3 clock delays in each data path from PE 1 to PE 4) | 2,323,670 | 27.3% | 27.4% |
| 3. GGBA III (3 clock delays in all data paths) | 3,198,620 | 75.2% | 0.0% |

(b) 200 MHz Bus Clock

| GGBA System | Execution Time [ns/packet] | Comparison I [increase in execution time] | Comparison II [decrease in execution time] |
|---|---|---|---|
| 1. GGBA I (0 clock delays in all data paths) | 3,644,003 | 0.0% | - |
| 2. GGBA II (0, 0, 1 and 1 clock delays in each data path from PE 1 to PE 4) | 3,862,686 | 6.0% | 10.1% |
| 3. GGBA III (1 clock delays in all data paths) | 4,297,056 | 17.9% | 0.0% |

(c) 100 MHz Bus Clock

#### 4.2.2. Comparison II

In Comparison II of Table 4, GGBA III is chosen as the baseline for performance improvement against the execution time of GGBA II. The impact of detailed interconnection delay estimation in the design phase results in a 35.3% reduction in execution time when we compare GGBA II, an interconnect delay aware GGBA system, with GGBA III, a non-optimized system with regard to memory access cycles. As shown in the cases of (a) 300MHz, (b) 200MHz and (c) 100MHz bus clocks in Table 4, different bus clocks result in different memory access patterns due to interconnect delays. Therefore, as the bus clock increases, the effect of detailed interconnect delay in a system is bigger as shown in Comparison II (and Comparison I) of Table 4.

## 5. Conclusion

In the design of a high-speed SoC, interconnect delay becomes a major concern. In this paper, we describe a methodology to generate a custom bus architecture based on accurate estimations of interconnect delay. The interconnect delay is provided from an accurate delay modeling established from an estimated chip floorplan. Our bus synthesis tool (BusSyn) [1] generates a custom bus system (GGBA) that adapts to detailed interconnect delay predictions, and the generated system is evaluated with a user application, an OFDM transmitter, in order to illustrate the impact of interconnect delay during the design phase.

Our methodology gives us great benefits in performance improvement as well as shortening SoC design time by quick bus architecture generation. The results of our case study show that there is performance improvement due to suitable memory access control that adapts predicted interconnect delay. In particular, we show up to 35.3% reduction in application execution time for a customized bus architecture.

## 6. References


[1] K. Ryu and V. Mooney, "Automated Bus Generation for Multiprocessor SoC Design," *Design, Automation and Test in Europe* (DATE'03), pp. 282-287, March 2003.

[2] IBM, "CoreConnect Bus Architecture," [Online]. Available: http://www-3.ibm.com/chips/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture, 2002.

[3] ARM, "AMBA Specification Overview," [Online]. Available: http://www.arm.com/armtech.nsf/html/AMBA?OpenDocument&style=AMBA, 2002.

[4] CoWare, "CoWare N2C: Design Automation Technology for System-Level Design," [Online]. Available: http://oradev.coware.com:7778/portal/page?_pageid=96,69853&_dad=dcowao&schema=STAGE, 2003.

[5] C. Shin *et al.*, "Fast Exploration of Parameterized Bus Architecture for Communication-Centric SoC design," *Design, Automation and Test in Europe* (DATE'04), pp. 352-357, February 2004.

[6] N. Thepayasuwan and A. Doboli, "Layout Conscious Bus Architecture Synthesis for Deep Submicron Systems on Chip," *Design, Automation and Test in Europe* (DATE'04), pp. 108-113, February 2004.

[7] Hewlett-Packard, "CACTI," [Online]. Available: http://research.compaq.com/wrl/people/jouppi/ CACTI.html, 2004.

[8] Motorola, "MPC 755A RISC Microprocessor Hardware Specification," [Online]. Available: http://e-www.motorola.com/webapp/sps/site/prod_summary, 2002.

[9] UMC, "chip sizer," [Online]. Available: http://eproject.umc.com/dse, 2004.

[10] The MOSIS Service, "TSMC 0.25 Micron Process," [Online]. Available: http://www.mosis.org/products/fab/vendors/tsmc/tsmc025/index.html, May 2003.

[11] A. Talpasanu and J. Davis, "Bus Interconnect Structure for a System-on-a-Chip Multiprocessor System," Georgia Institute of Technology, Atlanta, GA, Technical Report GIT-CC-04-03, [Online]. Available HTTP: http://www.cc.gatech.edu/tech_reports, 2004.

[12] T. Sakurai, "Closed-Form Expressions for Interconnection Delay, Coupling, and Crosstalk in VLSI's," *IEEE Transactions on Electron Devices*, vol. 40, issue 1, pp. 118-124, Jan. 1993.

[13] M. Lee, "A Fringing and Coupling Interconnect Line Capacitance Model for VLSI On-Chip Wiring Delay and Crosstalk," *IEEE Intern. Sym. On Circuits and Systems* (ISCAS '96), vol. 4, pp. 233-236, May 1996.

[14] J. Uyemura, *Introduction to VLSI Circuits and Systems*, John Wiley & Sons, NY, 2002.

[15] R. P. Dick and N. K. Jha, "MOCSYN: Multiobjective Core-Based Single-Chip System Synthesis," *Design, Automation and Test in Europe* (DATE'99), pp. 263-270, March 1999.