CHAPTER 3

DYNAMICALLY RECONFIGURABLE SAMBA
BUS ARCHITECTURES

3.1 INTRODUCTION

Shared bus is the most commonly used bus architecture in SoC. In order to improve the scalability, arbitration delay, communication delay, and bandwidth characteristics of shared bus, high performance SAMBA bus architecture was proposed in Lu and Koh (2003), Ruibing Lu et al (2007). In SAMBA bus architecture, the arbiter grants only one module to access the bus, same as in traditional bus architectures. During a data transfer between two modules in the SAMBA bus, a bus segment (i.e. portion of the bus connection between two neighboring modules), is used only if it is a part of its communication path (Lu and Koh 2003; Ruibing Lu et al 2007). The idle bus segments which are not involved in the data transfer can be used by other pending communications automatically without introducing additional arbitration complexity. Therefore, with a single arbitration, multiple bus accesses are allowed to take place in order to improve scalability, bandwidth and delay of the SAMBA bus. The disadvantage of the existing SAMBA bus is, simultaneous multiple overlapping transactions are not possible (in this work, the pending transactions which share a common bus segment are referred as overlapping transactions).

In order to overcome the disadvantage of SAMBA bus, two dynamically reconfigurable SAMBA bus architectures have been proposed in
this work. Both the proposed bus architectures have the capability of adapting its topology based on the run time requests made by the modules. The proposed bus architectures adapt their topologies such that overlapping transactions are converted into non overlapping transactions thereby allowing multiple overlapping transactions to take place simultaneously. Therefore, the performance is improved in both the proposed bus architectures of this work. The design, working principle and simulation results of the proposed dynamically reconfigurable SAMBA bus architectures are explained in the following sections.

3.2 LITERATURE SURVEY

Standard shared bus architectures such as OCP, AMBA and Core-Connect are popular choices for on chip communication in current designs and open up a large exploration space because they can be configured in so many different ways. The advantages of shared bus architecture are simple topology, low cost, and extensibility (Hsieh and Pedram 2002). The main disadvantage of shared bus communications is that the performance of buses decreases significantly when the bus size (i.e., the number of modules on buses) increases. This is called scalability problem (Lu et al 2005). Another disadvantage is that in the traditional bus only one module can transmit at a time. Moreover, the available bandwidth of one module decreases significantly as the bus size increases. The bandwidth can be improved by hierarchical bus architecture in which multiple buses are connected with each other through bridges (IBM Microelectronics 1999). However, this hierarchical bus architecture suffers from long communication delay. To improve the bus performance, split shared-bus architecture is proposed in Hsieh and Pedram (2002), Lu and Koh (2004).

A high performance communication architecture, SAMBA bus, is proposed in Lu and Koh (2003). In Lu and Koh (2003), the structure,
operation and implementation details of SAMBA bus are explained. The SAMBA bus has large improvements over both effective bus bandwidth and communication latency. In SAMBA bus, multiple compatible transactions can be performed simultaneously with only a single bus access grant from the bus arbiter (Lu and Koh 2003). The main difference between SAMBA bus and traditional buses is that, in SAMBA bus, idle bus segments can be used by other pending communications automatically without introducing additional arbitration complexity. Therefore, with a single arbitration, multiple bus accesses are allowed to improve both bus bandwidth and communication latency. The results in Lu and Koh (2003) show that SAMBA bus architecture has up to 3.5 times improvement in the effective bandwidth and up to 15 times reduction in the average communication latency. Also, the performance of SAMBA bus architecture is affected only slightly by long arbitration latency. This feature is desirable in existing SoC designs with large number of modules and long communication delay between modules and the bus arbiter.

In Ruibing Lu et al (2007), Lu et al (2005) methods for bus cycle time reduction are proposed to improve the scalability of the SAMBA bus architecture further. Lu and Koh (2003), Lu et al (2005) are the preliminary versions of Ruibing Lu et al (2007). The bus clock period is determined by the data/address propagation delay from one bus end to the other end and the arbitration delay (Ruibing Lu et al 2007). As all interface units on a SAMBA bus are connected in series along the bus, the bus clock cycle time increases significantly due to the logic delay of interface units when the bus size is large. In Ruibing Lu et al (2007), Lu et al (2005) two methods of reducing the bus logic delay are proposed inorder to improve the scalability of SAMBA bus. The two methods are namely, control signal lookahead method and module clustering method. The module clustering method reduces the number of interface units on the bus, while the control signal lookahead method reduces the delay of one interface unit. Therefore, these two methods are
combined together to reduce the bus clock cycle time further. The results show that there is about 70% reduction in bus logic delay.

### 3.3 EXISTING SAMBA BUS ARCHITECTURE

The structure of the existing SAMBA bus architecture is described in this section. The structure is taken from Lu and Koh (2003), Lu et al (2005) and Ruibing Lu et al (2007). This architecture supports modules that can function as both masters and slaves. The SAMBA bus architecture requires that the addresses of modules from one end of the bus to the other end are in increasing or decreasing order. The overall structure of the SAMBA bus architecture is shown in Figure 3.1. It consists of two sub-buses with opposite signal propagation directions. Each of the two sub-buses is used for the transmission of data in one direction. The sub-bus transferring data from low-address modules to high-address modules is called forward sub-bus and the other bus is called backward sub-bus.

![Figure 3.1 Structure of a SAMBA bus](image)

A master/slave module is attached to the bus through an interface unit, which can communicate with other interface units through both forward and backward sub-buses (Lu et al 2005; Ruibing Lu et al 2007). Basically, there are two bus operation phases in each bus cycle: request phase and response phase (Ruibing Lu et al 2007). Communication requests are sent to destination modules by bus transaction initiators (source modules) on both sub-buses in request phase. Those requests are acknowledged by destination modules in response phase. Before a module performs a bus transaction, its
interface unit first decides which sub-bus should be used based on the destination address, and sends to the arbiter a bus access request for that sub-bus. At the same time, it monitors the bus activities and the arbitration result. The arbiter broadcasts the arbitration winner address to all interface units; therefore, every interface unit knows which module is granted the bus access.

In Figure 3.1 multiplexers are used to combine all signal sources, and each interface unit has a multiplexer on one sub-bus (Lu and Koh 2003; Ruibing Lu 2007). Through the multiplexer, either the address/data information received from the previous unit or the pending address/data of this unit will be propagated to the next interface unit (Ruibing Lu 2007). The main difference between the SAMBA bus architecture and traditional bus architectures is that a unit may access the bus even if it is not the arbitration winner. Instead, as long as the communication paths of these bus transactions do not have common bus segments, they can be performed simultaneously.

When a unit $M_S$ has a pending communication to unit $M_D$ ($M_S < M_D$) for the forward sub-bus and the forward arbitration winner is unit $M_W$. Unit $M_S$ can initiate its transaction on the forward sub-bus if any of the following three sets of conditions is met (Ruibing Lu et al 2007).

1) Unit $M_S$ wins the arbitration and obtains the forward sub-bus access right, i.e., $M_S = M_W$.

2) (a) The bus transaction destination unit $M_D$ is not after the arbitration winner $M_W$, i.e., $M_D <= M_W$ and

   (b) No units before unit $M_S$ perform bus transaction with units after unit $M_S$.

3) (a) Unit $M_S$ is after the arbitration winner $M_W$, i.e., $M_S > M_W$ and

   (b) No unit before unit $M_S$ perform bus communication with units after unit $M_S$. 
Obviously, the winner unit can access the forward sub-bus in the first situation. The second situation is for those bus transactions that can finish in the portion of the bus before the arbitration winner. Condition 2(b) avoids the bus conflict with transactions from other units before the arbitration winner, whereas condition 2(a) avoids the conflict with the bus transaction from the arbitration winner. Condition 3 is for those bus transactions performed on the bus segments after the arbitration winner. Since the unit $M_S$ and its communication destination $M_D$ are all after the arbitration winner $M_W$, there is no need to consider the conflict with the arbitration winner as long as condition 3(b) is satisfied. A pending communication which satisfies the previous condition (2) or (3) is called a compatible communication (Ruibing Lu et al 2007). Therefore, on a SAMBA-bus the winner communication and its compatible communications can be performed simultaneously (Ruibing Lu et al 2007).

Figure 3.2 Multiple bus accesses with single arbitration

Figure 3.2 shows an example for simultaneous multiple bus accesses in the SAMBA bus architecture (Ruibing Lu 2007). The bold lines show the data propagation paths in the request phase. Unit 3 is the arbitration winner; therefore, the bus transaction from 3 to 5 should be performed. The destination unit of the pending communication of Unit 1 is Unit 2, not after the arbitration winner, and no data is being transferred through Unit 3. These satisfy conditions 2(a) and 2(b). Therefore, Unit 1 can perform its communication on the bus in the same bus cycle.
3.4 PROPOSED BUS ARCHITECTURES

In the proposed work, a bus system with four modules is considered. The four modules are \( M_1, M_2, M_3 \) and \( M_4 \). In the existing SAMBA bus architecture, a pending communication satisfying the condition (2) or (3) (mentioned in section 3.3) is called a compatible communication (Ruibing Lu 2007). Therefore, on a SAMBA bus, the winner communication and its compatible communications can be performed simultaneously. But, if the winner communication and the pending communications have common bus segments (i.e., overlapping transactions), then multiple transactions cannot be performed simultaneously. This is the disadvantage of the existing SAMBA bus architecture. This disadvantage is overcome in the proposed work. In this section, two new dynamically reconfigurable SAMBA bus architectures are proposed. Both the proposed bus architectures are capable of adapting its topology according to the pending communication requests and can allow even multiple overlapping transactions to be performed simultaneously. Thus, the proposed dynamically reconfigurable bus architectures can overcome the disadvantage of the existing SAMBA bus architecture (Ruibing Lu 2007).

Reconfigurability or adapting the topology of the bus can be done by using any one of the two techniques (Sekar 2005; Sudeep Pasricha 2008) which are dynamic by-pass technique and dynamic component remapping technique.

3.4.1 Proposed Architecture I (By-pass architecture)

The proposed architecture I uses dynamic by-pass technique to incorporate dynamic reconfigurability in the bus. In this By-pass architecture, additional components such as multiplexers are added to the existing architecture such that every module has a direct path with every other module.
in the bus. Thus, a by-pass path is provided between each and every other modules in the bus. Therefore, when the pending communications have common bus segments on their paths (i.e. overlapping transactions), the modules choose the by-pass path instead of the normal forward/backward bus paths given in Lu and Koh (2003) and Ruibing Lu et al (2007). The proposed By-pass architecture is shown in Figure 3.3.

![Figure 3.3 By-pass architecture](image)

**Figure 3.3 By-pass architecture**

The By-pass architecture which is shown in Figure 3.3 consists of a bus with four modules namely, Module₁, Module₂, Module₃ and Module₄. A multiplexer is present between every adjacent modules in the bus. The multiplexers are named as Mux₁, Mux₂, Mux₃ and Mux₄. The output from every module is connected to one of the input lines of all the multiplexers. The select signals for the four multiplexers are generated by the arbiter based on the requests from the modules. In By-pass architecture, there is a direct path between each and every modules connected to the bus. For eg. if Module₁ (M₁) wants to communicate with Module₃ (M₃) then the arbiter generates the by-pass-3 signal. M₁ communicates with M₃. Simultaneously, if
Module_2 (M_2) is to communicate with Module_4 (M_4) then the arbiter generates by-pass signal. So, M_2 communicates with M_4. Thus, the two overlapping transactions, namely, M_1 to M_3 and M_2 to M_4 take place simultaneously.

Therefore, the bandwidth of the bus is increased by a factor of 2. Likewise, all modules have direct paths between them and they do not have to always pass through the interface units of the intermediate modules and hence, the delay because of passing through the interface units is eliminated.

Unlike the existing SAMBA bus, multiple overlapping communications can take place simultaneously. For eg., let there be two pending communications such as M_1 to M_4 and M_2 to M_3. In the existing SAMBA bus, both pending transactions cannot take place simultaneously because they have to share a common bus segment (i.e. portion of the bus between M_2 and M_3). Therefore, the two pending communications cannot be made during the same bus cycle. They can be completed only in two consecutive bus cycles. Therefore, one of the communications have to wait until the previous communication is over. So, the waiting time is increased and hence the communication latency is thereby increased. But in the proposed By-pass architecture, the two pending communications can take place simultaneously because of the by-pass paths. M_2 communicates with M_3 through Mux_3 and M_1 simultaneously communicates with M_4 through Mux_4 as shown in Figure 3.3. Therefore only one bus cycle is needed to complete both the pending communications mentioned.

Since, there is a direct path between each and every modules of the bus, more than one overlapping transactions can take place simultaneously. Suppose the bus has eight modules connected with it, then a maximum of four overlapping transactions can take place simultaneously. Thus, the bandwidth increases with number of modules connected in the bus. But in the existing SAMBA bus architecture, two overlapping transactions cannot take place
simultaneously. Therefore, in the existing SAMBA bus, the bandwidth increases only if there are nonoverlapping (compatible) transaction requests from the modules. Thus, one disadvantage of the existing SAMBA bus is overcome in the proposed By-pass architecture.

Thus, with this proposed By-pass architecture reconfigurability of topology is obtained and the draw back (multiple overlapping transactions) of the existing SAMBA bus has been overcome. The advantage of the by-pass topology is that the overall communication latency (delay) is decreased because in a single arbitration multiple overlapping transactions are performed. The second advantage is, during a data transfer between a source and destination module, the data need not have to pass through the interface units of the intermediate nodes and so the delay is reduced further. The bandwidth also increases due to simultaneous multiple transactions. Thus, the proposed By-pass architecture has a better performance than the existing SAMBA bus architecture.

3.4.2 Proposed Architecture II (Component-remapping architecture)

The Proposed Architecture II uses dynamic component remapping technique in order to incorporate dynamic reconfigurability in the bus. In the existing SAMBA bus architecture (Ruibing Lu 2007), the addresses of the modules are to be assigned either in ascending order or in descending order. For simulation of existing SAMBA bus (Ruibing Lu 2007), the order of the addresses assumed in this research work is ascending order. The principle of the proposed component-remapping architecture is, the position/location of the modules in the bus is dynamically remapped (according to the real-time requests from the modules) such that the destination module is placed next to the source module, i.e. source and destination are placed adjacent to each other. Hence, the name dynamic component remapping technique. In physical implementation, the process of remapping the position/location of a module is
to be understood as switching the connectivity of that particular master’s interface unit from its initial position to the new position designated by the remap unit, in the bus. The remap unit works on the basis of a remap algorithm which is proposed in the next section. Hence, in the proposed component-remapping architecture, the positions of the modules are dynamically remapped during every bus cycle and the process is such that after remapping, the addresses of the rearranged modules are not changed. Since the source and destination modules are brought to adjacent positions by dynamic remapping, the pending communications will not require common bus segments. Therefore, even overlapping communications are converted into compatible communications. Thus, the disadvantage of the existing SAMBA bus architecture is overcome in component-remapping architecture.

Another advantage of the proposed component-remapping architecture is, arbiter module is not needed here. This is because, the remap unit receives the requests from the modules and remaps the positions of the modules and thus enables all transactions (both overlapping and non-overlapping) to take place. Hence, arbitration process is not needed in the proposed component-remapping architecture at all. Since, arbitration delay is minimum, the communication delay is very less in the proposed component-remapping architecture and also the communication delay between any two modules is reduced by a large factor since the source and destination modules are brought adjacent to each other.

In component-remapping architecture, if the bus consists of four modules, then two simultaneous transactions can take place. If the bus consists of eight modules, then four simultaneous transactions can take place. But in the existing SAMBA bus, only if there are compatible transactions then they can be performed simultaneously else, if there are overlapping transactions then it cannot be performed simultaneously.
Figure 3.4.a shows the proposed component-remapping architecture. Since no additional components are added, the structure of the proposed component-remapping architecture is similar to the existing SAMBA bus architecture (Lu 2003; Ruibing Lu 2007). The only difference between these two architectures is, the remap unit present in the component-remapping architecture.

Inside the remap unit, the positions of the modules are represented by a 2 bit value. Suppose there are two pending requests B to C and A to D, when the bus is in initial condition, the position of the modules are as shown below:

Module A is 00
Module B is 01
Module C is 10
Module D is 11

![Diagram of component-remapping architecture](image)

**Figure 3.4.a Component-remapping architecture**

The remap unit reassigns the positions of the modules according to the remap algorithm. After re-mapping, the positions of the modules are reassigned as shown below.
Module A is 00
Module D is 01
Module B is 10
Module C is 11

Figure 3.4.b shows the positions of the modules after re-mapping process.

![Diagram showing positions of modules]

Figure 3.4.b Topology of the proposed bus architecture after re-mapping process is over

3.4.2.1 Remap unit

When four modules are connected in the bus, an unique 2-bit address is assigned for each module. If there are eight masters then 3-bit address is assigned to each module. At the beginning of every bus cycle, the remap unit, reads the requests of the masters and their respective destination addresses. Then, it reassigns the positions of the modules according to the proposed remap algorithm. The remap algorithm works in such a way that, for all pending communications, the sources and their respective destinations are placed adjacent to each other. The remap algorithm is given in Figure 3.5.
**Pseudo code:**

Let the set of modules in the bus be $M = \{M_1, M_2, M_3, ..., M_n\}$;
Assume their priorities be $n, n-1, n-2, ..., 1$ respectively;
Get the request signals from the modules at the beginning of a bus cycle;
Let $pos=0$;
while($M \neq \emptyset$)
begin
  Find the module $M_{hp} \in M$;
  Put $M_{hp}$ in the position $pos$;
  $pos++$;
  $M = M - \{M_{hp}\}$;
  Find the module $M_x \in M$ which wants to communicate with $M_{hp}$;
  Put $M_x$ in the position $pos$;
  $pos++$;
  $M = M - \{M_x\}$;
end;

where
$n$ is the number of modules connected to the bus
$pos$ is a variable which denotes the position of the modules connected in the bus
$M_{hp}$ is the module which has the highest priority

**Figure 3.5 Proposed Remap algorithm**

3.5 **SIMULATION RESULTS**

The bus architectures are modeled in Verilog HDL and the simulations are done using Xilinx ISE 9.2i tool. Area and Power results are obtained using Synopsys Design Vision tool. Different types of simulations are done and the results for various parameters are tabulated. The results of the existing and proposed architectures are compared and discussed in this section.
3.5.1 Simulation results for Area, Power and Delay

Simulations are done for the existing SAMBA bus, proposed By-pass architecture and proposed component-remapping architecture. Table 3.1 shows results when 4 modules are connected in the bus.

Table 3.1 Area, power and frequency results for 4 modules

<table>
<thead>
<tr>
<th>Bus parameters</th>
<th>Existing SAMBA-Bus Architecture</th>
<th>Proposed By-pass architecture</th>
<th>Proposed Component-remapping architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum clock period (ns)</td>
<td>0.898</td>
<td>1.422</td>
<td>0.903</td>
</tr>
<tr>
<td>Power (W)</td>
<td>869u</td>
<td>1.2m</td>
<td>893u</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>0.06</td>
<td>0.10</td>
<td>0.06</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>1113.77</td>
<td>703.24</td>
<td>1107.48</td>
</tr>
</tbody>
</table>

Simulation results show that for By-pass architecture, the power and area are higher than the existing SAMBA bus architecture and the frequency (maximum operating frequency) has reduced than the existing SAMBA bus architecture. This is because in By-pass architecture, extra multiplexers are used. So, because of these extra hardware components the power and area values have increased in By-pass architecture. Since the minimum clock period has increased, frequency has decreased for By-pass architecture. But in case of Component-remapping architecture, extra hardware components are not used and in this architecture, only changing or re-mapping the position of modules depending upon request signals takes place. So in Component-remapping architecture, there is a slight overhead in power and minimum clock period whereas the area remains the same as that of the existing SAMBA -bus architecture.
Table 3.2 shows results when 8 modules are connected in the bus. As the number of modules increases, the area, power, and delay values increase proportionately for all the three architectures in the table. When By-pass architecture is compared with the existing SAMBA bus architecture, power, and area has increased by a considerable amount whereas the frequency of operation has not changed with the increased number of modules. In component-remapping architecture, power and area remains the same as the existing SAMBA bus architecture and frequency of operation also remains approximately equal to the existing SAMBA bus architecture. This shows that in component-remapping architecture, when the number of modules increases area, power and frequency are not much affected. This shows that the proposed component-remapping architecture has high scalability.

### Table 3.2 Area, power, and frequency results for 8 modules

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Existing SAMBA Bus Architecture</th>
<th>Proposed By-pass architecture</th>
<th>Proposed Component-remapping architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum clock period (ns)</td>
<td>0.898</td>
<td>1.422</td>
<td>0.903</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>1.8</td>
<td>2.8</td>
<td>1.8</td>
</tr>
<tr>
<td>Area (mm$^2$)</td>
<td>0.13</td>
<td>0.27</td>
<td>0.13</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>1113.772</td>
<td>703.24</td>
<td>1107.48</td>
</tr>
</tbody>
</table>

**3.5.2 Simulation results for Average communication delay**

The time duration for a data transaction is called communication delay. i.e., the time it takes for the data to travel from source to the destination is called communication delay. Average communication delay is defined as the average of communication delays of all different possible transactions that
can be performed in the bus. Table 3.3 shows the comparison of communication delay for three architectures for different transactions.

In Table 3.3, for existing, proposed By-pass architecture and proposed Component-remapping architecture, communication delay is same for \( M_1 \rightarrow M_2, M_3 \rightarrow M_4 \) and \( M_2 \rightarrow M_3 \) transactions. When the distance between the source and destination modules increases (for e.g. \( M_1 \rightarrow M_4 \)), the communication delay increases proportionately in the case of the existing SAMBA-bus architecture.

**Table 3.3  Comparison of Average Communication Delay (using 4 modules)**

<table>
<thead>
<tr>
<th>Transactions between the Modules</th>
<th>Communication Delay(ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing SAMBA bus architecture</td>
</tr>
<tr>
<td>( M_1 \rightarrow M_2 )</td>
<td>102.4</td>
</tr>
<tr>
<td>( M_3 \rightarrow M_4 )</td>
<td>102.4</td>
</tr>
<tr>
<td>( M_2 \rightarrow M_3 )</td>
<td>102.4</td>
</tr>
<tr>
<td>( M_1 \rightarrow M_4 )</td>
<td>502.4</td>
</tr>
<tr>
<td>( M_2 \rightarrow M_4 )</td>
<td>302.4</td>
</tr>
<tr>
<td>( M_1 \rightarrow M_3 )</td>
<td>302.4</td>
</tr>
<tr>
<td>Average communication Delay</td>
<td>262.4</td>
</tr>
</tbody>
</table>

In case of By-pass architecture, due to the direct communication between one module to other module through multiplexer the communication delay is less. The distance between the source and destination modules does not affect the communication delay because the data from any source has to travel through only one multiplexer stage however far the destination may be from the source. So, the communication delay is fixed for all transactions that may take place in the bus. Therefore, the average communication delay for
By-pass architecture is decreased by 18% when compared to the existing SAMBA bus architecture and in the case of component-remapping architecture, communication delay is less because in this architecture no extra circuitry is added. So, the average communication delay for component-remapping architecture is decreased by 61% when compared to the existing SAMBA bus architecture.

Table 3.4 shows the comparison of average communication delay when 8 modules are connected in the bus. As the number of modules increases the average communication delay also increases. There are 28 different types of transactions that are possible in a bus which has 8 modules. The average of the individual communication delays for all the 28 transactions gives the average communication delay of the bus. In the existing SAMBA bus architecture, the communication delay increases when the distance between the source and destination modules increases. For e.g., the communication delay between M₁, M₂ is 100 ps whereas between M₂, M₆ is 700 ps. Therefore, the average communication delay is also as high as 517 ps as shown in Table 3.4. In By-pass architecture, all the transactions originating from a particular module (but different destinations) takes place through only one multiplexer stage. So, the delay is same for all the transactions. i.e, the communication delay between M₁, M₂ is 215 ps and between M₂, M₆ is also 215 ps. The delay depends only on the size of the multiplexer used in the bus.

<table>
<thead>
<tr>
<th>Table 3.4 Comparison of Average Communication Delay (using 8 modules)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average communication delay</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Therefore, the average communication delay of By-pass architecture is 58.4% lesser than the existing SAMBA bus architecture. The average communication delay of component-remapping architecture is 58.4% lesser than the existing SAMBA bus architecture.

### 3.5.3 Communication delay for multiple overlapping transactions

In the existing SAMBA bus architecture, multiple overlapping transactions are not possible, i.e. the transactions which use a common segment cannot be performed simultaneously. So, such overlapping transactions can be performed only one after the other. So it requires two separate bus cycles and one arbitration cycle in between two bus cycles. Therefore, the communication delay of overlapping transactions gets increased because of this reason. But, in both the proposed architectures this drawback is completely overcome because of dynamic reconfigurability feature of the bus. Table 3.5 shows the communication delay for multiple overlapping transactions for a bus having 4 modules connected to it. Table 3.5 shows the reduction in the communication delay in the proposed architectures when compared to the existing architecture.

For e.g., let there be two pending transactions $M_1 \rightarrow M_4$ and $M_2 \rightarrow M_3$. The bus segment between the modules $M_2$, $M_3$ is common for both transactions. So, according to the existing architecture, the two transactions cannot be performed simultaneously and so it requires two complete bus cycles. So, the total communication delay to complete these two transactions will be \([(\text{delay of } M_1 \rightarrow M_4 = 502.4 \text{ ps}) + (\text{arbitration delay}) + (\text{delay of } M_2 \rightarrow M_3 = 102.4 \text{ ps})]\) which would be definitely greater than atleast 600 ps. But in the proposed architectures, the two transactions can be performed simultaneously and the total communication delay to complete the two transactions simultaneously will be only 215 ps and 102 ps respectively. Table 3.5 shows that delay for multiple overlapping transactions has reduced
by approximately 64% in case of By-pass architecture and by approximately 83% in case of component-remapping architecture.

Table 3.5  Communication Delay for multiple overlapping transactions (bus with 4 modules)

<table>
<thead>
<tr>
<th>Multiple overlapping Transactions</th>
<th>Existing SAMBA bus architecture delay (ps)</th>
<th>Proposed By-pass architecture delay (ps)</th>
<th>Proposed Component-remapping architecture delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁ → M₄, M₂ → M₃</td>
<td>(502.4+arbitration delay+102.4) &gt; 600</td>
<td>Max(215, 215) = 215</td>
<td>Max(102, 102) = 102</td>
</tr>
</tbody>
</table>

Table 3.6 shows the communication delay in case of multiple overlapping transactions for a bus with 8 modules.

Table 3.6  Communication Delay for multiple overlapping transactions (bus with 8 modules)

<table>
<thead>
<tr>
<th>Multiple overlapping transactions</th>
<th>Existing SAMBA bus architecture delay (ps)</th>
<th>Proposed By-pass architecture delay (ps)</th>
<th>Proposed Component-remapping architecture delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₂ → M₆, M₄ → M₅</td>
<td>(700+100+Arbitration delay) &gt; 800</td>
<td>Max(215, 215) = 215</td>
<td>Max(215, 215) = 215</td>
</tr>
</tbody>
</table>

Table 3.6 shows that delay for multiple overlapping transactions has reduced by approximately 73% in case of By-pass architecture and component-remapping architecture.
3.5.4 Comparison of existing architecture, By-pass architecture and component-remapping architecture

Table 3.7 shows the overall comparison between existing architecture, By-pass architecture and component-remapping architecture. Table 3.7 summarizes the results shown in the tables Table 3.2 and Table 3.4.

Table 3.7 Overall comparison of existing architecture, By-pass architecture and Component-remapping architecture

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Existing SAMBA bus architecture</th>
<th>Proposed By-pass architecture</th>
<th>Proposed Component-remapping architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (mm²)</td>
<td>0.13</td>
<td>doubled</td>
<td>No overhead</td>
</tr>
<tr>
<td>Power (mw)</td>
<td>1.8</td>
<td>Increase about 50%</td>
<td>No increase</td>
</tr>
<tr>
<td>Frequency (MHZ)</td>
<td>1114</td>
<td>703.24</td>
<td>1107.48</td>
</tr>
<tr>
<td>Average Communication delay (ps)</td>
<td>517</td>
<td>Decreased by 58.4%</td>
<td>Decreased by 58.4%</td>
</tr>
</tbody>
</table>

From Table 3.7 shown above, area and power for By-pass architecture have increased when compared to the existing SAMBA bus architecture. This is due to extra multiplexers added in the architecture. There is no area and power overhead in the case of component-remapping architecture because in this architecture, extra circuitry is not used. In both the proposed architectures, the average communication delay is reduced by 58.4%. 
3.5.5 Average Communication Delay Vs Communication distance

In a bus transaction, communication distance (d) refers to the number of interface units between the transaction initiator module and the destination module (Ruibing Lu 2007). Table 3.8 shows the average communication delay/latency for different values of d for the bus using 8 modules.

Table 3.8 Average Communication Delay Vs Communication distance (d)

<table>
<thead>
<tr>
<th>Bus Architecture</th>
<th>Average communication delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>uniform</td>
</tr>
<tr>
<td>Existing SAMBA bus</td>
<td>517.4074</td>
</tr>
</tbody>
</table>

Table 3.8 shows how the average communication delay varies with respect to the communication distance. In the table, uniform communication distance (d) means that it is equally likely for any unit to be the destination (Ruibing Lu 2007). If communication distance (d) =1, one module should be present between the source and destination (e.g. M₁ → M₃). If communication distance (d) =2, it means two modules should be present between the source and destination (e.g. M₁ → M₄). If communication distance (d) =3, it means three modules should be present between the source and destination (e.g. M₁ → M₅). Thus, for seven communication distance distributions, the average communication delay is tabulated in Table 3.8.
For the existing SAMBA bus, average communication delay increases with communication distance because data has to pass through more number of interface units on the way of its path to destination. In By-pass architecture, average communication delay is less compared to SAMBA bus, because data has to pass through only one multiplexer whatever may be the communication distance. So, if the communication distance increases the delay does not increase. The delay depends only on the size of the multiplexer and the wiring lengths which in turn depends only on the number of the modules connected to the bus. In Component-remapping architecture, average communication delay is constant with the increase in communication distance. In this Component-remapping architecture, all the destination modules” positions are re-mapped near to their respective source modules. So, the proposed By-pass architecture and the proposed Component-remapping architecture give better performance for varying communication distance distributions when compared to the existing SAMBA bus architecture.

3.5.6 Average Communication Delay Vs Number of modules

Table 3.9 shows how the average communication delay varies with respect to the number of modules in the bus. Firstly, the three architectures are simulated using 4 modules only and their average communication delays are obtained. Then the three architectures are simulated using 8 modules and their average communication delay values are obtained and tabulated.
Table 3.9 Average communication delay Vs Number of modules

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Average communication delay for 4 Modules</th>
<th>Average communication delay for 8 Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing SAMBA bus architecture</td>
<td>262.4</td>
<td>507.41</td>
</tr>
<tr>
<td>Proposed By-pass architecture</td>
<td>215</td>
<td>215</td>
</tr>
<tr>
<td>Proposed Component-remapping architecture</td>
<td>102</td>
<td>215</td>
</tr>
</tbody>
</table>

Table 3.9 shows that for all the three architectures the average communication delay increases when the number of modules connected in the bus increases. For the existing SAMBA bus architecture, the increase % of average communication delay is high whereas for the proposed architectures the increase % of average communication delay is only little. This shows that for the proposed architectures, the performance does not get degraded as much as the existing SAMBA bus architecture. Therefore, the scalability is high for the proposed architectures.

3.5.7 Bandwidth characteristics

Bandwidth is calculated as number of transactions performed in single bus cycle. Compared to the existing SAMBA bus architecture, the bus bandwidth increases both in proposed By-pass architecture and proposed Component-remapping architecture when more number of modules is connected to the bus. The proposed By-pass architecture and Component-remapping architecture perform multiple overlapping transactions and due to that effective bandwidth is increased compared to the existing SAMBA bus. But in existing SAMBA bus, overlapping transactions cannot be performed simultaneously. So, when the pending communications are overlapping, the
bandwidth of the existing SAMBA bus is less. In the two proposed architectures, when the number of modules increases, then number of multiple transactions that can be performed simultaneously also increases. So, the bandwidth also increases. For e.g., if there are 4 modules in the bus, then 2 transactions can take place simultaneously and hence bandwidth gets doubled. When there are 8 modules in the bus, then 4 transactions can take place simultaneously and hence the bandwidth gets multiplied by four. Therefore, in general, if there are n modules in the bus, then n/2 number of simultaneous transactions are possible and the bandwidth becomes n/2.

3.6 CONCLUSION

In the existing SAMBA bus architecture, the main disadvantage is that, multiple overlapping transactions cannot be performed simultaneously. So, to overcome the disadvantage of the existing SAMBA bus architecture and also to further enhance its performance, two reconfigurable SAMBA bus architectures are proposed in this work. The two proposed architectures are capable of adapting their topology based on the real time requests from the communicating modules in the bus. The reconfigurability techniques used in the proposed architectures are dynamic by-pass technique (applied in By-pass architecture) and dynamic component remapping technique (Component-remapping architecture). Since these proposed architectures are dynamically reconfigurable they can adapt their topology and allows multiple overlapping transactions to be performed simultaneously.

By-pass architecture shows 58.4\% reduction in average communication delay. In case of multiple overlapping transactions, there is 73\% reduction in communication delay. This proposed architecture”s performance does not get much degraded with increase in communication distance and increase in number of modules. Therefore, By-pass architecture is highly scalable. Bandwidth increases with increase in the number of
modules connected to it. So bandwidth performance is also high for the proposed By-pass architecture.

Component-remapping architecture shows 58.4% reduction in average communication delay when compared to the existing SAMBA bus architecture. In case of multiple overlapping transactions, there is 73% reduction in communication delay. Component-remapping architecture also has better performance when there is increase in communication distance and in number of modules connected. Therefore, Component-remapping architecture is highly scalable. Bandwidth increases with increase in the number of modules connected to it. Moreover, there is no area overhead when compared to the existing SAMBA bus design.

Since the proposed architectures have the advantages like, high bandwidth, less communication delay and high scalability, they can be used for high speed SoC applications.