Journal of Computer Science 2 (1): 63-71, 2006
ISSN 1549-3636
© 2006 Science Publications

Communication Architecture Synthesis for Multi-bus SoC

Abdelkrim Zitouni, Sami Badrouchi and Rached Tourki
Laboratoire EμE, Faculté des Sciences de Monastir, 5019 Monastir, Tunisie

Abstract: In the systems on chip (SoC) design, the synthesis of communication architecture constitutes the bottleneck which can affect the performances of the system. Various schemes and protocols can be necessary, just as various topologies of interconnection. To reduce the complexity of the communications refinement, we present in this study a model and a synthesis approach for multi-bus communication architecture containing centralized bridge. The automation of the arbiter synthesis step profited from a detailed attention. This stage generates a hierarchical arbiter integrating various priority arbitration modules. The proposed approach was integrated in a toolbox based environment.

Key words: SoC, adaptation bridge, communication synthesis, arbiter synthesis

INTRODUCTION

The systems on chip (SoC) equip more and more systems in fields as varied as general public electronics or communications. These systems are carried out starting from preset blocks (IP: Intellectual Property) that allow the reuse of the simple blocks (memory, etc) and complexes blocks (DSP, DMA, ASIP, etc). There is already thinking about designing reusable SoC in their turn, comprising a reconfigurable interconnected network and a RISC processor. The reconfiguration technique is appropriate for a series of applications that could be classified in the same field (image processing, telecommunication, etc). Because of important fall of cost realized thanks to the efforts made at the hardware level, an additional cost is added for the software part of these applications reaching today about 80% of the total cost of development[1]. To reduce in a significant way the software costs, the manufacturers of SoC generally choose to use the hierarchical standard buses (AMBA, STbus, etc). The objective is to be able to use owner's solutions, although IPs are not always compatible. Another solution consists in using the networks on chip (NoC) which are completely reconfigurable and which are connected with programmable circuits with a very important rate of integration. The use of SoC is also justified by the great availability of IPs components from a large number of manufacturers such as ARM, Hitachi, MIPS, etc. These companies propose RISC processors in the form of IP with performances which vary according to the application to realize. ARM preceded its principal competitors while selling more than 400 million RISC processors for embarked systems only for the year 2000[2].

The establishment of a solution of communication in a SoC is a crucial task which must take account of the delays due to the communications. It should be noted that these deadlines are increasingly weak taking into account the fact that technology is more and more accurate. The principal manufacturers of SoC propose standard buses which are used and recognized by the majority of the IPs. The choice of architecture of adequate communication then poses a broad range of problems which consist of optimizing several criteria at the same time.

Our objective is to develop an extensible model and an interactive approach of synthesis of communications architectures containing multi-bus bridges. The basic tasks of this approach consist of the synthesis of adapters of the communications protocols, the synthesis of arbitration and the generation of a parameterised multi-bus bridge.

Related works: The work undertaken by several teams, researchers and industry, referring to the integration of the communication within a SoC focused on the study of the new resources for the design of the SoC. New technologies appeared and various types of communication are studied. These resources from now on are among the elements influencing the performances of the SoC. The majority of these works concentrated on the exploration of the solutions space[3-5] known as optimal to lead sometimes to a system which can include more than one bus. The components having then the most affinities at communication regards will be placed on the same bus. The various buses will be generally connected between them by a set of bridges. These bridges will be given the responsibility to make the adaptation (two by two), according to needs for the application to realize.

A comparison between various architectures of SoC equipped with a guide of selection of an architecture given according to the application to establish was presented[6]. Liang proposes an infrastructure of adaptive communication of a SoC which can be reconfigured on the application level[7]. New high performance architecture for the design of the

Corresponding Author : Abdelkrim Zitouni, Laboratoire EμE, Faculté des Sciences de Monastir, 5019 Monastir, Tunisie

SoC (LottryBus) was presented[8]. This architecture is equipped with an arbiter (Lottry Manager) which allows the assignment of the priorities to the askers while basing itself on a law of probability which takes account of the history of communication for each requester performance of this technique was compared with a set of traditional techniques of arbitration (Fixed, TDMA, etc) for various classes of applications. The weak point of this technique is that it does not consider the problem of arbitration for a multi-bus system. The Sonics Company[9] proposed a new generation of bus Silicon Backplane using micro-networks. This technology consists in connecting several components through modules of interface to the micro-network. The limit of this new generation of bus is that surface used is more important than that used by a bus. As the performances of micro-networks although they are better as those of a bus are not always enough to meet the needs for certain applications.

The presented approaches do not allow the exploration of the arbitration space. They generally choose a given mechanism of arbitration or in extreme cases leave the choice with the designer to fix the mode of priority which adapts best to its application. Moreover, the approaches suggested are not addressed to the automation of the phase of generation and synthesis of a single bridge in the case of a multi-bus architecture. They use a structure of distributed bridges by affecting a bridge for each couple of buses. In this study we focused on architecture of centralized bridge. The selection of the modules to be used for the adaptation and the arbitration is done starting from a parameterised communication library. This library makes it possible to mask the communications protocols details from the user. The bus arbiter present inside the bridge allows to manage the communications between two components belonging to the same bus (internal communication) and even allow exchanges between components belonging to different buses while passing by the bridge (external communication). In order to minimize the latency time, we propose a hierarchical arbiter by associating a level of arbitration each level of communication (internal or external). In the other approaches, the external communication is done in two stages. The initiator of the communication or master takes the control of its bus and reaches the bridge behaving then like a slave. The bridge then formulates a request to obtain the access to the bus connected to the slave and thus acts as a master on this bus. The priority assignment in our approach is based on a cost function to be minimized and can lead to a mixed arbitration using various priority modes.

Synthesis of communication architecture for SOC: The use of the bridges in a SoC is necessary when the adaptation between several sub-networks communication must be operated. This established fact arises when the architecture of communication is treated on a hierarchical basis in order[10]:

* To offer the necessary performances locally;
* To accentuate parallelism by using concurrent resources of transport;
* To make defer the problems of incompatibility of protocols of the various sets of processor/bus, on single composing and allow as much as possible the use of the native protocols.

In this work we assume that we have a set of buses and that the processors of same affinities in terms of communication were allocated on the same buses. This phase can be made by an approach of architectural exploration such as[3]. The problem of synthesis can be formulated as a problem of generation of a centralized bridge containing structures of adaptation and levels of arbitration. This bridge will make it possible to adapt two resources of communication belonging to the same bus or different buses but guaranteeing a given bandwidth.

Bridge model: The model of the bridge proposed can be regarded as an application specific of the adapters to the connection of several buses supporting each one several masters. It allows two essential tasks which are arbitration and adaptation. To ensure a communication, the initiator sends a request and addresses to the component with which it wants to communicate. By a mechanism of address decoding and priority management inside the bridge, the target components will be identified. If the target belongs to the same bus with the master it will be managed by an internal level of arbitration associated with the bus itself. If it belongs to a bus other than that of master it will be managed by a level of external arbitration associated with the various buses of architecture. The data exchanged by a master on its bus follow a protocol of exchange which can not be supported by the bus connected to the targeted slave. The bridge then allows the adaptation of these two protocols.

To uncouple the synchrony from the various buses we call upon mechanisms of buffer memory (FIFO). The costs and the performances relating to these components rise from the implementations of the finite states machines carrying out the various protocols and managing these memories. These components must support the bandwidths suitable for each bus adapted in order to optimize the use of these shared resources of communication. FIFOs being dimensioned to deal with a small number of packets are small. As for the adapters, when the protocols to be adapted are not more usual, it is necessary to develop specific models to the considered pairs of bus. These components answer a quite precise specification: to adapt two resources of communication supporting a given bandwidth.

If we take a system which, after study of the space of solutions, led to the use of two different buses only, then it will be necessary to adopt the architecture of Fig. 1. The exchanges are done while passing by two FIFOs (FIFO I and FIFO O) allow the management of the transfers between buses with different speeds.

In addition to the units of adaptation, arbitration levels, address decoder and FIFOs, the bridge includes a control unit of communication and a set of multiplexers. Figure 2 presents the model which will be used to control the communication in the case of three buses and to define the dataflow direction according to whether the operation is a reading or a writing. This task is assured by the control unit. This unit must command at the same time the adequate modules of adaptation for the selected operation.

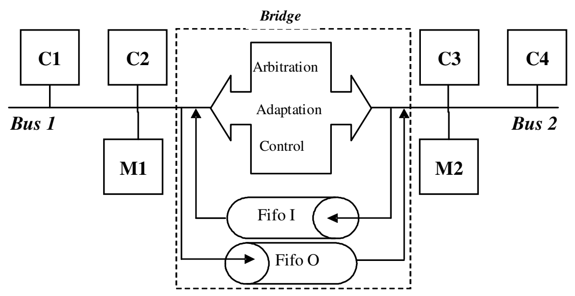


Fig. 1: Structure of a bridge for a system with two buses

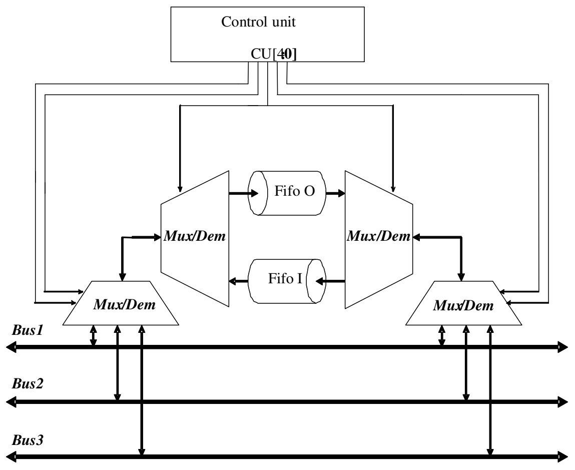


Fig. 2: Control of data path in the case of three buses

Adapter's synthesis: Our approach consists in realizing, starting from the non-coherent interfaces, models of adapters for each couple of bus on the level of width, control signals and speed. The synthesis method is based on models of interfaces for buses which are stored in a library envisaged to this end. This library contains various models of bus with the necessary modes which are sometimes specific for them.

The design of an adaptation module starts with the study of the temporal specifications of the protocols for each couple of bus. These specifications contain the main part of information which will lead to validate the model to be conceived with knowing the relations of causality as well as the temporal constraints between the different signals. The next step consists in extracting the respective finite states machines (FSM) of these buses for the chosen mode, from the temporal specifications. The finite states machines approach is used for modelling sequential protocols. For a complex protocol which can be the seat of several states which are active simultaneously, the adapted model is the extended finite states machines or EFSM[11]. By amalgamation of the two FSM corresponding to the two incompatible protocols then we generate the FSM of the adapter. The technique of amalgamation of the FSM was presented[12]. Finally, the FSM of the adapter will be transformed into a VHDL description on RTL level so that it can be synthesized by the commercial tools.

Figure 3a shows result of simulation of the adaptation between the PCI/AMBA adapter in read mode. Figure 3b shows result of simulation of the PI-BUS/AMBA adapter in write mode. The signals master_size and master_items are combined to maintain the signal initially called LOCK in the AMBA interface model and this as long as the Count signal did not reach zero value, to be compatible with the sending of the burst type supported by these two models of bus.

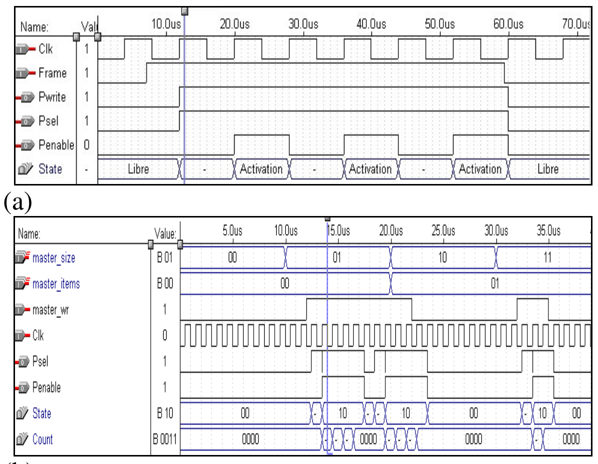


Fig. 3: Simulation results of (a): PCI/AMBA buses in read mode and (b): PI-BUS/AMBA buses in write mode

The addressing of the components, IP blocks or memories present within architecture is done by an address bus. The selection of a component present on a bus is done while placing an address included in a quite precise range, which was affected to it at first. Each component will have a range of addresses by which one will be able to address it.

FIFO dimensioning: The data path is primarily made of FIFOs. This mechanism allows the adaptation of buses having different periods of transfers, different widths buses or both at the same time. For example, for a bus of 16 bits which communicates with another of 8 bits width, the transfer will be done in two times (two bytes).

The buses used can produce data in burst mode. The data storage in an intermediate buffer can appear a better solution than to make function the whole of the system at the frequency of the slowest bus. The data can thus be transferred by using a FIFO with two possibilities of protocols (blocking or non-blocking). The blocking protocol uses the signals Full and Empty for synchronization. The non-blocking protocol does not use any signal to announce the current state of the FIFO. Such a protocol is used when the specifications of the consumer and the producer guarantee that none of both can overload the other.

The width in bits and the depth in words of the FIFO must be dimensioned for a minimal loss of data[13]. In order to reduce the number of transfers for a communication between the two buses, the width of the queue bwQ is given by: bwQ = max [bwbus1, bwbus2] with bwbus1 and bwbus2 respectively represent the width of bus1 and bus2. The depth of the queue Qn must be given in a manner to minimize its size by the formula: Qn = max (0, Qn-1 + (Pn – Cn)) where Pn and Cn represent the quantity of data respectively produced and consumed at moment n.

Bridge operating mode: The activation of the various modules present in the adapter is controlled by the couple arbiter/decoder of addresses. When the adapter is not requested, it is in the Idle state (Free), it does not activate any of the modules. It waits then for Reset signal or a request for exchange coming from one of the applicants, the target of the communication is identified by the address variable Dest and the state passes then in a state where the applicant and the target are both activated. The control unit of the bridge communicates with the modules of the adapter via two lines of acknowledge at 4-phases (Req, Ack). Once selected, a module awaits an event on the Req line and carries out the control of communication. Data are transferred via FIFOs. These FIFOs are controlled, at the wished moment, by the control signals. After having finished the control of communication, the selected module activates the Ack line. As long as they are not selected, the other modules put the Ack line in high impedance in order to protect this line from a possible conflict. Having received the response via the Ack line, the control unit de-asserts the Req line and waits until the Ack line is de-asserted, so that it de-asserts the function in course of operation. Finally, the control unit waits until the Grant lines are de-asserted. Indeed, the control unit will be activated only when it sees an event on one of the Grant lines generated by the arbiter indicating that a master gained the access to system bus. The decision to select a block or another is done by the

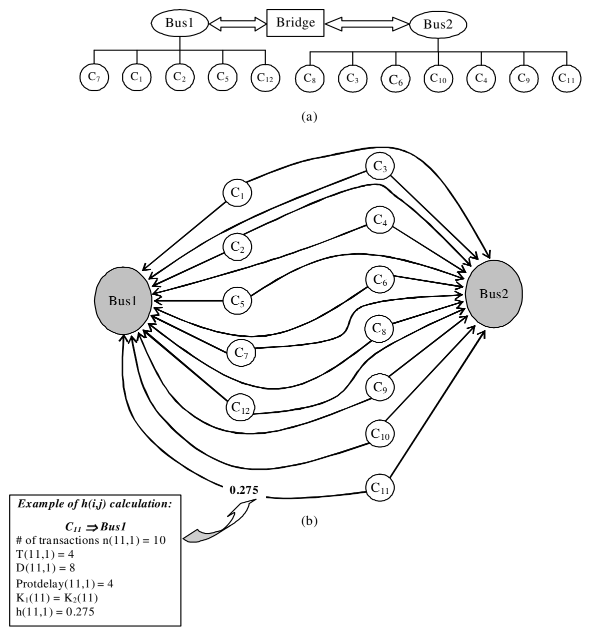

Bus1
Bridge
Bus2
C7 C1 C2 C5 C12 C8 C3 C6 C10 C4 C9 C11
(a)
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12
Bus1
Bus2
0.275
(b)
Example of h(i,j) calculation:
C11 ⇛ Bus1
# of transactions n(11,1) = 10
T(11,1) = 4
D(11,1) = 8
Protdelay(11,1) = 4
K1(11) = K2(11)
h(11,1) = 0.275

Fig. 4: (b) CG associated to the hypothetic system presented in (a)

function of control following the address transmitted to the bus by the Master who gained the access. When an address is not defined in the model of the decoder of addresses, an Illegal_Address signal is positioned and could be used for a treatment of error. For any other value of address there is positioning CS to 1 corresponding to the specific range of the addressed module.

During the activation of the Read burst block or Write burst block, a signal Mask-Reqs is sent by the Master of bus to the module of arbitration so that it can answer no applicant of bus during all the duration of the transfer. This signal will be de-asserted as soon as the transfer in progress is finished and the bus was put in high impedance.

ARBITER SYNTHESIS

The need for an effective arbitration is one of the most important constraints in the communication synthesis for SoC. It is necessary to synthesize arbiters who allow as well as possible to manage the requests for access to the various buses in order to optimize the latency times between two successive accesses. We presented a technique which allows the assignment of the priorities to the askers, in a manner which minimizes the latency time between two successive occupations of the bus[14]. This time represents wasted average time so that a component gains the access to the bus. The orders of the priorities are assigned while being based on a cost function, combination of two metric, to minimize.

```
Algorithm Priority Assignment
  { /* Internal communication*/
  For each system bus busj Do {
      read user set parameters /* $K_1(j)$, $K_2(j)$ */
      For each component $C_i$ which is allocated to busj Do {
          read the metrics associated with $C_i$ /* $f_{access}(j,i)$, $F_{trans}(j,i)$*/
          $Internal\_H\_List(j,i) = K_1(j)\times f_{access}(j,i) + K_2(j)\times F_{trans}(j,i)$ }
          ASSIGN_PRIORITY(Internal_H_List(j), Internal_Priority_List(j))
          GENERATE_ARBITER_BUSj(Internal_Priority_List(j), Internal_VHDL_Description) }
  /* External communication*/
  For both bus1 and bus2 Do {
      read user set parameters /* $K_1(1)$, $K_2(1)$, $K_1(2)$, $K_2(2)$*/
      For each component $C_i$ of system components Do {
        If $C_i$ is allocated to the bus1 Then {
            read the metrics associated with $C_i$ relatively to bus2 /* $f_{access}(2,i)$, $F_{trans}(2,i)$*/
            $External\_H\_List(i) = K_1(2)\times f_{access}(2,i) + K_2(2)\times F_{trans}(2,i)$ }
        Elsif $C_i$ is allocated to the bus2 Then {
            read the metrics associated with $C_i$ relatively to bus1 /* $f_{access}(1,i)$, $F_{trans}(1,i)$*/
            $External\_H\_List(i) = K_1(1)\times f_{access}(1,i) + K_2(1)\times F_{trans}(1,i)$ }}
  ASSIGN_PRIORITY(External_H_List, External_Priority_List)
  GENERATE_ARBITER_BUS_1_2(External_Priority_List, External_VHDL_Description) }
  GENERATE_MULTIBUS_ARBITER(INTERNAL_VHDL_Description, External_VHDL_Description, Multibus__VHDL_Description)}
```

Fig. 5: Algorithm of arbitration synthesis

The first metric is based on the shared frequency of access of the requester to the bus. The second metric is based on the size of the data to transfer by a requester through the bus during all the lifetime of the system. The limitation of this technique lies in the fact that we treated the case of only one bus by supposing that the Masters allocated with this last have different sets of priorities. This assumption enabled us to generate only an arbiter in fixed priority mode.

In this work we propose the extension of this technique in the case of a SoC by taking account of the internal communications within the same bus as well as of the external communications between two components belonging to different buses. Another extension relates to the generalization of the modes of the priorities is also made. Indeed, a requester can have more priority to reach a bus than another and at the same time less priority to reach another bus. A set of askers can have as the same set of priorities to reach a bus or another, but to profit from a priority different for another group of askers, etc. The arbiter to be developed must be hierarchical, integrating various modes of arbitration which are configurable according to each bus.

Algorithm of arbitration synthesis: The proposed algorithm (Fig. 5) in the case of two buses starts while having a library of arbitration which gathers the most used mechanisms (Fixed, Round-Robin, Daisy-Chain, TDMA, First Came First Granted, etc). The second parameter is a Communication Graph (CG) which brings various information's necessary for the communication of each component with each bus.

The basic task of this algorithm consists in the generation of an RTL description of a hierarchical arbiter to be integrated in the bridge. The assignment of the sets of priorities is based on a generalized performance index. The richness of the library also makes it possible to the user to choose a mode of well defined priority without taking account of the cost function.

A CG is a direct graph made of nodes and arcs. A node is associated with each component and an arc (Ci, Busj) is directed from the component to the bus. This arc utilizes the parameters of communication between component Ci and the busj. These parameters are used to calculate the cost function $H(I, J) = K1(J) \times faccess(I, J) + K2(J) \times Ftrans(I, J)$. The metric $faccess(I, J) = 1/T(I, J)$ represents the access frequency of component Ci to the busj. (T(I, J) represents the interval of time which separates two successive accesses). Metric Ftrans(I, J) = (1/N(I, J)) × ftrans(I, J) takes account of the size of the data to transfer. In this expression, N(I, J) and ftrans(I, J) respectively represent the number of transfers of the data D(I, J) by component Ci through the busj and the frequency of a transfer (ftrans (I, J) = 1/TempProt (I, J)). TempProt (I, J) represents the total time employed during only one transfer. The two parameters K1(J) and K2(J) representing the weights of the function are fixed by the designer in order to privilege one of the two metric ones for the busj. For each bus, the highest priority is assigned with the component which have the greatest value of the function H(I, J). The components which have the same value of the function H(I, J) have the same set of priorities. The Fig. 4b, shows an example of a communication graph for the hypothetical architecture of the Fig. 4a.

In the case of the internal communication, the algorithm carries out the stage of assignment of priority for each busj and arranges the values in a list (Internal_H_List(j)). The ASSIGN_PRIORITY procedure schedules this list in an order ascending and generates a list of priority (Internal_Priority_List(j)). After having fixed the sets of priorities, another procedure GENERATE_ARBITER_BUSj) will be activated in order to generate a synthesisable VHDL description of the arbiter for each bus (INTERNAL_VHDL_Description).

In the case of the external communication, the algorithm carries out the stage of assignment of priority for the various buses and arranges the values in a list (External_H_List). Assignment of the sets of priorities and the generation of VHDL description (External_VHDL_Description) are realized by the same procedures used in the internal communication.

The instantiation of the various modules of arbitration is carried out by the GENERATE_MULTIBUS_ARBITER procedure. The output of this procedure is a VHDL description Multibus_VHDL_Description which will be integrated in the communication bridge. The details of VHDL descriptions of different sub-modules from arbitration as well as the technique of RTL synthesis are presented[12]. The method of instantiation in the case of system of Fig. 4a is presented as follows.

Instantiation method of the arbitration modules: In the case of a multi-bus system, two possibilities of communication are possible for a given initiator. Either that the target is on the same bus with the initiator, or on a different bus. In the first case (internal communication), it is the module of arbitration associated with the bus in question which is given the responsibility to manage the priorities of the various components on this same bus. What improves the performances of the system since some of the transfers will be done concurrently. The arbiter who includes the modules of arbitration associated with each bus is called internal level of arbitration. As for the second case (external communication), it is the external level of arbitration which will manage the communication by checking the site of the component emplacement. In the suggested example (SoC with two buses), we suppose that the assignments cost functions are ordered by the arbitration synthesis algorithm like presented in Fig. 6.

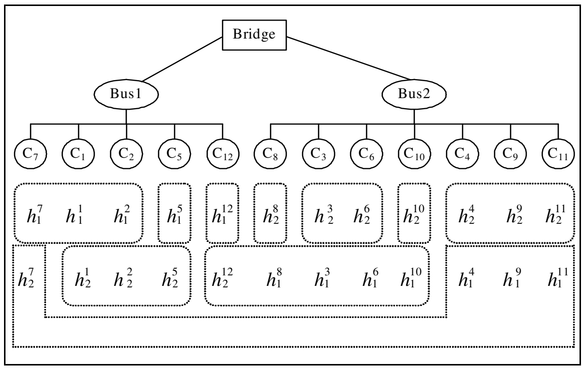

Bridge
Bus1
Bus2
C_7 C_1 C_2 C_5 C_{12} C_8 C_3 C_6 C_{10} C_4 C_9 C_{11}
h_1^7 h_1^1 h_1^2 h_1^5 h_1^{12} h_2^8 h_2^3 h_2^6 h_2^{10} h_2^4 h_2^9 h_2^{11}
h_2^7 h_2^1 h_2^2 h_2^5 h_2^{12} h_1^8 h_1^3 h_1^6 h_1^{10} h_1^4 h_1^9 h_1^{11}

Fig. 6: Scheduling of the sets of priorities in the case of an example of SoC with two buses

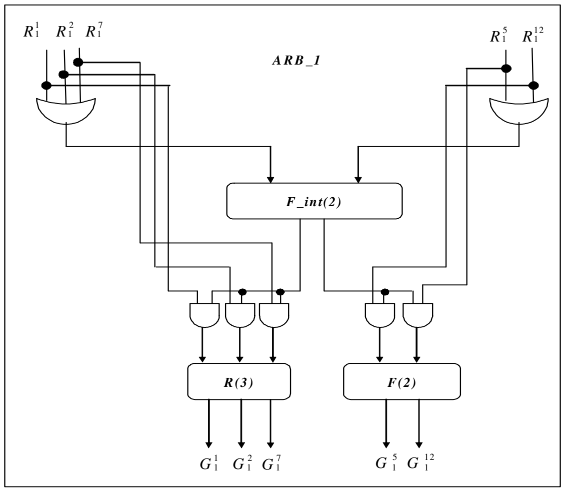


Fig. 7: Arbitration modules for internal communication with bus1

For the components allocated with the bus1, let us suppose that ($h_1^7 = h_1^1 = h_1^2$) > h_1^5 > h_1^{12} ; where h_1^i represents the cost function associated with component Ci. From this assignment we notice that the applicants (1, 2, 7) have the same set of priorities to access bus1. A module of arbitration (R(3) in Fig. 7) in round-robin priority mode or TDMA mode with identical temporal window must be designed. Indeed these modes are generally reserved to the applicants of equal priorities. Each applicant associated with this module must have a set of priorities higher than the component 5 which has a priority higher than component 12. Therefore, applicants 5 and 12 are grouped together and will be used in fixed priority mode for two applicants (F(2) in Fig. 7). Indeed, this mode is generally reserved to the applicants of different priorities. As long as applicants 1, 2 and 7 have a set of priorities which is higher than that of applicants 5 and 12, each applicant of the module F(2) is only served if there is no request deposited with the module R(3). This is assured by the fixed priority arbiter with two requesters (F_int(2) in Fig. 7). In this figure, R_1^i and G_1^i respectively represent the signal of request for bus deposited by component Ci to the arbiter and the signal of acknowledge sent by this arbiter to component Ci.

For the components allocated with the bus2, we have $h_2^{10} > (h_2^3 = h_2^6) > (h_2^4 = h_2^9 = h_2^{11}) > h_2^8$. The same procedure as in the case of the communication through the bus1 is to be followed for the development of the module of arbitration. We notice that we cannot group the applicants which have different sets of priorities but non consecutive in the same module at fixed priority (e.g. applicants 10 and 8). The two modules of arbitration corresponding to bus 1 and 2 constitute the internal level of arbitration and can be activated simultaneously in the case of an internal communication.

In the case of an external communication, the transfer of the data and control is ensured through the bridge. To communicate with an external bus, a given component must gain the access to its local bus and the external bus. Indeed, we can encounter the case where a component allocated with the bus1 wants to communicate with a component allocated with the bus2 and at the same time another component allocated to the bus2 wants to communicate with another component allocated to bus1. What we call cross communication. Thus the components allocated with the bus1 and the bus2 must be managed by the same level of arbitration like it was envisaged by the algorithm of arbitration. In the suggested example, let us suppose for example that $(h_2^7 = h_1^4 = h_1^9 = h_1^{11}) > (h_2^1 = h_2^2 = h_2^5) > (h_1^3 = h_1^6 = h_1^8 = h_1^{10} = h_2^{12})$. It would be necessary to follow the same steps as in the case of the internal communication to generate the modules of arbitration corresponding at the external level of arbitration.

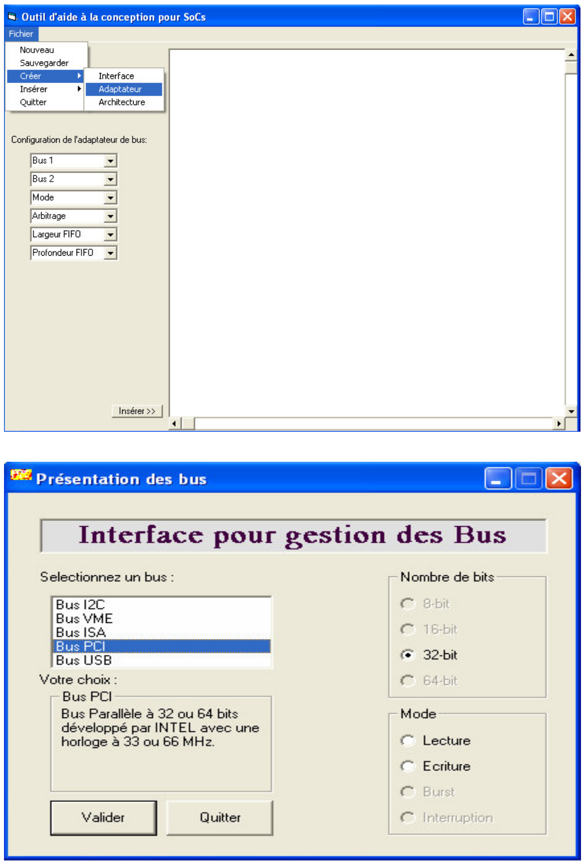
For the instantiation of the various levels of arbitration, we suppose that the priority to reach an internal bus is higher than the priority giving access to an external bus. This is obvious as long as the applicants who frequently communicate are allocated within same buses[4]. For this reason, the external level of arbitration will not be activated that if there is no request deposited at the internal level of arbitration. This is done by the same principle as in the case of instantiation of sub-modules of a given module of the internal level of arbitration. The modules of the internal level of arbitration are not instantiated as long as they can function in a concurrent way.

The generated arbiter is hierarchical. The modules of arbitration (internal and external) constitute the state machines and represent the first level of hierarchy. Sub-modules of arbitration of each machine constitute the macro states and represent the second level of hierarchy. The states of each macro-state constitute the states on RTL level and represent the lowest level of the hierarchy.

AUTOMATION OF THE BRIDGE GENERATION TASK

The proposed approach consists of the use of primitives making it possible successively to generate the elements of a bridge of communication, to arrive at a realization. This approach was integrated in a toolbox based environment[14]. Being given that the elementary units are modelled on RTL level, the environment allows the generation of a synthesisable description by the existing tools. This environment can be used with other tools for performance analysis, partitioning, allocation, architectural synthesis, etc. For a designer, the availability of an environment of the toolbox type with a set of primitives of communication synthesis enables him to concentrate only on fixing the parameters related to its application.

The extension which one brought to the environment interests in the management of the library and the synthesis of the bridge of adaptation. The management of the library is based on graphic concepts. The synthesis of the bridge is done by successive stages allowing the generation and the instantiation of the various modules until obtaining a synthesisable description. Each module of the library is independent and has in addition to its specification, other information such as the types of data, the generic parameters, its name, its interface, etc. The types of data are those of the internal elements to description, including the inputs and outputs of the component. The generic parameters are those quoted in the interface. This last includes the list of the generic internal parameters, the inputs/outputs of the component and their nature (in, out, in/out). The generation of a module of adaptation is carried out according to criteria such as the sizes of bus, the nature of the modes of communication, etc. These parameters also constitute an essential part of the library. The parameters related to each protocol are put in a file which will be accessible only by the manager in order to generate the required module of adaptation. The Fig. 8b presents the interface which allows the fixing of the choice of the selected bus like these parameters. The use of the chart of a protocol in the developed environment allows the re-use of the protocols for several projects and improves the manner of exploitation of a module of adaptation in a new project. This makes it possible to



Bus Parallèle à 32 ou 64 bits développé par INTEL avec une horloge à 33 ou 66 MHz.

Fig. 8: User interfaces: (a) synthesis of bridge and (b) management of the library

the designer to handle graphic symbols to carry out the bridge of communication. The selected component appears with its symbol, which makes it possible to position it on the sheet of edition. Once the component placed, it can be adapted and connected to the other components. The types of relations and generic parameters (width of a bus, depth of a FIFO, etc) are read and integrated in the description of the bridge. Each element being a repertory and management is carried out by handling of files. The Fig. 8a presents the user interface which makes it possible to choose the buses to be adapted and the mode of transfer to create the adapter. The right-hand side zone will accommodate the various components in their graphic form, of architecture and thus will be able to connect them the ones to the others until drawing up the final diagram.

CONCLUSION

In the proposed approach we studied and developed a model and a toolbox based environment of communication bridge synthesis. This bridge includes a set of elementary modules of adaptations, a hierarchical arbiter, an addresses decoder and a data path consisting of FIFOs and of multiplexers/demultiplexers.

The various problems concerning the dimensioning of FIFOs for the adaptation in width and frequency and the minimization of the latency time while being based on a cost function were studied. This study enabled us to generate a hierarchical arbiter allowing various alternatives of arbitration.

The model of architecture preached by our work seems to be a happy medium between multi-bus architectures and the NOC. Indeed, thanks to a preliminary phase of exploration, once multi-bus architecture is defined, we adopted a step to adapt interfaces of the buses brought into play. Those carrying the components sharing the most affinities. The advantage of such a step is that one benefits from the small overall dimensions which one finds in the architectures built around a bus and speed of the exchanges specific to architectures containing NOC. Another advantage of this approach is the possibility of having simultaneous exchanges between components present on the same bus what will be able appreciably to improve the performances of the target system. Owing to the fact that each bus has its level of arbitration in the bridge, the communications between components present on the same bus can be organized in a completely independent way, except if resources are awaited to continue a transaction on the other bus.

Our work could be fully exploited in the high level design of the communication in the SoC. It will be necessary for that to take care of the design of other models of adapters and interfaces which will come to enrich those already carried out. Other work of communication architectures synthesis containing a higher number of buses is much more delicate to treat with our approach. It is then necessary to have recourse to the NOC as solution for this problem and thus to develop the corresponding models of communication.

REFERENCES

1. Auguin, M., L. Capella, F. Cuesta and E. Gesset, 2001. CODEF: A System Level Exploration Tool. ICASSP, Salt Lake City.
2. Auguin, M., 2004. Conception de Systèmes sur Puce : Nécessité d'Approches Globales Face à la Concentration des Difficultés. 8ème Symposium en Architecture, Sympa8 Hammamet, pp : 291-298.
3. Poujet, J., 2002. Test des Systèmes sur Puce: Ordonnancement et Exploration de l'Espace des Solutions Architecturales. Ph.D. Thesis, Université de Montpellier II : Sciences et Techniques du Languedoc.
4. Lahiri, K., A. Raghunathan and S. Dey, 1999. Fast Performance Analysis of Bus-Based System on Chip Communication Architectures. ICCAD, pp: 566-573.

5. Lahiri, K., A. Raghunathan, S. Dey and G. Lakshminarayana, 2000. Communication architecture tuners: A methodology for the design of high performance communication architectures for system on chips. Proc. 37th Design Automation Conference (DAC), Los Angeles.
6. Ryu, K., E. Shin and V.J. Mooney, 2001. A comparison of five different multiprocessor SoC bus architectures. Euromicro Symposium on Digital Systems Designs, Poland, pp: 202-209.
7. Liang, J., S. Swaminathan and R. Tessier, 2000. A SoC: A scalable, single-chip communication architecture. Intl. Conf. Parallel Architecture and Compilation Techniques, Philadelphia USA, pp: 37-46.
8. Lahiri, K., A. Raghunathan and G. Lakshminaraya, 2001. LOTTERYBUS: A New High-Performance Communication Architecture for System on Chip designs. Proc. DAC, Las Vegas, Nevada, USA.
9. http://www.sonicscorp.com
10. Nicolescu, E.G., 2002. Spécification et Validation des Systèmes Hétérogènes Embarqués. Ph.D. Thesis, INPG, Grenoble, France.
11. Harel, D. *et al.*, 1990. Statemate: A working environment for the development of complex reactive systems. IEEE Trans. Software Engineering, 16 : 403-413.
12. Zitouni, A., 2001. Synthèse de Communication pour les Systèmes Distribués dans le Cadre du Co-Design. Ph.D. Thesis Faculty of Sciences of Monastir, Tunisie.
13. Shin, D. and D. Gajski, 2002. Queue Generation Algorithm for Interface Synthesis. Tech. Rep. CECS-02-12. University of California, Irvine, USA.
14. Zitouni, A., M. Abid, K. Torki and R. Tourki, 2002. Communication synthesis techniques for multiprocessor systems. Int. J. Electron., 89: 55-76.