SoCBUS: The solution of high communication bandwidth on chip and short TTM

Dake Liu, Daniel Wiklund, Erik Svensson, Olle Seger, and Sumant Sathe
Computer engineering, Linköping University
58183, Linköping, Sweden, email: {dake, danwi, sumant}@isy.liu.se

Abstract – The paper is divided into two parts. The first part of the paper gives the motivation to use SoC bus for system integration of a SoC as well as a brief review of the SoC concept, IP reuse, and research on SoC bus and SoC interconnect network. We define the SoC network as "classical SoC network" that delivers data packets on a network. We point out in the paper that the classical SoC network introduces long data latency and contributes to high silicon cost. In the second part of the paper, we introduce the SoC bus research from Linköping University. We introduce a novel concept, *PCC*: Packet Connected Circuit. Based on *PCC*, a data transaction is initialized by packet routing. The rout is locked as a bus circuit after proving and acknowledging the rout. Therefore, we reach fixed and low transfer latency based on high bandwidth supporting multiple simultaneous data transfer. At the same time, buffers in the nodes are eliminated so that the silicon cost of a SoC network becomes feasible. *MESA* and *MISA* models are proposed for improving wrappers. A SoC system designer's methodology is proposed so that a short TTM (Time To Market) SoC design is possible.

1. Introduction

Many challenges exist in SoC designs. In this paper, we discuss two of them. The first challenge is the on chip communication with high bandwidth and high flexibility. High bandwidth here means supporting multiple accesses simultaneously based on high data rate and low latency. High flexibility here means open and scalable during system design time and dynamically configurable during the system run time. The second challenge is to design an ultra complex system within limited time (short TTM). Design for short TTM is based on reuse methodology or IP (Intellectual Property) based design; IP at the silicon level and the firmware level will be used as the basic components in the design. The basic problem in IP based design is to eliminate software and hardware glues between SIP (Silicon Intellectual Property), memories, and other functional building blocks. Researchers have proposed a solution to eliminate glue logic between IP by introducing another IP named SoC network. The SoC network is used to configure and converge the custom port to an on chip network standard. The ultimate goal is to plug IP instead of designing for IP connections.

We review the current SoC network research and introduce our research. In the second part, we give the background of IP and SoC design. The SoC network research is reviewed in the third part of the paper. We introduced and evaluated our research in detail in the fourth part. We finally describe the on going work and conclusion in parts five and six.

2. SoC, IP reuse, and SoC BUS in general

There is no clear definition for a SoC (System-on-a-Chip). The basic concept on SoC could be a kernel part of an embedded system. A SoC normally consists of a number of processors including DSP processors, micro controllers, and other ASIP (Application Specific Instruction Set Processor), memory blocks including scratch pad memories and caches, and custom ASIC (Application Specific Integrated Circuits) building blocks. An ASIC block could be a digital circuit or an analog or analog-digital-mixed circuit. A gateway chip-set as a SoC example is given in the following figure:

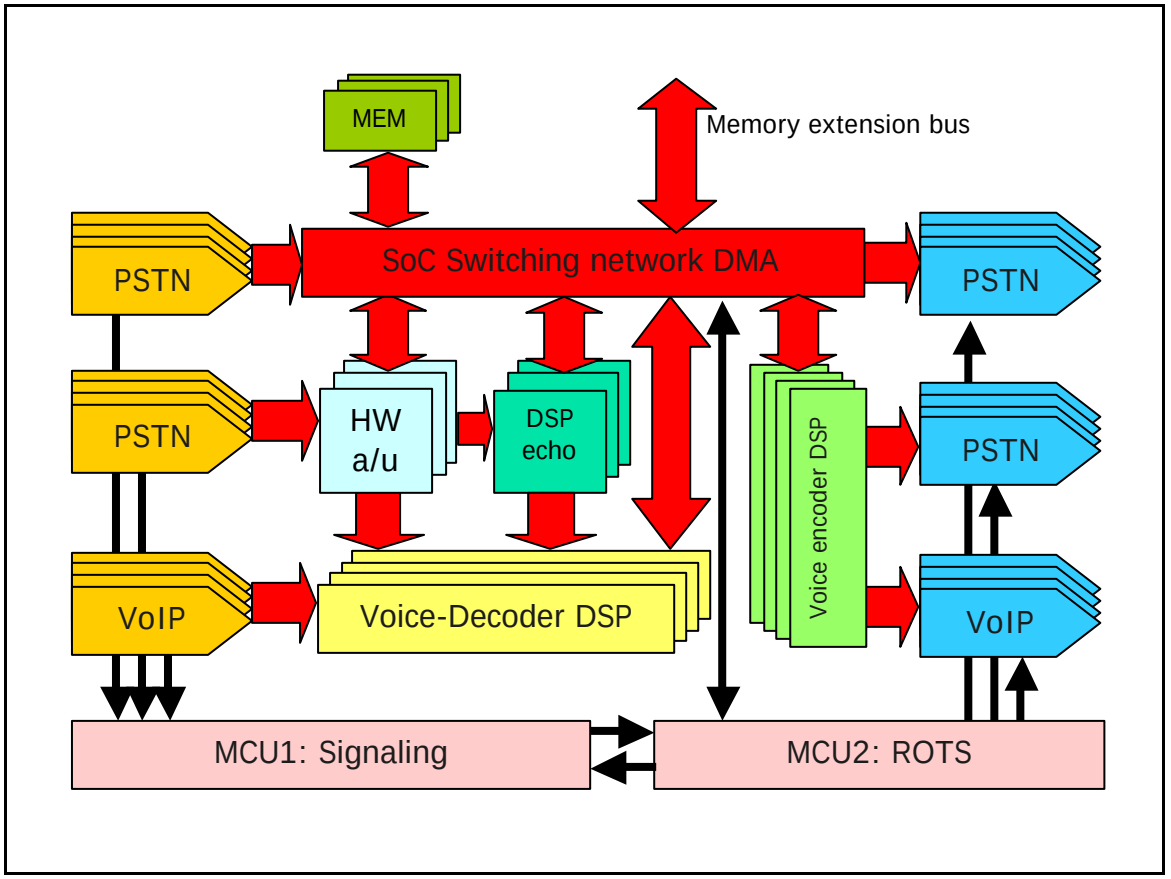


Figure 1. A voice over IP gateway SoC (one of our research project)

In the example above, the system supports PSTN (Public Switching Telephone Network) lines to PSTN lines; PSTN to Voice over IP with A–law / u–law converter, echo canceller with noise suppression, voice decompression for one subscriber, and voice compression for another subscriber; Voice over IP to PSTN or another voice over IP with both voice decompression and voice compression for different standards. The chip set consists of hardware building blocks, such as A-law/u-law encoder and decoder, and Ethernet PHY; DSP processors, such as echo cancellers, voice CODEC (coder decoder) including voice quality enhancement process; micro controllers, such as the controller for signaling and the controller supporting operating system; and memories. A SoC interconnect network is necessary for the dynamic connections including program loading, payload delivery, computing buffer passing, and controls. Another SoC example is the car integrated electronics system. Three buses are defined in the system, the IDB (), the proprietary bus, and the analog audio bus.

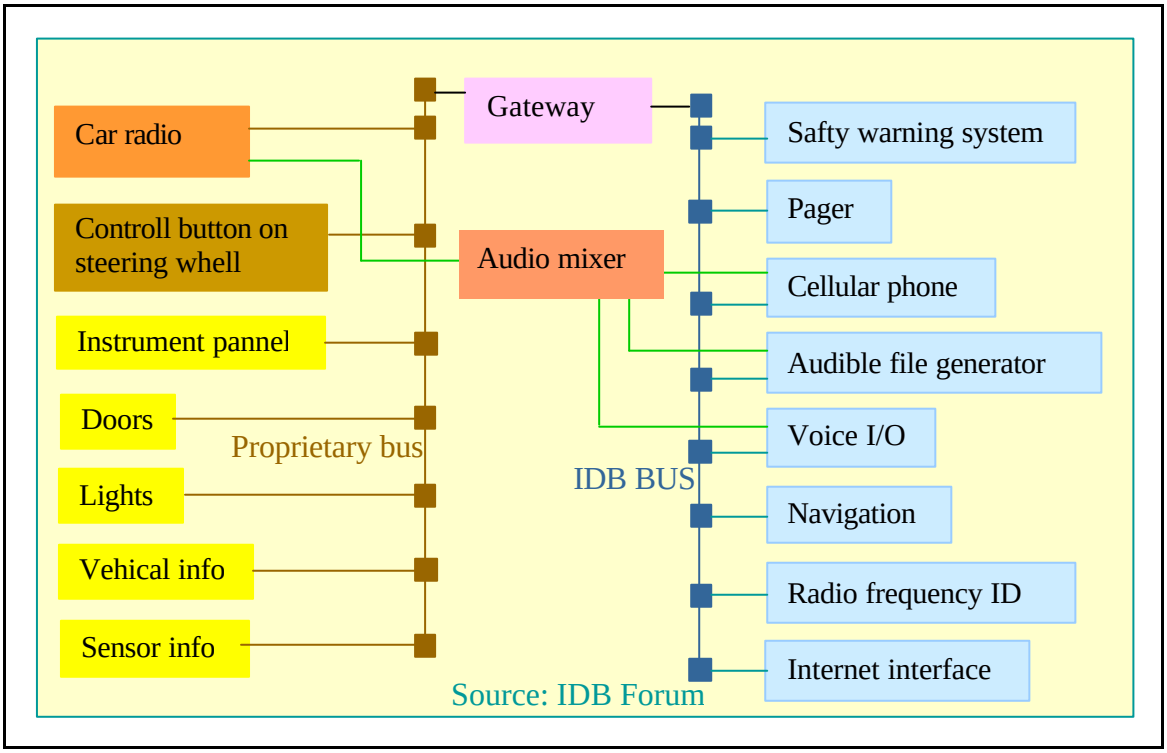


Figure 2. Another SoC example: Integrated Car Electronics

It is not realistic to design a SoC including all sub-systems in house. Too long design time and excessive knowledge needs to be collected. Design a SoC as an IC should be based on IP based design methodology (or reuse design methodology). IP based design was proposed during the middle of 90's thanks to the possibility supplied by the silicon technology. Different from RTL (Register Transfer Level) design, the IP based design releases the functional design and verification on IP level instead of on RTL. The basic concept is good because we can release the functional design and verification on higher level to save the design time. As soon as the IP based design was proposed, researchers and engineers found two problems. One is how to design qualified IP to support IP based design. Another is how to eliminate glues between IP.

Many designs as RTL codes or silicon layout files were available when the IP based design was proposed. Seldom, designs can be used as IP because of the low reusability. When we analyze and evaluate an available design for reuse, we found the legacy RTL code, the confusing specification documents or no supporting document, the not adaptable custom interface, no debugging environment or no qualified debugging environment, unusable firmware development tools, unusable synthesis and backend reference, and many more problems popping up. Engineers spent even more time to understand, adapt, and debug an available design than the time they can design the IP themselves. The conclusion was, an IP could never be just an available design. An IP must the available design based on regulated design methodology with qualified and readable code, enough debug supporting environments, enough firmware development environment, and even supports for integration from the IP vendor. Therefore, in later 90's, reuse design methodology was deeply investigated and developed. Design for reusable IP has been formally regulated for designing an IP [1] and SoC design experiences are accumulated both from research and in industries.

A SoC consists of many IP cores or blocks. To design a SoC embedded system, we needs to connect all hardware interfaces of IP cores and blocks together as well as integrate all interface related firmware together. During the integration or the design phase of a SoC, we generate functional hardware connecting IP and customize port protocol software and hardware interfaces. We therefore generate a lot of glue circuits and interface drivers. The verification of glue between cores and blocks is not easy. First, it is difficult to define how much IP function should be involved while verifying glue logic connecting IP ports. Second, A deep understanding of an IP core may be required to support the glue verification. Moreover, the deep knowledge of an IP core may not be available because of legal reasons. A classical way of verification for a SoC is based on running applications. We either need a test chip or an emulator; we therefore suffer from the long iteration time and high cost.

Custom glue logic design for inter IP connection gives another problem; the limited flexibility based on high throughputs. We may need a high connectivity from any port of an IP to any other port of another IP. Custom connection cannot give the flexibility required in the future. Only a knockout switch [2] can give enough flexibility and high throughputs, but no one will accept the high interconnection and fan-out fan-in cost when the number of ports is high. There is a proposed SoC bus based on TDM (Time Division Multiplexing) and arbitration, for example the VSI bus [3]. The problem is the one-D TDM bus cannot support multiple data transfers simultaneously so that the throughput is low.

Researchers realized the problem and are tring to find solutions supporting multiple data transfer simultaneously with low latency and reasonable cost, therefore defined the expected solution as a SoC interconnection network as a SoC bus [4], [5], [6], and [7]. Except for the high throughput, low latency, and reasonable cost, other basic expectations from the SoC interconnection network should be based on open and scalable structure during SoC design phase, flexible and configurable dynamic connection during the run time, easy to learn and easy to use, supporting design with short time to market and longer time in market. Researchers define this SoC on chip interconnection network as a kind of two-dimensional network.

The classical way of thinking for a SoC switch fabric is limited within OSI (Open System Interconnection), an open network concept. The main tasks or applications supported by OSI-layer based classical SoC network are backbone bus related applications.

The network proposed by most researchers is based on two-dimensional mesh of switches [4] to [10]. The source IP initializes a requirement packet; the requirement packet is sent to and delivered to the destination IP port according to the destination ID following routing algorithms; the destination IP decides to accept or reject the requirement packet and sends an acknowledgement packet; the source IP receives the acknowledgement packet and a data packet transfers start. Not many researches are reported related to routing algorithms. The network quality, capacity, bottlenecks, and possible deadlocks a not yet investigated in detail or not much reported from researchers.

A node is the essential component in the SoC network. Nodes connect nets and IP ports setting up the SoC network. A node could be based on four [10] or five ports

[6]. As packet based switching in a classical SoC network, buffers have to be built in a node. The buffer size defines the packet size including the packet header. The number of buffers in a node will be decided according to the capacitance analysis and requirements on performance. Around four buffers are necessary for every port so that there are about 16 to 20 buffers required in a node.

Wrappers are used in a SoC network to connect the SoC network to IP ports and off chip interface. A wrapper has its standard interface toward the network and a configurable interface to an IP port. Buffers and control circuits are required in a wrapper. SoC network given in the following figure is an example with 2D-mesh net and 12 nodes:

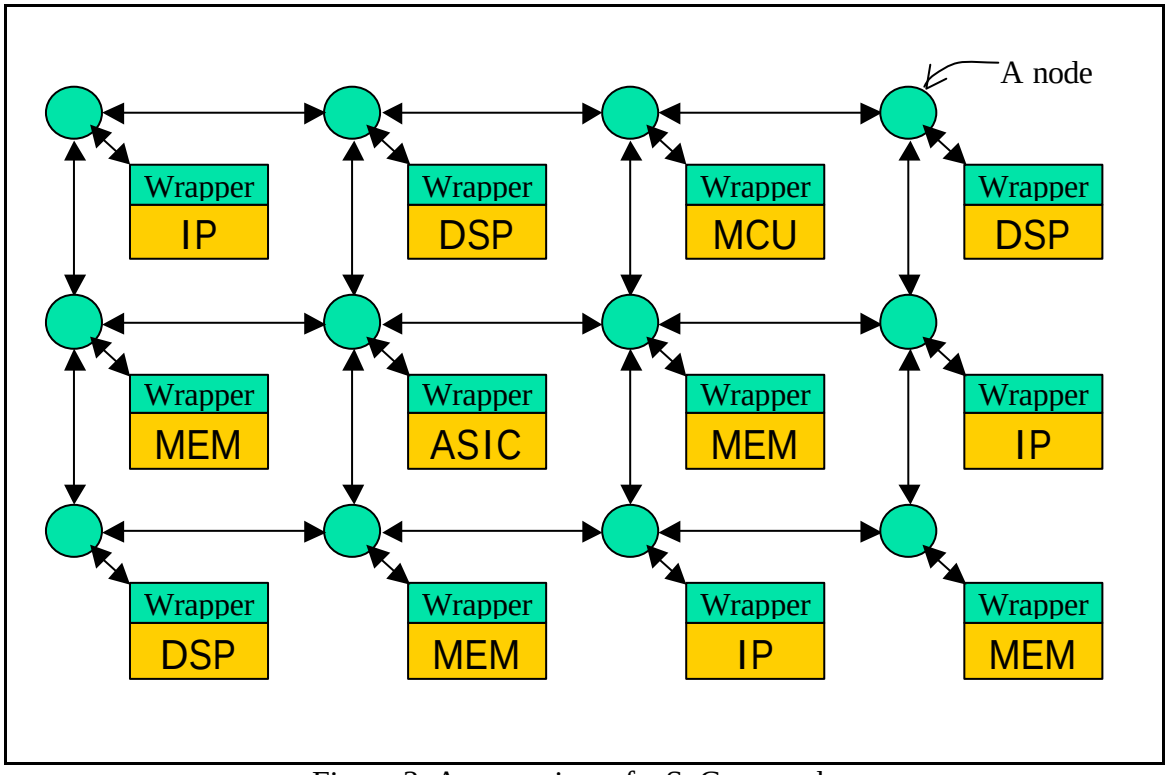


Figure 3. An overview of a SoC network

3. Review of classical SoC network research

The classical SoC network based on packet routing cannot support timing–critical real time embedded system design. Let us define embedded systems into timing–critical system and timing–not–critical system. *In timing–critical system, jobs are scheduled into cycle accurate firmware and a critical cycle count or cycle count margin must fit!* Communication systems, voice and image systems, and most embedded DSP are timing–critical systems. Ethernet or Internet is not a timing critical system because no exact arrival time can be expected. Based on packet routing, the classical SoC network buffers packet on every node and waits for a chance to deliver the packet. Time consumed for delivering a packet includes routing time, buffering time in every node, sending time in every node, receiving time in every node, and wrapper sending and receiving time. If a packet delivery fails, even longer time will be required for retransmission. Another problem from packet based SoC network is the out of order arrival time. A long data packet must be divided into more packets because of the

limitation of the size of buffers in nodes and in the receive wrapper. It further increases the network delay and enlarges the receiver buffer size because the data cannot be used until all data packets are received. In a-10X10 2D mesh network, the minimum average delay even in a small data package could be 10,000 cycles.

Buffer size in a node cannot be too small because applications are unknown when defining a SoC network for formal designs. In our research, we analyzed buffer sizes according possible applications. 240 words seem on average as a reasonable size. Including the header, a data packet of 256 words is the minimum acceptable size for data transfer. Suppose four buffers are required for a port in a node, in a five-port node, the buffer size will be 5X4X256X16bits = 81920 bits or 82k bits. In 10X10 2D mesh network, 100 nodes are required and 8.2 megabits SRAM will be required just for intra-chip communication. Using 0.13micron technology, in the most advanced foundry silicon available in the year 2002, less than 8 megabits SRAM can be integrated in a chip for a memory dominant application with acceptable yields.

Except for the silicon cost by buffers, Professor Dally estimated the silicon cost by the network logic and interconnections. Less than 6.6% of the area cost was estimated based on available silicon technology in 2000.

Therefore, the SoC on chip network proposed are either redundant, or low throughput with long latency. The authors realize that the open network concept is not necessary for SoC switch fabric when designing an embedded system. Our paper will bring in a Synthesis based design for switch fabric based on our library of macro cells.

4. Linköping SoC BUS

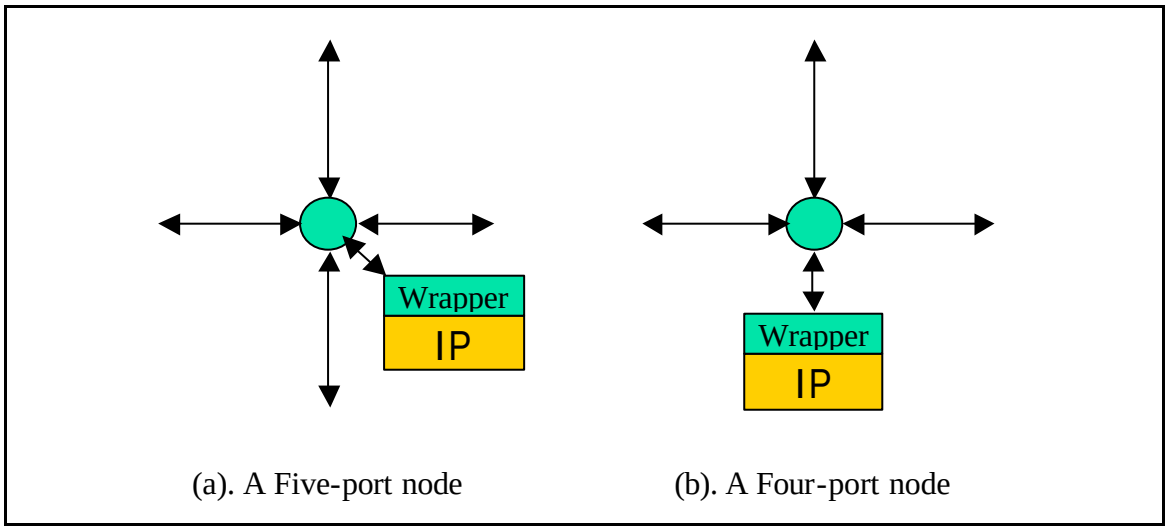
The basic starting point is to design our SoC bus for formal design of an embedded system. The basic feature of an embedded system is the “known application” during the system design phase. We need configure-ability while running applications and never need scalability when the system is designed. The scalability of our SoC bus is given only to the system designers during the system synthesis phase. The topology and the network capacity will be fixed during the system design phase. The hardware configuration for IP connection is fixed during the system design phase.

The flexibility or configure ability of the SoC bus is limited and related to setting up of dynamic connections. Flexible and high performance is only specified for routing during the run time because dynamic setting and connecting a bus will give high bandwidth and short latency whenever a transaction is required.

4.1 Network and routing

We define that our SoC bus will be used for supporting both timing–critical embedded systems and other embedded systems. The average time for setting up and acknowledge should be less than 15 machine clock cycles and the data latency should be less than 8 clock cycles when the network is based on a 10X10 2D mesh. The timing requirements we gave is based on two most critical DSP applications, the 1024 PSTN channel + 256 IP channel VoIP gateway and high-end MPEG 4 encoder plus other videoconference applications.

Two-dimensional mesh topology is decided after investigation and comparing the cost-performance with other topologies, such as 2D-hexagon, knockout, fat-tree, and omega. 2D mesh network has the relative low wire cost and high throughput. The topology of our SoC bus is therefore based on figure 3. We carefully investigated the difference between four-port and five-port topology.



(a). A Five-port node (b). A Four-port node

Figure 4. The difference between 4-port and 5-port topologies

The four-port topology gives lower performance and low connectivity because four-way switching is degenerated to three-way switching when a source-drain port is required. In most applications, a node of four-way switch without source-drain to an IP is only used to further enhance the switching capacity and a node with an IP is the normal case. Using five-port in (a) of figure 4, we need 3 bits for port coding; one more bit comparing to four-port node in (b).

Circuit switching or packet switching is compared in our research. Packet switching gives high utility of the network and packet switching does not need a central controller as an arbiter. To handle multiple transactions simultaneously, we can never use centralized control or arbitration because any centralized control-arbitration can only manage a transaction at a unit time. Therefore, a real circuit switching should not be considered for the SoC network. There are drawbacks within packet based switching. A packet-based switch cannot give quality data transfer because of the asynchronous transfer gives no certain arrival time. A packet based switching need buffers in each node to keep the data packet and wait for another hop. A buffer gives reliable transfer and extra silicon cost as well as long latency. Therefore, the conclusion here is that we cannot use circuit switch because of the multiple and simultaneously transaction is required; we cannot use packet switching because the cost is too high! *The novel solution proposed by our research is then the trade off between circuit switching and packet switching. We use packet switching to set up the connection between IP and we lock the set up as a circuit for data transmission.* We define the technology ***PCC, Packet Connected Circuit***. Using *PCC*, high reliability transfer with high throughput and fixed short latency is reached from any IP to another IP without central arbitration control. The only drawback is relative low routing flexibility when many circuits are running when using PCC technology for SoC network. We are managing this drawback in our research on our SoC bus synthesizer.

Application software in the source IP initializes the routing for a data transaction. The port driver compiles a routing packet in the source IP. The source wrapper sends the packet to the network. Package routing algorithm is investigated for setting up a bus circuit delivering only a small packet with highest priority. It is different from any routing algorithm based on OSI for an unknown network. When investigating routing algorithm, we analyze the requirements or environments and found; a routing in a node is *knowledge based routing* because the topology was fixed during synthesis and the routing is on a known hardware! This is the fundamental difference and that is why we do not use OSI seven-layer concept, it is simply not necessary. The shortest rout is given directly by the switching node if it is available. The first alternative is suggested based on the fixed network topology. The deadlock routing is simply avoided and cancelled by checking the current node name, which is given when topology was synthesized.

The size of a routing packet must be minimum to minimize the routing time. A packet carries the source ID, the destination ID, and the source HW information. The destination ID arrives at the first cycle receiving the packet so that a routing starts during the packet receiving time thus the routing time reaches the minimum. If no congestion is present, a routing finishes while still receiving a packet.

We define our SoC bus a full duplex network. Physical specification of the net for our SoC bus is based on 22 bits wires. 11 bits as a bundle for one direction including 8 bits data plus one bit network clock plus one bit forward control and one bit acknowledge control. The critical path in a node must be less than 10 of unit gate delay time. Including clock uncertainty, we can reach at least four times higher speed than a system clock because a system clock should support a critical path of 40 of unit gate delay time (a minimum critical path in a 16X16 multiplier). Therefore, every four SoC bus clock is equal to or less than one system clock. Much higher bus clock rate gives much less routing time and data latency. Later in the benchmarking section, we will describe that the average time consumption on set up and acknowledge is about 40 bus clock cycles, which equals to or less than 15 machine clocks. The network connection between nodes are specified and given in the following figure.

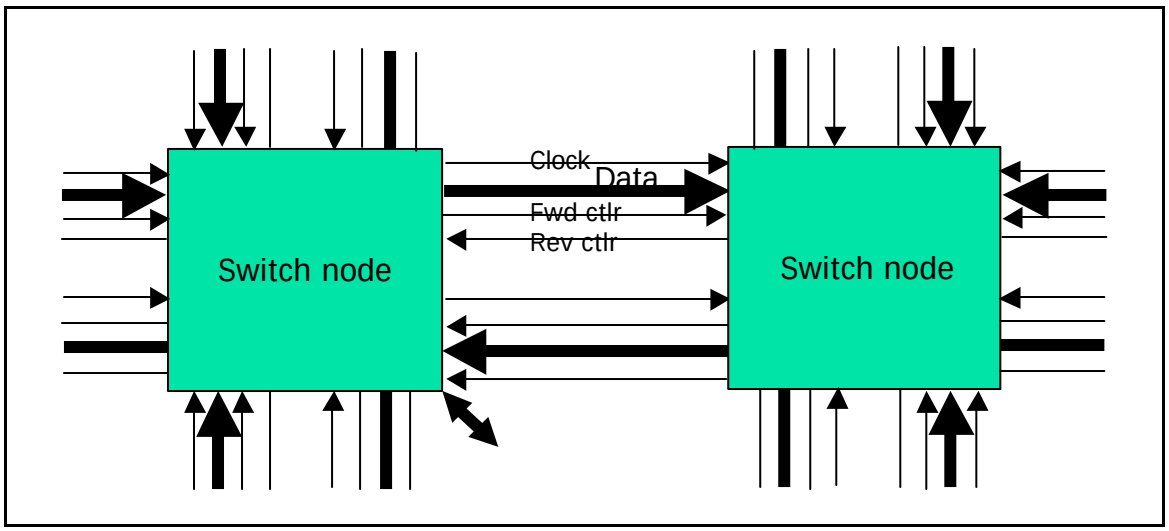


Figure 5. Full duplex net connections between two nodes

To carry clock together with sending data, data synchronization can be managed in an easy way. Mesochronous signaling is used for data recovery in our SoC bus [11].

4.2 Nodes

A node is a connector switching packages to the shortest or optimized rout toward its destination. We do not use any buffer in a node so that the silicon cost can be minimum. The functional schematic is given in the following figure 6.

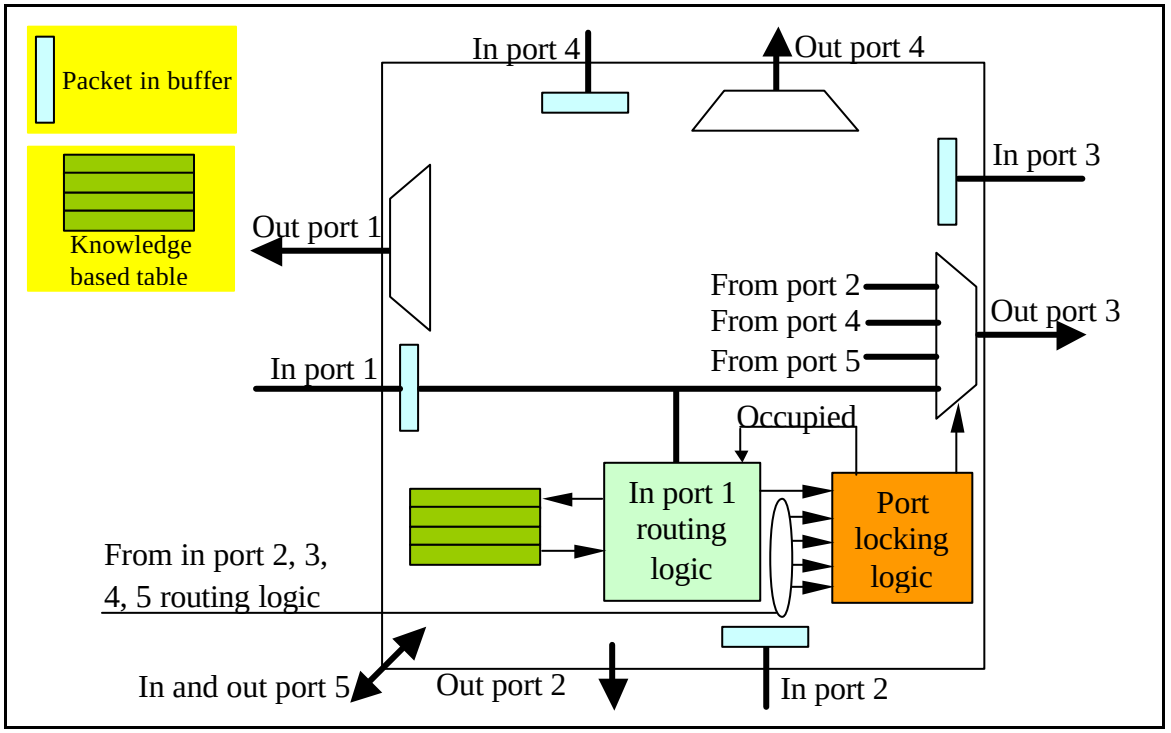


Figure 6. Node functional block diagram

There are five groups of routing circuits in a node. In figure 6, the routing circuit from in port 1 to out port 3 is illustrated. The “packet in buffer” saves the current in packet and gives the destination and source ID to the routing logic. The “knowledge-based table” gives routing suggestions according to the destination ID and the synthesized network. The output from the “in port 1 routing logic” gives multiplexing control to “port locking logic” to lock the port and send packet out if there is no *occupied* signal. The “port locking logic” gives occupied signal when the output port is locked. When two in ports apply both the output to one port, the “port locking logic” will give a hardware-based arbitration. A watchdog timer is built in the port locking logic to wait the acknowledgement and rout locking procedure. From every in port to any out port, the circuit is the same as described above.

4.3 Wrappers

A wrapper is the custom interface between the switching network and an IP port. An IP port could be based on any interface standard and it must be connected to the network with a formal connection and minimum design time. Two groups of configurations need to be done during the time for the system integration and synthesis, the physical layer configuration link the IP physical interface to the

network including matching the clock rate, the way of synchronization, the bus width, and the control signaling. The data link layer configuration gives data format matching, data transparence control if there is any, data buffering, and adaptation to the IP port driver program in the IP.

While executing an application, the port driver program sends a data frame or request to get a data frame from the network via the port wrapper with source ID and destination ID. The source port wrapper in this case is responsible for the packet compiling and buffering and the destination port wrapper in this case is responsible for decoding and buffering the received packet.

The definition for capacity–cost trade off can be divided into, and based on, two groups of applications to save silicon area as well as keeping high capacity. We define the first group of application as "*MESA* – Memory cost sensitive application" and the second group of application "*MISA* – MIPS cost sensitive application". *MESA* requires the minimum memory cost so that there is minimized buffer in the wrapper connecting to the IP. At the same time, certain MIPS will be required to adapt the communication and the size of the port driver firmware is relatively large and the run time cost for a data transaction is relatively long. It is just good if the job load in the IP is not high and of course the buffering memory in the wrapper can be eliminated to decrease the silicon cost. *MISA* requires minimized MIPS from the connected IP so that the related wrapper has to support enough size of the memory buffer and the memory cost will be relatively high. Following the definition, we will implement different wrappers in the cell library supporting the SoC bus synthesis.

To connect as many kinds of ports as possible, BVCI, Basic Virtual Component Interface and AVCI, the Advanced Virtual Component Interface were proposed by VSI [12]. BVCI defines 13 control signals and 7 buses to support the configuration and AVCI defines 20 control signals and 14 buses to support the configuration. It is too complicated for a system designer to design and configure all interface parameters. Actually, it is not necessary to leave all complex and tough issues to system designers. Because the principle of reused based design is to decrease design time, we need to move the complexity for wrapper adaptation into the SoC bus synthesis. In our research, we propose a way to classify ports based on two-dimensional features. One dimension has been discussed from *MESA* and *MISA*. Another dimension is defined according to the bandwidth and the performance of a port. The classification is given in the following figure 7.

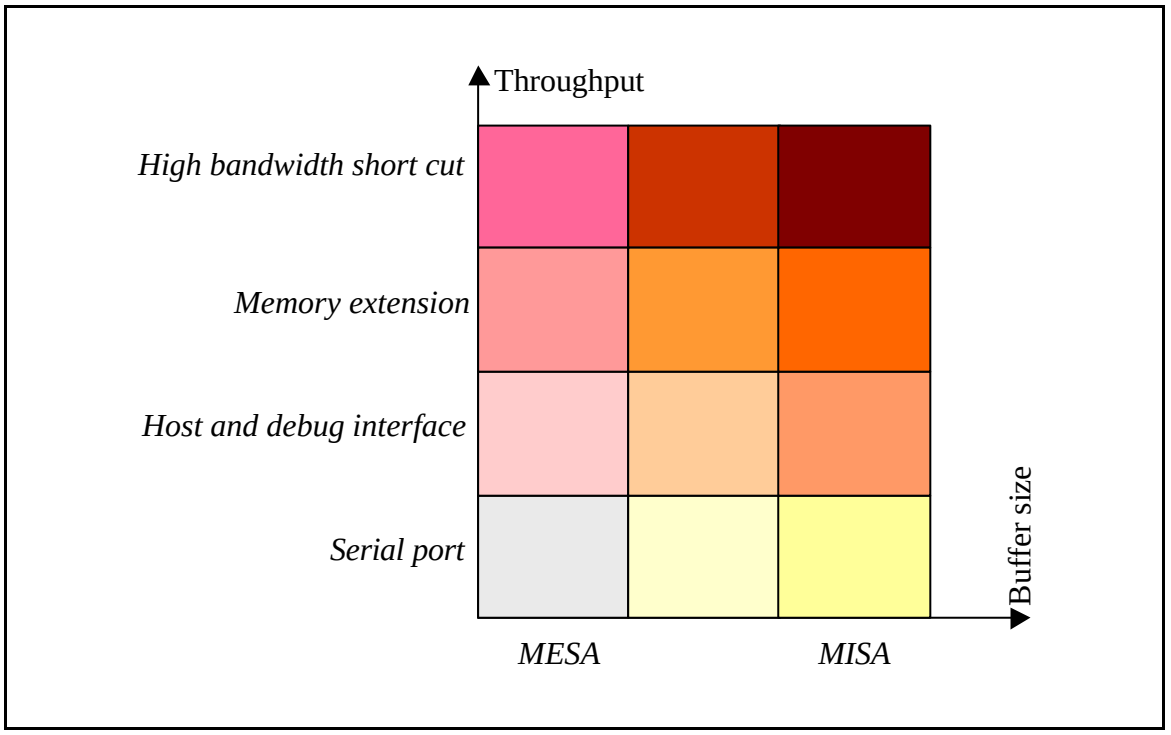


Figure 7. Wrapper classification

Following the definition in Figure 7, we will specify a group of wrappers, 12 kinds of wrappers in this case, to support different kinds of applications. Therefore, the custom configuration will be much simplified when system designers use the SoC bus as an IP core. Notice that we actually only need to design 4 wrappers instead of 12 wrappers because the difference from *MESA* to *MISA* is only the buffer size.

4.4 A transaction handling process

The port driver in the source IP initializes a SoC bus data transaction. It sends a request to the network via the source wrapper. The source wrapper compiles a packet, buffer it and send to the network. The packet in the source wrapper will be kept until the accepted acknowledge is received. The source wrapper resends the packet either receiving a rejection or time out (after a basic wait time plus a random time if the acknowledge is not received). Keeping and resending a packet in the wrapper instead of in the IP, we can down load and distribute unnecessary jobs from the IP.

When a packet is in the SoC network, a node close to the wrapper sends the packet to the neighbor node following the shortest distance to the destination. If the packet cannot be delivered, the network rejects the packet and acknowledges rejection to the source wrapper. If the destination wrapper accepts the packet, the destination wrapper sends an acknowledgement to the network and the acknowledgement packet goes back via the forward rout and locks the rout from the destination wrapper to the source wrapper. An accepted forward rout in a node will temporally be locked for certain clock cycles waiting for its acknowledgement. As soon as the rout locked, a circuit is set up for the data transmission and data is on the circuit bus instead of sending to the network.

The destination wrapper first coordinates the source hardware and configures for the data transaction. The destination wrapper informs the connected IP port and send acknowledgement.

Both nodes and the destination wrapper monitor the data transfer via the control wire. The source wrapper sends finish signal on the control wire in parallel with the data bus. The lock will be released as soon as the finish is detected. Finally, a FSM (Finite State Machine) in a node is given in the following figure to improve the description on a data transaction.

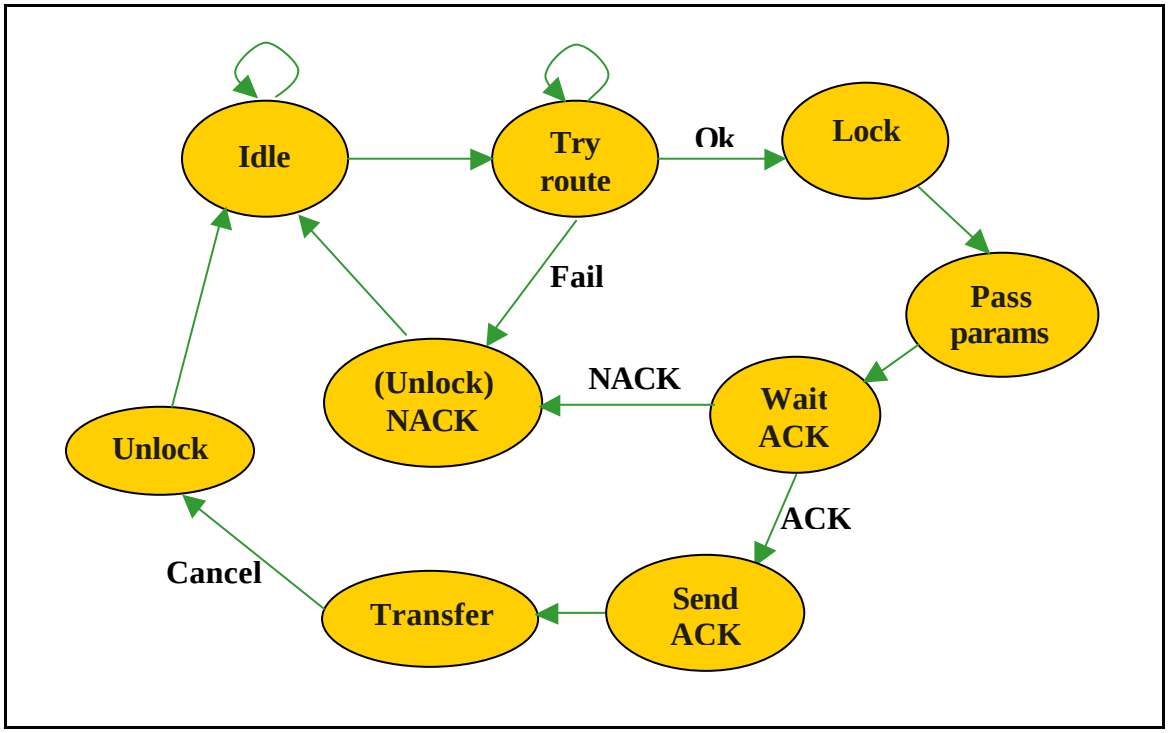


Figure 8. A FSM in a node

4.5 User's design flow

System designers cannot use the SoC bus without a design methodology and a strong synthesis environment. Our research and demonstrator will contribute also the integrated design environment for SoC bus users. We will first generate the network library including the sub set of topology, the sub set of nodes, and the sub set of wrappers. For extensive timing critical system, we may even include physical implementation of library cells. We are now developing the behavioral simulator for debugging and benchmarking. The α version of the behavior simulator is now in use for the benchmarking research. We will investigate and develop the network synthesizer. The synthesizer is a behavior synthesis tool generating RTL code and C based port driver program as reference. The input of the synthesizer is the port related specification and the synthesizer analyzes the requirements on performance and cost. The synthesizer generates both RTL codes and C-based port driver to the simulator. The simulator gives benchmarking to match the requirement specification. Iteration is necessary and re-synthesis will be performed to follow up the requirements.

The designer's design flow is given in the following figure 9. System SoC designers start from partition the interface related specification to the related port hardware and related port driver in IP. After the behavior job partition on paper, a step of writing C-based behavior model is necessary for starting the synthesis. The synthesis will be performed based on iterations until the requirements are satisfied. The iteration cannot be automatic according to the project plan. As a hardware research team, we only give the infrastructure of the design environment. The integration on CAD might be given by other SW based research teams.

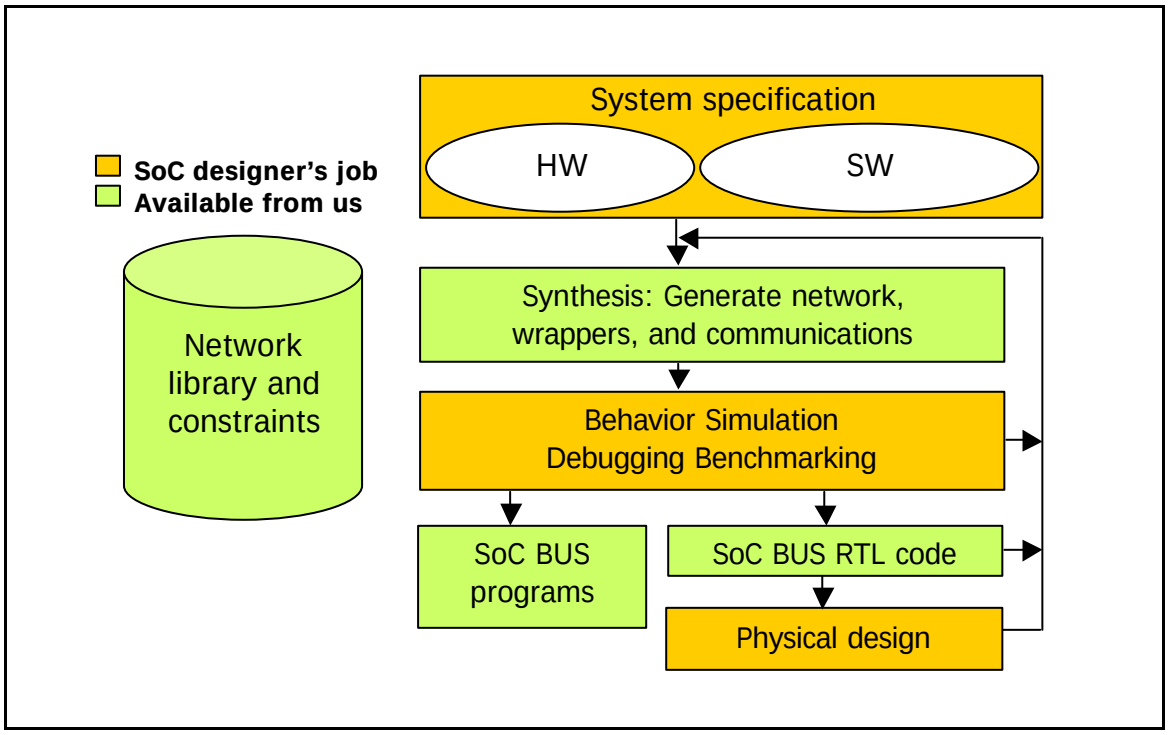


Figure 9. The designers design flow

4.6 Capacity analysis and Benchmarking

The benchmarking is based on a predicted chip set, which could be a possible product after 10 years. The data traffic loads on a gateway chip are 1024 PSTN channels with electronic echo cancellation plus 256 VoIP channels with voice CODEC of G.723 and G.729. The benchmarking is simulated based on 8X8 2d-mesh network. The SoC bus clock is 1GHz and the system clock is 250MHz. It means that we have to manage at least 16 mega packets per second and each packet is about 0.5k words. Simulation is undergoing and there is no congestion has been found until now. We are increasing the job load and the congestion point of the SoC bus will be recognized soon.

5. On going work

The project became a demonstration project sponsored by socware of Invest Sweden Agency, ISA, since 2002. The project has been also financed by SSF, the Swedish Strategic Research Foundation since 2000. Acreo AB might joint in the project soon. If Acreo joins the project, we will speed up the implement for a demonstration chip. Related research on physical layer of our SoC bus is also going well and low power lowlatency interconnect circuit was investigated. [11]

6. Conclusion

We have described our SoC bus and our related research based on detailed analysis of other current on going research projects. We define the so-called “classical SoC network” by delivering data packet on network. We point out that the classical SoC network is not feasible because of the excessive silicon cost and long data latency. We introduce a novel concept, PCC: Packet Connected Circuit. Based on PCC, a data transaction is initialized by packet routing. The rout is locked as a bus circuit after proving and acknowledging the rout. Therefore, we reached fixed and low transfer latency based on high bandwidth supporting multiple simultaneously data transfer. At the same time, buffers in nodes are eliminated so that the silicon const of a SoC network becomes feasible. MESA and MISA model are proposed for improving wrappers. Benchmarking is discussed base on the predicted heaviest integration for complex functions. A SoC system designer’s methodology is proposed.

7. Acknowledgement

Socware of Invest Sweden Agency (ISA) and SSF the Swedish Strategic Research Foundation are acknowledged for financing the project. Authors would thank to Professor Christer Svensson for his cooperation.

References

[1]. Michael Keating and Pierre Bricaud, Reuse Methodology Manual for System-on-a-Chip designes, KAP, 1998, ISBN 0792381750.

[2]. Andrew S. Tanenbaum, Computer networks, third edition, PTR, 1996, ISBN 0133499456.

[3]. www.vsi.org

[4]. Dake Liu and Daniel Wiklund, SoC BUS, EAD träfa 2000, April-2000. Stockholm, Sweden.

[5]. Daniel Wiklund and Dake Liu, Switched interconnect for System-on-a-Chip design, in proceeding of the IP2000 Europe conference, 2000.

[6]. Daniel Wiklund and Dake Liu, Design of a system on chip switched network and its design support, IEEE ICCCAS02, 2002.

[7]. William J. Dally and Brian Towles, Route Packets, Not wires: On chip interconnection network, DAC 2001, Las Vegas.

[8]. M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, A. Sangiovanni-Vincentelli, Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design, DAC 2001, Las Vegas, NV, USA.

[9]. Luca Benini, and Giovanni De Micheli, Networks on Chips: A New SoC paradigm, IEEE Computer, January, 2002.

[10]. Ilkka Saastamoinen, et al, Interconnect IP node for future System-on-Chip Deigns, IEEE int. workshop on Electronic design, Test, and Applications, 2002.

[11] Peter Caputa and C. Svensson, Low-power, Low-latency global interconnect, 15th international ASIC/SOC conference, Rochester, NY. 2002.

[12]. VSIA: On Chip Bus Development Working Group, Virtual Component Interface Standard Version 2 (OCB 2 2.0) April 2001.