Flit Synchronous Aelite Network on Chip

Examensarbete utfört i Elektroniksystem

vid Tekniska högskolan i Linköping

av

Mahesh Balaji Subburaman

LiTH - ISY - EX -- 08 / 4198 -- SE

Linköping 2008

Flit Synchronous Aelite Network on Chip

Examensarbete utfört i Elektroniksystem
vid Tekniska högskolan i Linköping
av

Mahesh Balaji Subburaman

LiTH - ISY - EX -- 08 / 4198 -- SE

Handledare: **Kent Palmkvist**

ISY, Linkopings Universitet

Andreas Hansson

NXP Semiconductors

Kees Goosens

NXP Semiconductors

Examinator: **Kent Palmkvist**
ISY, Linköpings Universitet

Linköping, 10-June-2008

LINKÖPINGS UNIVERSITET

Avdelning, Institution
Division, Department
Division of Electronics Systems
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Datum
Date
2008-06-10

Språk
Language
☐Svenska/Swedish
■Engelska/English
☐ _____

Rapporttyp
☐Report category
☐Licentiatavhandling
■Examensarbete
☐C-uppsats
☐D-uppsats
☐Övrig rapport
☐ _____

ISBN -

ISRN
LiTH-ISY-EX-- 08 / 4198 -- SE

Serietitel och serienummer ISSN

Title of series, numbering -

URL för elektronisk version

Titel
Title
Flit Synchronous Aelite Network onChip

Författare
Author
Mahesh Balaji Subburaman

Sammanfattning
Abstract

The deep sub micron process technology and application convergence increases the design challenges in System-on-Chip (SoC). The traditional bus based on chip communication are not scalable and fails to deliver the performance requirements of the complex SoC.
The Network on Chip (NoC) has been emerged as a solution to address these complexities of a efficient, high performance, scalable SoC design. The Aethereal NoC provides the latency and throughput bounds by pipelined time-division multiplexed (TDM) circuit switching architecture. A global synchronous clock defines the timing for TDM, which is not beneficial for decreasing process geometry and increasing clock frequency. This thesis work focuses on the Aelite NoC architecture. The Aelite NoC offering guaranteed services exploits the complexities of System-on-Chip design with real time requirements. The Aelite NoC implements flit synchronous communication using mesochronous and asynchronous links.

Nyckelord
Keywords Network on Chip, Mesochronous, GALS

Abstract

The deep sub micron process technology and application convergence increases the design challenges in System-on-Chip (SoC). The traditional bus based on chip communication are not scalable and fails to deliver the performance requirements of the complex SoC. The Network on Chip (NoC) has been emerged as a solution to address these complexities of a efficient, high performance, scalable SoC design. The Aethereal NoC provides the latency and throughput bounds by pipelined time-division multiplexed (TDM) circuit switching architecture. A global synchronous clock defines the timing for TDM, which is not beneficial for decreasing process geometry and increasing clock frequency.

This thesis work focuses on the aelite NoC architecture. The aelite NoC offering guaranteed services exploits the complexities of System-on-Chip design with real time requirements. The aelite NoC implements flit synchronous communication using mesochronous and asynchronous links.

Acknowledgements

Above all, I would like to thank my supervisor at NXP, Andreas Hansson for his thoughtful guidance and support. He has motivated my understanding and approach to this research work. I am grateful to Prof. Kees Goosens for his valuable suggestions and continued encouragement.

My sincere gratitude goes to my university supervisor Dr. Kent Palmkvist who has always been supporting this thesis work and helped me with proof reading. Special thanks to my friends who worked with me at NXP.

Finally, I am deeply indebted to my parents and friends for their never ending support.

Table of Contents

1. Introduction

System on Chip (SoC) complexity grows with deep sub micron technology enabling multiple intellectual property (IP) blocks exist on a single chip. The different traffic types in the same design due to application convergence forces the resources to be shared. The time to market pressure and IP reuse strategies demand scalable interconnects.

Traditionally buses have been implemented for interconnection between IP blocks in complex SoCs. Buses do have poor separation between computation and communication. Thus they cannot adapt to the system architecture changes. The change in the system architecture has impact on the bus architecture, which has to be redesigned [6]. In Deep sub-micron process technologies, buses dominate in terms for silicon area, speed, and performance [7]. These deep sub micron effects have made the global synchrony hard or expensive in terms of area and power to maintain. The shrinking size and increasing clock frequency poses problem for designers to have a single global synchronous clock [5].

These considerations have moved data communication from shared buses to the packet switching networks. The Network on Chip (NoC) addresses the problems of complex System on Chip (SoC) with multiple cores and memories. The NoC characteristics are:

- Decouple computation from communication through layered architecture that helps to optimize them independently [1].
- User defined network topology and flexible configuration based on the application requirements [1].
- Effective reuse of resources [1].
- Structure and manage global wires in deep sub micron technologies [9].
- Good wire utilization through sharing, reduction of global wires [8].
- Scales better than buses [4].
- Solves SoC timing convergence problem through (GALS) Globally Asynchronous and Locally Synchronous design style [5].
- Utilizes less silicon area than multi layer bus [6].

1.1 Problem Description

The Aethereal NoC has three major components Routers, Network Interface and links between them. In the Aethereal NoC guaranteed services (time –critical) and Best effort services (non-time-critical) are offered to IP modules based on their connections and requirements. The real time

application has cores demanding time related guarantees. These guarantees are provided by the TDMA slot reservations in the Network interfaces (NI) which are run time reconfigurable [10]. The components have logical notion of synchronicity in NoC, such that they all remain in same fixed duration slot. This results in a contention free routing. The single global clock supports the global synchronicity in Aethereal NoC for both NoC and IP Blocks.

The SoC design has major problems with the timing closure and power dissipation [5]. One of the major reasons is the use of centralized single global clock across the chip. The clock distribution of the global clock over large area with negligible skew consumes lot of power. Secondly longer wire has more wire delay, which becomes a performance bottleneck [11]. In Clock lines, it contributes to clock skew. The placement and routing has more workload to tackle these timing closure problems. Thus the Synchronous design does not scale well with the increasing process and design complexity [12, 13].

The hard real time embedded application depends not only on the logical or functional correctness but also on their timing constraints. The multiple cores or blocks in a complex SoC have their own specific performance requirements such as minimum throughput and maximum latency demanding the guaranteed services.

1.2 Contribution

The Aelite NoC is the lighter version of the Aethereal NoC offering guaranteed services only. The Guaranteed services are the lossless, in order delivery of the uncorrupted data with guaranteed throughput and bounded latency [3]. In comparison with the Aethereal NoC, Aelite NoC architecture provides higher performance of about 1.2 X the frequency and lesser design complexity contributes 4X the lesser area.

The problems with the Synchronous design and application convergence has made SoC to favour multiple clock domains and to relax the strict requirements of the clock skew. The mesochronous architecture helps to have different clocks, which are the derivative of the global clock in a system. These clocks have constant clock frequency with unknown phase differences. It avoids the problem of distributing clock with minimum or negligible skew and clock tree balancing between different blocks of the chip.

The better solution to the problem of running at the same clock frequency for both IPs and NoC is to partition the design into synchronous blocks each running at their native frequency. The data is exchanged asynchronously using handshakes. The GALS architecture promises these solutions. The Asynchronous clock bridging on the NoC edges between IPs and NI, also inside NoC between NI and Router.

1. Introduction

The clocking rate of individual blocks need not have any dependencies with the other blocks or the master clock. The blocks such as routers and network interfaces (NI) run synchronous locally and communicate asynchronously with each other. The same applies for NIs and IPs. It mitigates the problems associated with redesigning due to timing closure issues in the back end design.

2. Aethereal NoC

The Aethereal NoC provides guaranteed services (GS) such as uncorrupted, lossless, ordered data delivery, with guaranteed throughput and bounded latency [1]. Every IP core has its own performance requirements in real time applications such as minimum throughput and bounded latency. The Best effort services (BE) are introduced to increase the network resource utilization. The GS-BE architecture can concurrently support the real time communication which uses guaranteed services and non-real time communication using best effort services. The GS has higher priority over BE services [14]. The Aethereal NoC properties can be highlighted as:

- Decoupling computation (IP) from Communication (NoC)
- Backward compatibility with existing bus protocols.
- Supports real time applications.
- Flexible topologies according to the design requirements.
- Run-time reconfigurable system.

The major components of the Aethereal, the Network Interface (NI) and Router are interconnected based on the topology selection and configuration. The services are mapped according to the traffic demands as connection, which have 2 channels to send and receive data packets. The packets access the shared resources. Resource conflict arises when two packets try to access the same resource at the same time. This leads to unpredictable behaviour by the either dropping the packet or delaying them. Guaranteeing throughput and latency needs to address this issue by reserving and releasing the shared resources.

The links are used and shared by a connection for a fixed duration using Pipelined TDM circuit switching [1]. Each link is multiplexed in time, thus single link can carry several flows. The arbitration to access a link is controlled by TDM slot table, which results in a contention free network [10]. The contention free routing is implemented by TDM based slot table in the Network interface (NI). The data packets are organised as flits. The TDMA has fixed timeslots for every flit. The shared resources are reserved for a flit for a particular time slot and released at the end of the slot. This result in no two flits will conflict to use a resource at the same time. The time period (T) of a slot is the time taken for the flit size words (N) at each hop. In other words, the propagation delay of the router. So, in a time period T of a slot, N words can be transferred. The NoC components such as Routers and Network Interface (NI) consume and deliver flit size words in one time slot. The slot table algorithm depends on the specific application and design. It is run time programmable. All components should be globally synchronous to occupy the same fixed time slot to avoid contention. For every hop the slot table advances one slot. So, every component should

synchronize with their neighbours for every slot. This global synchronicity is achievable by distributing a single centralized global clock. However for larger design it has problems due to global clock skews and wire delays. The simple solution is to run the design at lower frequencies to avoid timing closure issues. Also, the slot synchronization could be implemented in a distributed manner using synchronous data flow model [discussed later in section 5].

A TDMA Slot table is used to allocate bandwidth for different flows. Throughput is guaranteed by the number of slots reserved for a connection. Slot corresponds to the bandwidth. Total bandwidth = N* Bi, where N = No. Of slots for a connection; Bi = Slot Bandwidth. The latency is bounded by the waiting time until the reserved slot arrives and the number of routers or hops to reach the destination [1].

2.1 NI – Network Interface

NI converts the IP protocols such as AXI, DTL, OCP etc., to packets. It has two parts: NI- Shell and NI –Kernel
NI-Shell – Implements connections based on the application demands and take care of the protocol compatibility between NoC and IP cores.
NI- Kernel – Implements channel, packetizes messages, manages the end-to-end flow control and Clock Domain Crossings (CDC) to the IPs.

NI Kernel communicates with NI shells through FIFO ports. Each port can have number of connections to allow differentiated services. A peer-to-peer connection has two channels, request and response channel. Each channel has two queues (Input and output) in each NI-Kernel (Source and destination). The Guarantees are provided for a connection by reserving a slot for that connection in the Slot Table Unit (STU), configuring for end-to-end flow control in the respective credit registers and routing information in connection table. The end to end flow control ensures that no data is sent unless there is sufficient space at the destination buffer. Hence buffer overflow is avoided.

2.2 Router

The router routes the data from source to destination ports based on the packet header. The NI programmable registers embed the route description or path details in the packet header and the packets reach their destination by source routing. The Router decodes the packet header for the routing information and forwards them accordingly. The two classes of traffic Best effort services (BE) and guaranteed services (GS) have two separate queues or channels inside router. The link is shared between these channels based on priority. The GS packets have higher priority over BS packets. During absence of GS packets, the BS packet uses the link. Thus the router needs to have

the scheduler and local link level flow control. The local link level flow control is necessary to check the space availability in the BE queues of the neighbouring router.

In the current prototype of Aethereal NoC we use the centralized configuration where:

- Routers do not need to be configured.
- Source routing is used with packet headers having the routing information.

The contention is avoided when two packets trying to access the same resources at same time [10].

- All routers go from current slot to next simultaneously using a synchronous global clock.
- Filling slots such that the sets of simultaneous GS connections are contention free.

2.3 Limitations of GS-BE architecture

* Global Synchronous Clock: Currently GS-BE architecture runs at global clock frequency, there by synchronizing with IP cores and NoC. This lacks scalability with the design complexity, high on-chip data rates and reducing process technologies. [as explained in section 1]

* Area: The GS-BE architecture in Aethereal has both BE and GS channel in the router parallel occupying more area when compared with GS- only Aelite. The Best effort packet queues and round robin scheduler with local link level flow control increases the complexity and size of the router.

* Performance: The Best effort blocks cannot guarantee the performance explicitly such as guaranteeing the lower bounds on throughput and upper bounds on latency. The BE arbiter which performs round robin scheduling is in the critical path of the router. This limits the maximum frequency it could run.

The GS traffic has resource reservation for the worst case. This results in under utilization of the resources during the normal case. On the contrary, combined GS-BE services enhance better resource utilization by using the unreserved or reserved but unused slots for the BE traffic. But it is not most favourable in the real time applications having strict requirements on the timing, area and frequency.

The GS-BE system can also be implemented by allocating the resources with guaranteed features only. This increases the number of GS only routers, which has positive impact on the system occupying lesser area and running at higher frequencies. We get good performance normalised to cost.

3. Aelite NoC - Guaranteed Services Only

In the Aethereal NoC, all components survive on a single centralized global clock, which has difficulties in the back end placement and routing in high performance complex SoC. The NI/ Router components must be cycle level synchronous, in other words the output data word from a component should reach its neighbour in the next clock cycle. The link delays due to longer wires are not addressed.

The aelite NoC addresses these issues by providing only the guaranteed services with bounded latency and throughout. The clock is allowed to have skews or phase difference inside the NoC by using the mesochronous link architecture. The muliple clock domains could be implemented using the asynchronous wrapper, which introduces globally asynchronous, locally synchronous architecture.

The aelite NoC composed of network interfaces (Shell & Kernel), links and routers. The processors or IP modules are connected to each other and with their memories through aelite NoC. It guarantees the minimum bandwidth and maximum latency, end-to-end flow control, FIFO scheduling. These services are needed to control the IP communication traffic. In the next section we discuss about the Aelite router architecture.

3.1 Aelite Router

The Aelite router has major blocks such as: Header Processing Unit (HPU), GT queue, one hot encoder (OHE), GT reservation unit (GRU) and Switch.. The design supports 3 pipeline stages corresponding to the router latency of 3 cycles as drawn in figure 1.

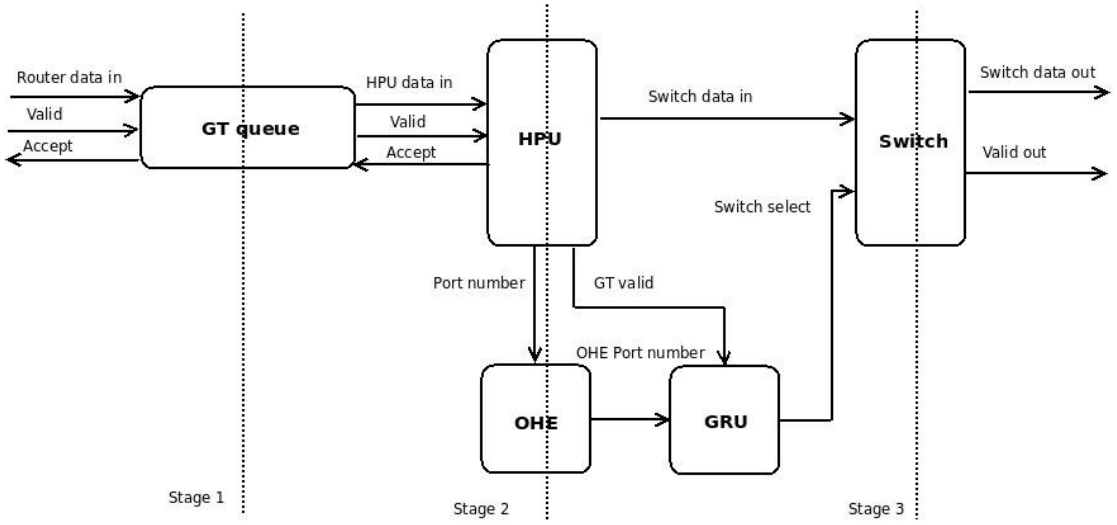

Figure 1: Aelite Router architecture

3.2 GT queue

The Flits from the Network interface Kernel are buffered in the queue. From here, the Header-processing unit (HPU) fetches the word using valid - accept handshakes. The Queue size is determined by depth * width based the design. The depth is number of flits to buffer in the queue that is normally one and width corresponds to the flit size.

3.3 Header Processing Unit (HPU)

The GT queue FIFOs handshakes with HPU for every word transaction. Based on the input word, either header or payload of the words are treated accordingly using its finite state machine states.

Every word in a flit is accompanied with the control words denoting its validity (data_valid) and if it is End of Packet (eop). Where as in the Aethereal router where every word has its own tightly mapped control bits denoting different aspects such as GT/BE flit, number of valid words in a flit, and eop. Thus the HPU has to keep track of which word it process to decode the correct control message.

The aelite router has eliminated this complexity, which is necessary for the mixture of GT and BE traffics. The seperate bits (data valid and eop) for every word have relieved HPU from the router critical path.

Another advantage is the flit is not necessarily fixed to constant or fixed flit size ie.,3.

The design has flexibility to adapt the variable flit size, which could even be configured at run time. Two FSM states of the HPU are Init and GT. The HPU remains in Init state when it processes the header of a flit and in GT state for the payload transactions.

Init State: The Packet header is parsed in this state. The header of the packet holds the path information with other useful details like Control bits, End to End flow control Credits, Output Queue.

Example of header mapping:

For Data width of 32 bits, the above details are encoded as,

EOP	: 33
Data Valid	: 32
End to end flow control credits	: 31: 27
Destination Output queue	: 26:22
Path	: 21:0

GT state: The payload of the packet is analysed for their validity and End of packet. If end of packet has been reached the state goes back to init expecting a new packet header. If not, it forwards the payload to the router output ports indicated by their header. The payload of a packet is trails or follows the path mentioned in the header uninterrupted.

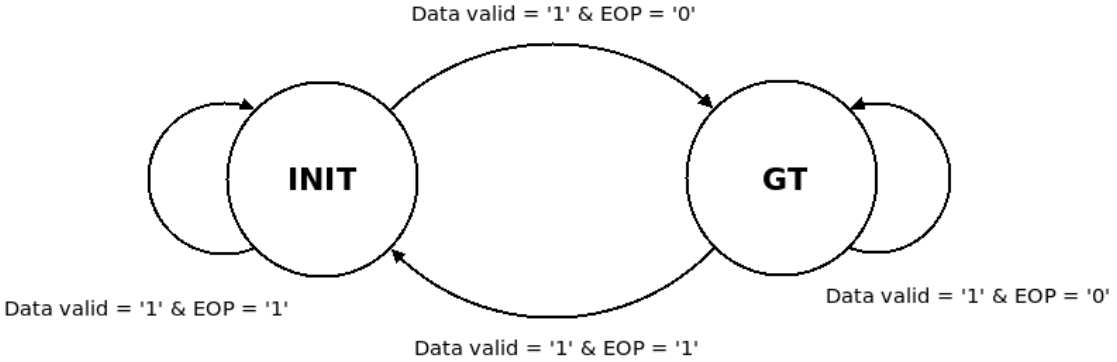

Figure 2: HPU-FSM states

Data_valid	EOP	Current State	Status	Next State
0	X	Init	Invalid	Init
1	0	Init	Parse header	GT
1	1	Init	No Payload	Init
1	0	GT	Payload continues	GT
1	1	GT	Packet Ends	Init

HPU signals data validity (data valid) and its target port number to its successors.

3.4 One hot encoder - (OHE)

The high performance synthesizable switch favours the one hot encoder on both ASIC and FPGA [15]. The port number to which the word has to be delivered is sent to one hot encoder (OHE). The one hot encoder sets one bit in the state register for each state, in other words it uses one flip flop per state. So that only one flip-flop is turned on at any time. The one bit change during single transition helps to detect the single bit error. OHE simplifies the logic and interconnect between the logic. It results in faster design but utilize more area than using the tri-state buffers [15]

Example:
An Arity 5 router requires 5 flip-flops for one-hot encoding.
000 – 00001
001 – 00010
010 – 00100
011 – 01000
100 – 10000
101 – 00000 - Invalid state
110 – 00000 - Invalid state
111 – 00000 – Invalid state

3.5 GT reservation unit (GRU)

The encoded output of the OHE flows to the GRU. The valid words are identified by the data valid signal from HPU. These valid signals help GRU to mask the invalid port numbers. The valid output port numbers are passed on to the switch as switch select.

Example:
Arity 5 router, having valid words on ports 1 and 4.
Encoded GRU Inputs * GT_valid = Output_ports
(Transposed one hot outputs)

00000	*	10010	=	00000
00001	*	10010	=	00000
00110	*	10010	=	**00010**
01000	*	10010	=	00000
10000	*	10010	=	**10000**

3.6 Switch

The switch logic is an important part of the router. The processed data from HPU is switched to correct destination or router output ports using switch select values. The switch implementation has effect on the router performance and cost.

Example:

Port no.	Switch_data_in	Switch_Select	Switch_data_out	
4	**ddddd**	00000	00000	<= o/p port 4
3	00000	00000	00000	<= o/p port 3
2	00000	00010	**ddddd**	<= **o/p port 2**
1	**ddddd**	00000	00000	<= o/p port 1
0	00000	10000	**ddddd**	<= **o/p port 0**

3.7 Simulation results

Figure 3 shows the simulation results of the Aelite router.
Point A: The data word enters the router (Router_data_in). It get buffered in the queue.
Point B: The data word leaves the queue in the next clock cycle to HPU.
Point C: The header got parsed and ouput port numbers are determined. These port numbers are one hot encoded. These encoded port numbers are masked by the GRU and sent as switch select. The hpu ouput word reaches the switch as switch data input.
Point D: The data words are switched to the respective output ports based on the switch select signal. Thus a word takes 3 cycles to hop the router.

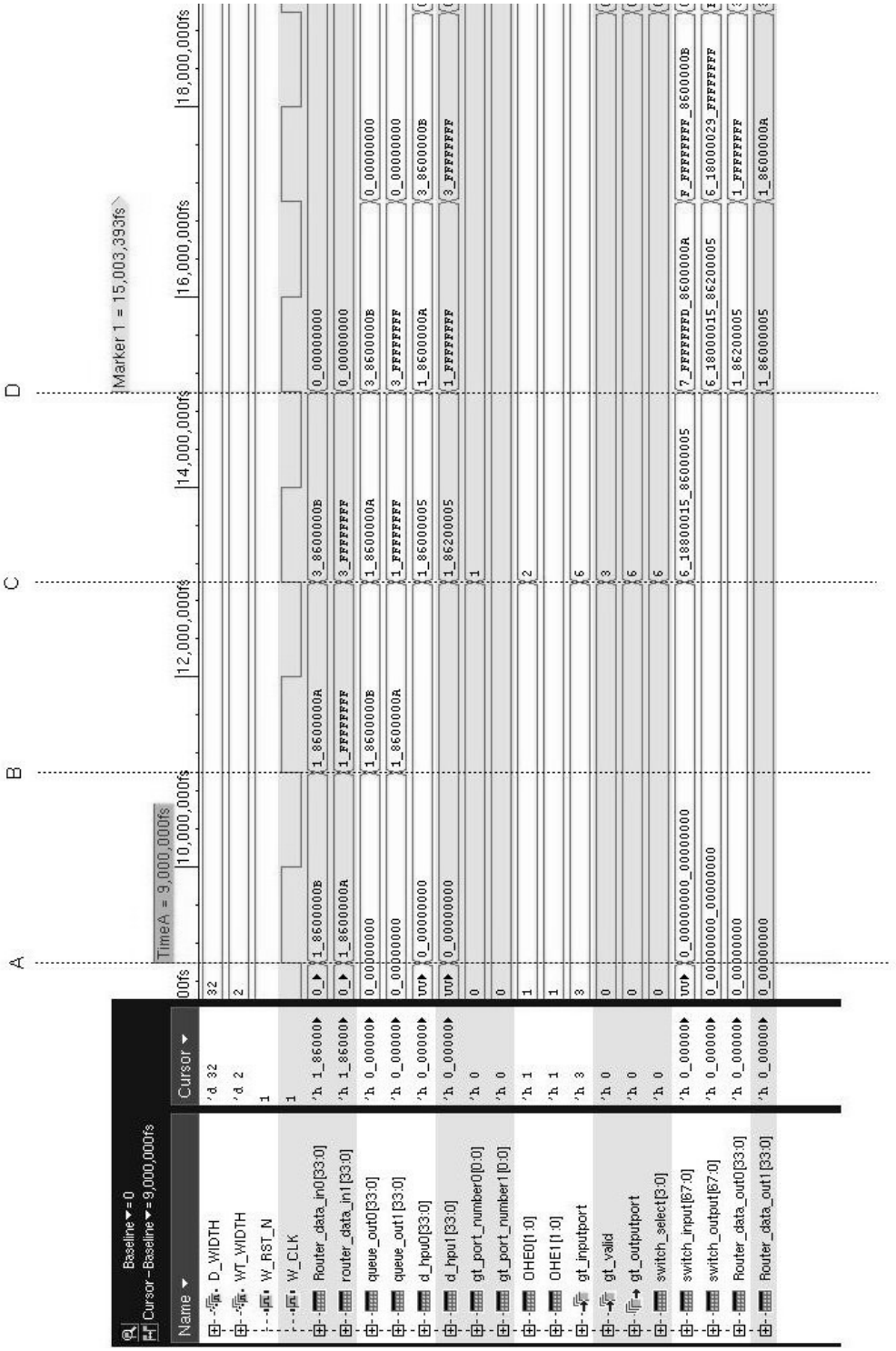

Figure 3: Simulation results of Aelite router

3.8 Synthesis results

Arity	Architecture	Technology	PVT	Max. Freq. (MHz)	Area (um2)
5	Aethereal	Cmos90nm	Wccom	680.27	69935.779
5	Aelite	Cmos90nm	Wccom	826.45	17390.37

Table 1: Synthesis results comparison

Area : Aelite router occupies 4 X less area
Performance : Aelite router operates 1.2 X higher clock frequency.

3.9 Limitations

The aelite NoC uses the sychronous global clock which imposes limitations on the NoC operating frequency. The link between the routers can have one cycle delay which limits the placement of routers. The long links with delays larger than one cycle could be pipelined. In that case, the input register in the aelite router could be moved to their links.

To ensure the correct data transfer, low skew clock has to be maintained. The NoC design that supports mesochronous links eliminates these problems with synchronous global clock.

4. Mesochronous NoC

The increasing die size made the clock distribution harder for future multi-GHz design due to interconnect parasitic, jitter, clock skew, duty cycle, power dissipation etc., [16,17,18]. The Mesochronous design getting more attention where data and clock travel together and data is realigned with the local clock phase of the receiver. The clock distribution problems are alleviated by allowing, every logic block running at the same clock frequency but with unknown phase shift.

4.1 Mesochronous Link

The link between the network components of Aelite architecture has been used to transfer data signal (32 data bits, 2 control bits). The low link level flow controls between the routers are not necessary during transfer, as it does not have the best effort traffic. In other words, routers have only one virtual connection. The data synchronization is necessary when we allow different clock phase relationship inside the NoC. Thus the physical links are implemented as link buffers between the components for signal synchronization.

The Link buffers are bi-synchronous FIFOs or Embedded asynchronous FIFOs used for fixing the skewed phases. These link buffers could be allocated a time slot in the slot table unit where slot duration is the time taken for a flit to cross the link. The FIFOs are combined with a finite state machine (FSM), which is synced with the read clock. The link buffer is accessed once in every 3 cycles by the reading or receiving router/NI. This is ensured by the FSM states. The FSM along with link buffers promises that a slot could be allotted for the link between the components. Thus the Guaranteed services are obtained by the allotting a slot for the link model.

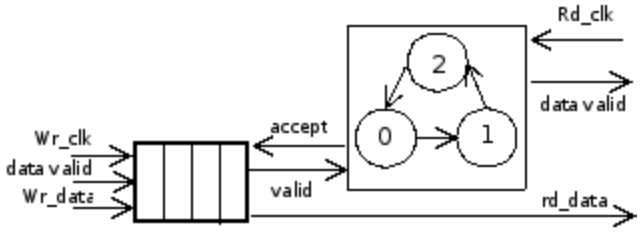

Figure 4: Link Model

The clocks can be skewed either positive or negative. The FIFOs fix these phase differences of read and write clock. The synchronised flits waits for the FSM handshakes or permission to enter the Router/NI. The read clock aligned FSM, fetches the flits for the components. It checks the FIFO for a flit and signals the Router/NI to accept the flit. The checking happens once in every 3 cycles. So we have three cycles for the link buffer to transverse the data words/ a flit.

Allowing the clock skews of half a clock cycle, the propagation delay of the FIFO buffer not more than 2 and a half cycle, we can safely read the input flit aligned with the reading clock at state 0. The FSM has 3 states namely state 0,state1, and state2. When it is state0 it checks for the word in the link buffer. If word is available, FSM sends the read accept signal to the link buffer and signals router for data valid. During the state transition for every read clock cycle i.e. from State0 to State3, three words of a flit are fetched and passed on.

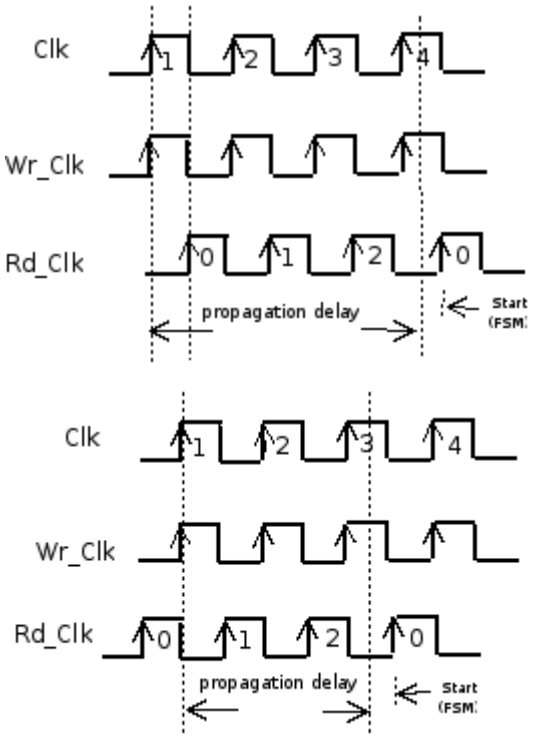

Figure 5: Clock cycle waveforms for both skews

Every flit must arrive the router/NI on its reserved slot to avoid contention during routing. The flits that arrive early or late made them scheduled at wrong slot. Therefore the words of a flit must be available on their respective FSM state or before. When it arrives earlier it has to wait for the FSM handshake (read accept). The word0 must present at FSM state 0 or 2 cycles earlier and word 1 and word2 must be forwarded in the following FSM states 1 and 2 consecutively without any delay.

It resembles a synchronous data flow actor which fires upon token arrival and produces the token at the output [as explained in next chapter]. The Link buffer should be sized such that it would not underflow or overflow under any circumstances. When the skews are bounded, sizing could be done by analysis [19,20].

The total time spent for link traversal will not exactly be 3 cycles from the global clock point of reference due to the clock skews or phase difference between writing and reading clocks. But the

FSM associated with the FIFOs absorbs this difference by reading the flit once in every 3 local clock cycles. Every component moves from one slot to another according to its local clock thereby accepting the new flit. The slot advancement differs between the components absorbing the phase differences between them. Thus the flits are realigned with the reading clock cycle and made flit synchronous.

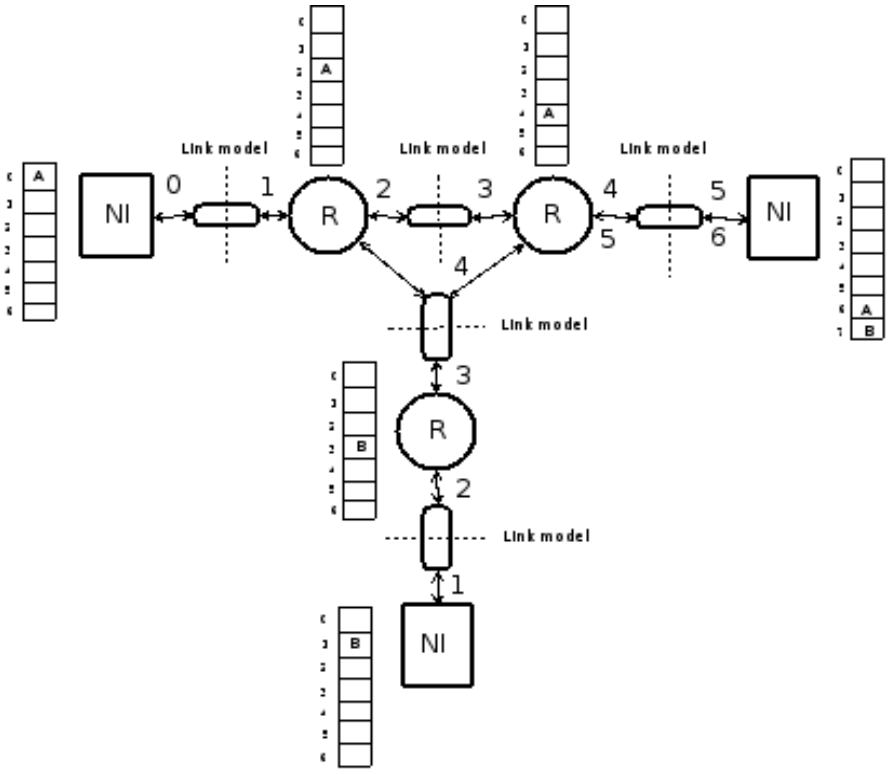

Figure 6: Mesochronous Architecture

4.2 Limitations

The Mesochronous adaptation of the Aelite NoC is simple, but the Mesochronous link architecture is only suitable for the design where all components have the same clock frequency with phase difference. When the frequencies are different, each component has to be synchronised asynchronously and the components should be patient to synchronise every input flits upon its arrival.

5. Data Flow Model

In the future, many embedded product will be heterogeneous in terms of cores, memories and interconnects. The resource management and communication and synchronization between the components are crucial to meet the strict deadlines in the real time application domain.

A Firm real time embedded applications have hard deadlines in terms of throughput and latency. A predictable system needs to guarantee these timing requirements. The resource management of the system must be known for bounding the timing behaviour. The SDF model of computation can be used to model the behaviour and fulfils the requirements of a predictable system by using analytical techniques [21]. This chapter explains the Synchronous Data Flow Model of computation to design the flit synchronous Aelite NoC.

5.1 Model

The SDF graph nodes or actors represent a specific functional computation. The communication channel or arcs between the actors represent the data path. Thus the data dependencies between the actors are revealed. The actors communicate with each other by tokens. Tokens represent data containers.

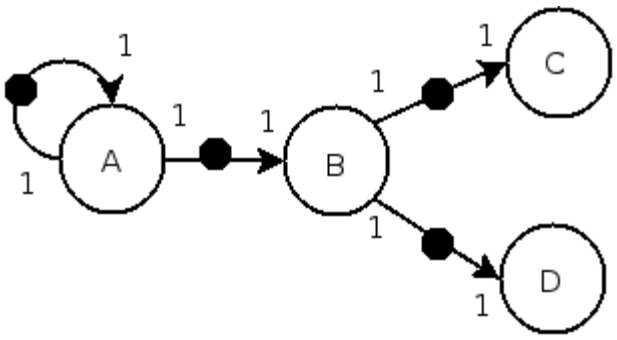

Figure 7: Synchronous Data flow actors and tokens

An actor can perform its computation or in other words it can Fire when it has sufficient input tokens available on its input arcs. A Firing rule specifies the number of tokens consumed and produced by each actor.

The Firing has the following steps:

- Actor consumes a token from its input arcs
- Actor produces a token on its output queue after the execution time.

The input arcs holds the sufficient number of initial tokens to avoid deadlock. Deadlock causes the actors to get stalled without sufficient tokens on its inputs to fire during execution. The act of firing the actors is controlled by the availability of the input tokens. This is called as Data Driven. The arcs are assumed to have unlimited capacity for SDF. Each actor fires whenever there are sufficient tokens at the input arcs. As a result of firing, the actor produces tokens in the output arcs.

An actor firing is atomic and cannot be interrupted. The Firing rule clearly specifies when an actor could be enabled, the number of tokens consumed and number of tokens produced. In SDF, if the actors consume and produce one token per firing, it is called as homogeneous SDF. Thus all actors have fixed rates or deterministic behaviour. The deterministic nature of the SDF helps to design the predictable system [22]

5.2 Implementation

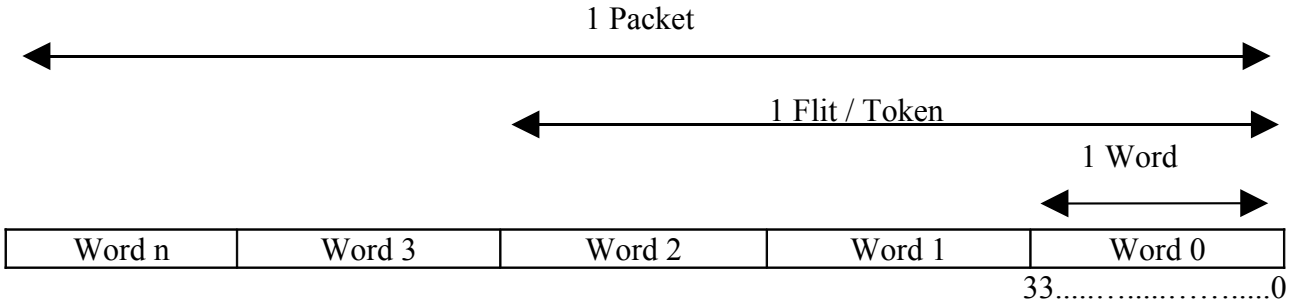

Figure 8: Packet format

In SDF model, each actor can Fire independently or concurrently based on the arrival of the input tokens instead of the global clock signal. The components such as routers and NIs are modelled as actors. The delay caused in a channel doesn't affect the slot table allocation because the router and NI wait for the token to arrive in order to fire. Thus the Global synchronicity between the NoC components can be implemented in a distributed manner on the flit level or slot level not on clock cycles. This implementation of SDF model has no effect on the NoC design flow.

The arcs or communication channel between actors are implemented as FIFO buffers.
The tokens reside in these buffers before/after they are consumed and produced. The Initial tokens are the Empty token that has no valid data and initialized in the input buffers to avoid deadlock condition.

Every component will follow the same firing rule such as:
- An actor (Router/NI) fires when there is sufficient flits (N words) in the input buffer and sufficient space for the flits in the output buffer.
- As a result of firing, an actor (Router/NI) produces a flit (N words) in the output buffer.

The data synchronization has issues with the data path delay, which has been addressed by pipelining them or mentioning them as the Multi cycle path in the synthesis tools or by using latency insensitive design. The bi-synchronous FIFO or embedded asynchronous FIFO could be used to fix the issues with clock phase and frequency.

5.3 Analysis

Every actor has bounded execution time i.e., the time span between the consumption of a input token and production of a output token. The execution time refers to their worst-case execution time. When we use the actors with fixed execution time, the overall worst-case throughput of the system can be determined by periodicity property and overall latency of the system is determinable using bounded ness property. [24] The SDF actors have fixed number of token consumption and production. The self-timed execution i.e. all actors fires as soon as firing rule is satisfied, enable us to determine the maximum throughput of the SDF graph [23].

Allocating the sufficient resources for each actor is necessary to fulfil the constraints. The resources can be utilized for a fixed amount of time corresponding to the slot period. The token communication between actors has the transfer delay (t), where t = Token size / Bandwidth [24]. The predictable system must have all their task executions bounded. Each arc or the communication channel should have bounded storage capacity. It should never underflow or overflow. The optimal buffer sizing is necessary to have minimum buffer without affecting the timing requirements of the system.

6. Latency Insensitive Design

The Application convergence has made present and future chip to allow higher degrees of heterogeneity in System on Chip (SoC). Each application can be mapped on any of the cores. The IP cores and modules run at their native frequency, independent of each other enabling them to process the data in parallel to meet the computational demands. Thus the synchronous system is not scalable for growing chip complexity in terms of size and performance requirements.

In Synchronous Distributed system, every component is modelled as an individual actor and they communicate via channels. Every actor must be functionally correct and able to run independently. Such a system must be prone to channel delay or latency insensitive. [26]

- Latency insensitive design is not affected by inter connect or channel delay.
- We can split the larger design into modules of synchronous blocks handshaking asynchronously among themselves.
- The Synchronous blocks are made latency insensitive using wrappers.
- The time critical long wires or communication channel could be pipelined.
- When the actors and FIFO channels has fixed latency, the whole distributed system has a bounded latency. So, for any chosen architecture we can guarantee the throughput and latency.

In an asynchronous system, each module or components operate not depending on the global clock. Each module can be modelled as a patient process whose behaviour is insensitive to the wire delays. The data must be latched correctly with their receiving clock to avoid the Meta stability.The communication channel can be modelled as bi-synchronous fifos or Embedded Asynchronous FIFOs, which will synchronise the data between read and write clock acting as a clock domain crossing.

The channel or interconnect delay between components is hard to predict on earlier phase of the design. The timing closure problem arises during placement and routing while dealing with longer wires and balancing the clock tree. In a synchronous data flow model, every actor fires when it has sufficient tokens on its input buffers. Each process is patient or stalled until all tokens arrive in all its input ports tolerating the channel delay.[27]

In our implementation of the model, every router & NI waits for a token or flits to perform an action and produces the token or flits. By making the component to stall for every flit cycle to gather required flits, all components are made flit synchronous with each other. The varying

communication channel delay doesn't alter the functionality of the Router/NI. The channel has been implemented as bi-synchronous fifo or embedded asynchronous fifos which helps the data to travel safely between different clock regions. Wrappers are designed to stall the components for flits. Wrappers help each component to act like a stallable process. These processes communicate with communication channel using valid & accept handshakes.

6.1 Skew Insensitive Design

The Skew insensitive design is needed in the case of mesochronous and asynchronous design.[28] The skew on the clock tree distribution is tolerated which helps the design to run at higher clock frequency. The difference between the arrival times of the clock signal across the chip is called as clock skew. While designing a system, the data propagation delay and the clock skew must be taken into consideration to avoid timing closure problems. The allowable clock skew is always less than the shortest data path, which allows us to have larger clock period or lower frequency.

The major components of NoC are NIs (Kernel + Shell), Routers and Link between them. Those links are implemented as FIFO buffers such as Embedded asynchronous FIFO, which exists in NXP library. It is asynchronously micro-pipelined and has series of stages or cells communicating with each other by handshakes. The
Read and write interfaces of the fifo are connected with read and write clocks along with their data.

These interfaces synchronises the internal asynchronous control signals like read / write enable, empty / full signals with the read and write clocks. Thus it is possible to have clock signals with different phase and /or frequency to read and write data in the FIFO [30].The Embedded Asynchronous FIFO exhibits area efficient, low power and high-speed operation. (Refer table 2).

	Register-based Gray FIFO [31]	Bi-synchronous FIFO [31]	Embedded Asynchronous FIFO [30]
Depth*Width	16*32	16*32	32*37
Area (um2)	20364	13384	6554
Process technology	Cmos 90nm	Cmos 90nm	Cmos 90nm

Table 2 : Synthesis results of different FIFO designs

The delay in the data path and clock skew affects the data transfer from one region to another. Thus the skew and transmission delay or wire delay between blocks cause synchronization problems in the time division multiplexing over the pipelined path. [29]. The Timing closure and clock tree distribution problems could be easily fixed using the GALS model. In which all components or

processes think locally and act globally. They have freedom to run at different phase and / or frequencies. The above mentioned latency insensitive and skew insensitive design fixes the synchronization issues in the real time embedded applications where multiple clock domains exists.

7. Globally Asynchronous Locally Synchronous (GALS) NoC

The Synchronous global clock needs more care to be taken and limited by the physical Size of the chip and the Maximum frequency it could run. The clock distribution has a major share in power consumption [5, 18]. For the high performance design these issues are mitigated by considering each design blocks running synchronous locally and communicating with each other asynchronously.

7.1 GALS Design

The GALS architecture is helpful in designing the distributed systems, allowing the blocks as synchronous islands. To implement the latency and skew insensitive model the NI / Routers do have asynchronous wrappers. So all components run synchronously locally but communicate asynchronously by means of handshakes. These synchronous islands communicate through link buffer such as embedded asynchronous FIFOs or bi-synchronous FIFOs used for clock domain crossing. For guaranteed services, each component must be slot synchronised with each other. This results in having asynchronous wrapper for every component, which buffers the token or flits from its neighbours.

7.2 Asynchronous Wrapper

The Asynchronous wrappers has Port interface (PI) and Port interface Controller (PIC) as shown in figure 9. The NoC components communicate with each other via these wrappers and clock domain crossing link buffers.

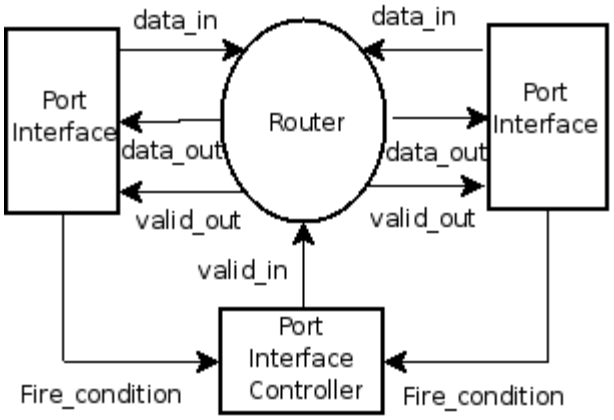

Figure 9: Asynchronous wrapper

7.3 Port Interface (PI)

The Port interface (PI) acts as the interface between the Router/NI and link buffers as depicted in figure 10. It has FIFO buffers and counters. The FIFO buffers are used as the Input and output queues for the corresponding port. The counters are used to count the number of words enter or leave the input and output queues.

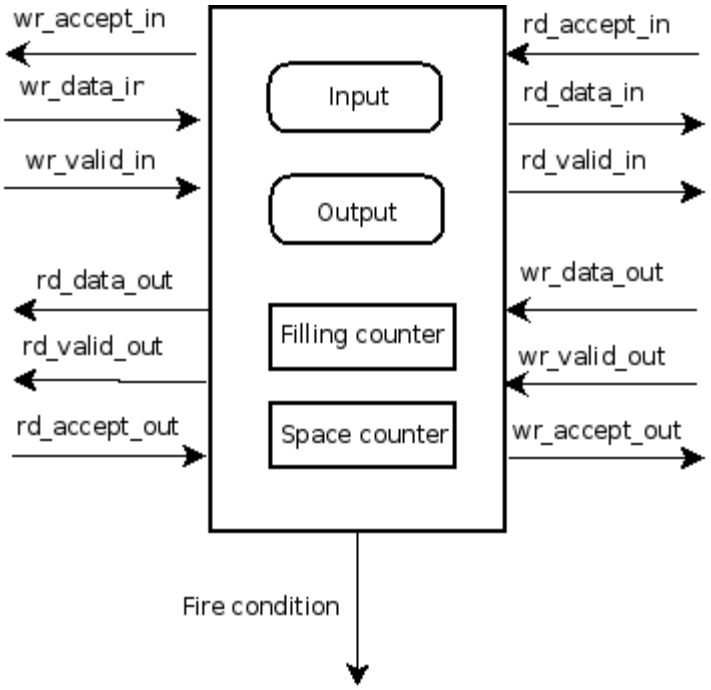

Figure 10: Port Interface (PI)

Figure 11 explains the functional block diagram of a port interface. The words or flits are collected in the Input and Output queues. Fill and space counter act as their corresponding counters. Fill counter keep track of the number of words available in the input queue. Space counter counts number of free spaces available in the output queue. During flits transaction, the filling counter increments when a word enters the input queue and decrements when it leaves. The space counter increments when a word leaves the output queue and increments when it enters the queue.

To achieve flit synchronicity, every component modelled as the stallable process such that it buffers the flits from all its neighbours. When there are sufficient number of words i.e. a flit (a flit = 3 words) in the input queue and sufficient number of spaces for a flit in the output queue, the fire condition is satisfied. When all port interfaces has fire condition signal enabled high. This guarantees the availability of flit in all its input queues and has sufficient spaces reserved for them in the output queues.
The router/NI can start accepting the flit. The signal Fire condition acknowledges this firing rule in the data flow model [as explained in chapter 5].

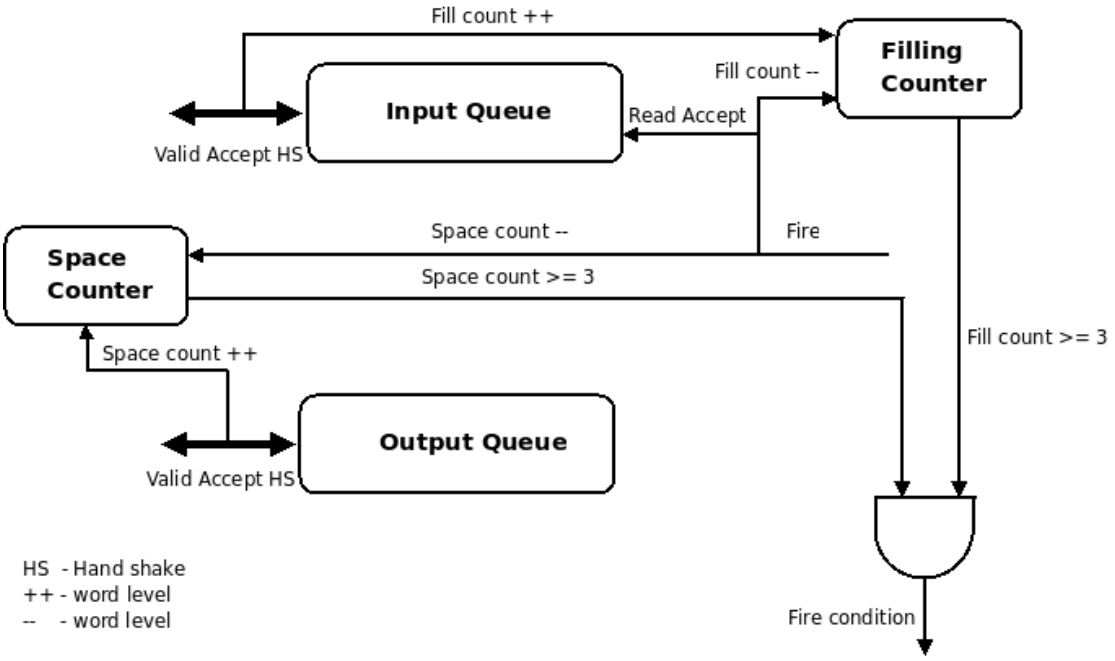

Figure 11: Port Interface (PI) functional block diagram

7.4 Port Interface Controller (PIC)

In the previous section – Data Flow model, we have discussed about the Initial Tokens to avoid deadlock. PIC instructs Port interfaces, when they have to generate the Initial tokens in their input queues.

7.5 Tokens Generation:

The initial token generation are taken care by these wrappers instead of router to reduce the complexity in the Header processing Unit (HPU).

The PI & PIC are responsible for token generation and token passing. Their action points are:

- Generates Initial “empty” tokens after reset
- Pass the tokens from the adjacent Link buffers.

To perform these actions, they have 2 states namely, Init or Initial state and Start or Normal state as shown in figure 12.

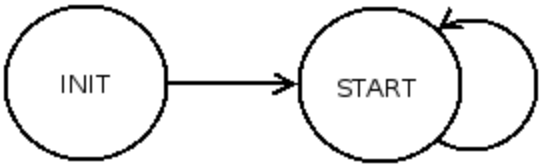

Figure 12: Token states

INIT:

- Empty token generation.
- Word Counter is incremented each time a token is generated.
- When the counter reaches the flit size in other words when the flit size tokens are generated, it moves on to start state.
- The Empty token valid from PIC is active during Init state after reset. This signal enforces the port interface to produce sufficient Initial empty tokens in the PI - input queues.

START:

- The available spaces in the output queue and available words or token in the input queue are checked for firing rule in all Port interfaces around the router.
- If every link control has flit size tokens (empty /data) in the input queue and flit size spaces in the output queue then FIRE is ordered.
- The normal operation is resumed.

7.6 Functional Description

Port interface controller receives the Fire condition signal from all the port interfaces (PI). When all the port interfaces has Fire signal high/active, the Port interface controller issue Fire signal high. The Fire signal from the PIC confirms that all the port interfaces has the firing condition satisfied. This is the valid in for the Router and read accept for the PI queues. The Token (data /empty) enters the router followed by the Fire signal.

The Router processes the Input flit and routes them to the corresponding output ports based on their header details. The consumed flits appear at the output queues after router latency cycles, indicated in figure 13. The router latency is 3 cycles representing the pipeline depth of the router's data path. When the input registers are removed, the router has 2 cycles latency. The Fire signal (valid in) accompanying the input data is delayed or registered for router latency cycles and used to signal writes valid accompanying output data word. The Output data with write valid is used for writing data in the Output queues of the PIs.

The PI and PIC guarantees that the Flits or tokens that are passed to the router, belong to the same flit cycle. By asserting the fire condition signal high, every PIs confirms that they have the flits

arrived and have spaces reserved for the output data from router. The collective fire condition or logical Anded fire condition is the Fire signal output from the PIC. It confirms the flits are passed synchronously from PI – input queue to router. Thus every router is synchronized at flit level.

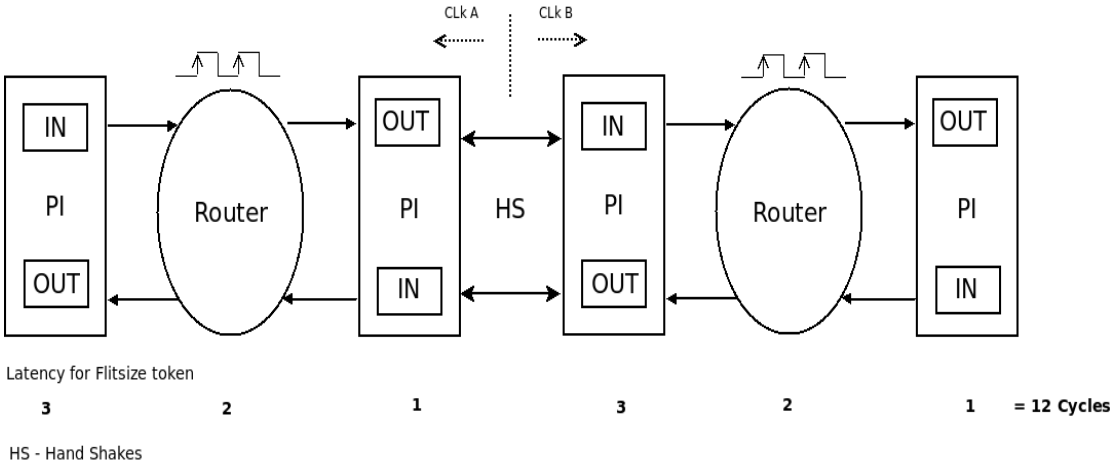

Figure 13: Token Latency

Unlike the router, which has no data stalling inside the pipelined registers, the NI stalls the data before they get scheduled for every flit cycle. When a router or NI output port has no data to be sent, it sends empty token signalling the neighbour that it has advanced one slot in the slot table. This produces every output port a token either data or empty for every transaction. As we have seen the fire signal is the combined signal of all the fire condition from PIs. Without these empty tokens the adjacent router/Ni will wait for the token from all its neighbours and cannot fire immediately, this would cause the entire system to deadlock eventually.

7.7 Simulation Results

Point A: The flits from 2 different links are buffered (w_wr_data_in0, w_wr_data_in1) and synchronised. As soon as the entire flit arrives, PI enable hign the fire condition (fire_condition). The router will not fire (fire) until it receives a flit in all its input. When the fire condition from both PIs are high, Port interface controller commands fire. It indicates the router that the firing condition is satisfied. The flit from the queue enters the router followed by the fire. We could see 3 words entering the router in the next 3 cycles (data_in0, data_in1).

Point B: The flit entering the router processed out (data_out0, data_out1) after router latency cycles. Here in this case it is two cycles. The ouput word gets stored in the output queue in the next clock cycle ready to be read (w_rd_data_out0, w_rd_data_out1).

Point C & D: The flit remains in the output queue until it receives the read accept handshake signal (w_rd_accept_out0, w_rd_accept_out1) from the link buffer. The flit is released from the queue when the handshake is done.

Point D: The New flit entered in the input queue waits to be scheduled. Fire condition is not high because the output queue is not free, which is occupied by the last flit. As soon as the last flit leaves the output queue the fire condition goes high.

7. Globally Asynchronous Locally Synchronous (GALS) NoC

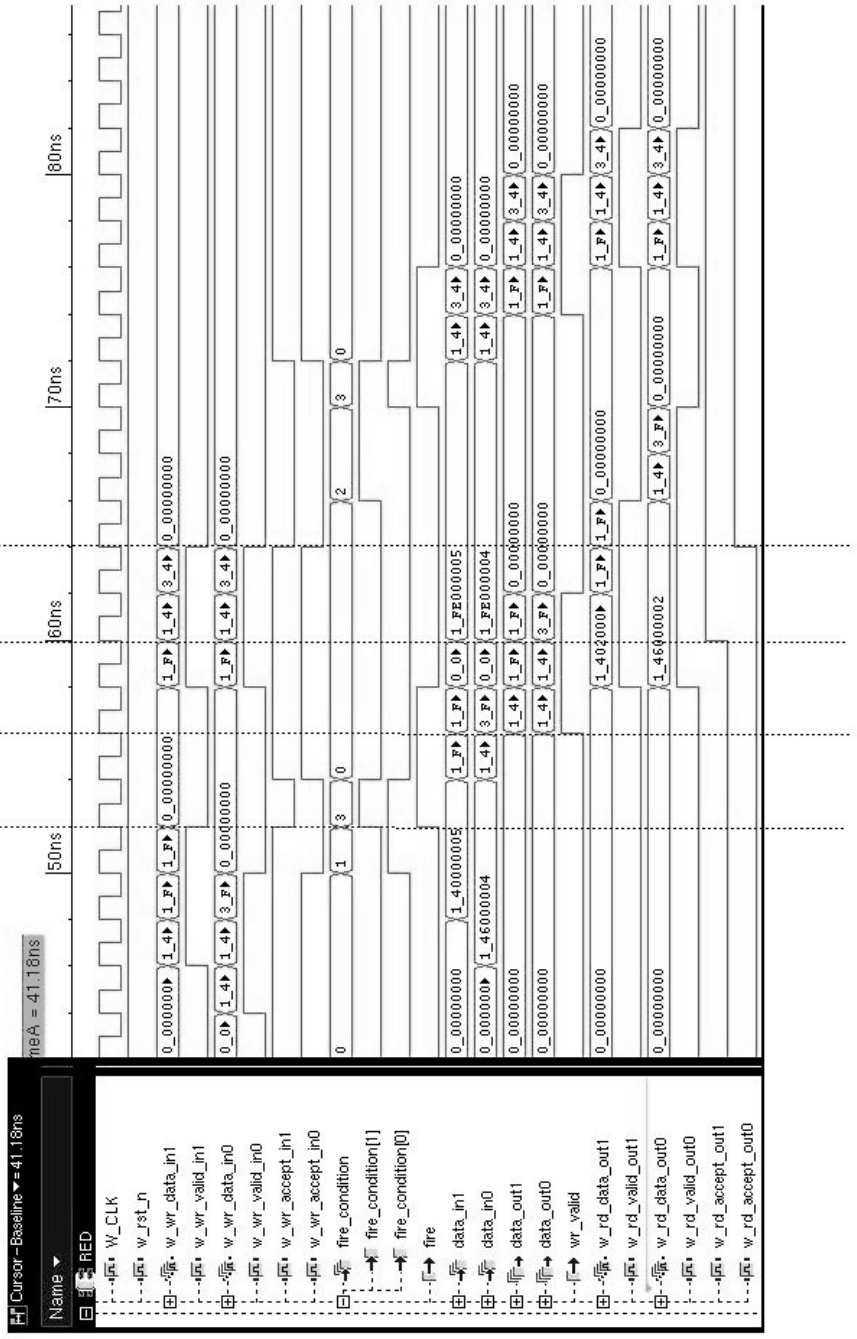

Figure 14: Simulation results ofAsynchronous wrapper

7.8 Synthesis Results

The synthesis has been done on Cmos 90 nm process technology under the condition of worst case commercial (WCCOM). The total area of the arity 2 router with asynchronous wrapper architecture is 30325.590 um2. The Port interface has the FIFO queues of width 34 and depth 6.

Design Blocks	Total area (um2)
PIC	229.39
Port Interface	13216.201
Router	3646.227
Combinationa	17.562

7.9 Limitations

The flit synchronous Aelite NoC overall frequency is determined by the slowest component. The overall frequency could be improved by implementing link width conversion in the component wrapper architecture.

8. Results Comparison

Figure 15. shows the trade-off between the target frequency and cell area of Arity 5 router with data width 32 bits using Cmos 90 nm technology.

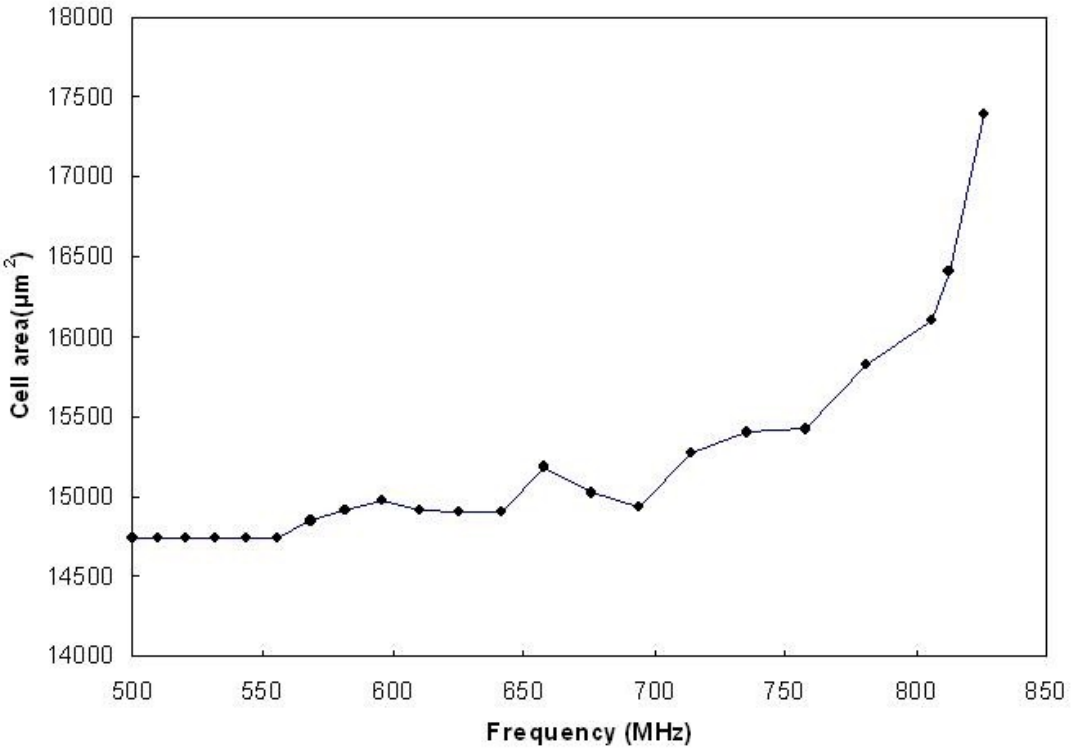

Figure 15: Frequency and Cell area trade off

The maximum operating frequency and cell area scaling with different data width for arity 6 router are shown in figure 16 and figure 17. we could observe maximum frequency decreases linearly with the increasing word width. The area increases linearly with the increasing word width.. Thus the router offers larger throughput at a lower area. e.g. An arity 6 router offers 64 Gbyte/s at 0.03 mm2 for a 64 bit word width.

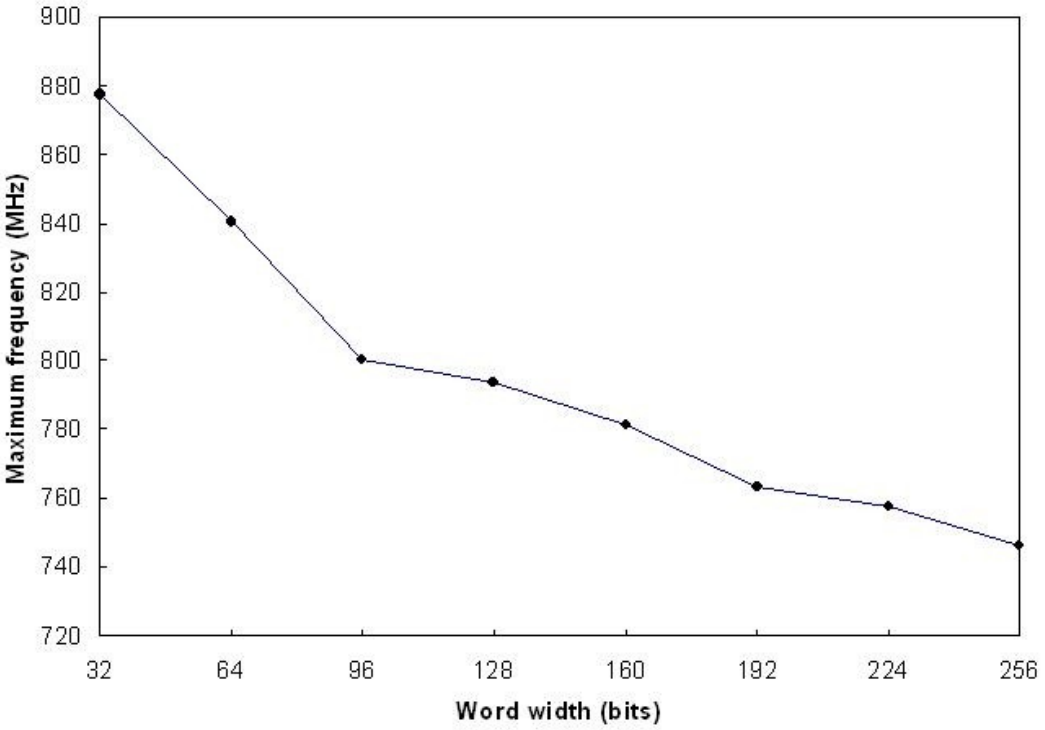

Figure 16: Maximum frequency for varying word width

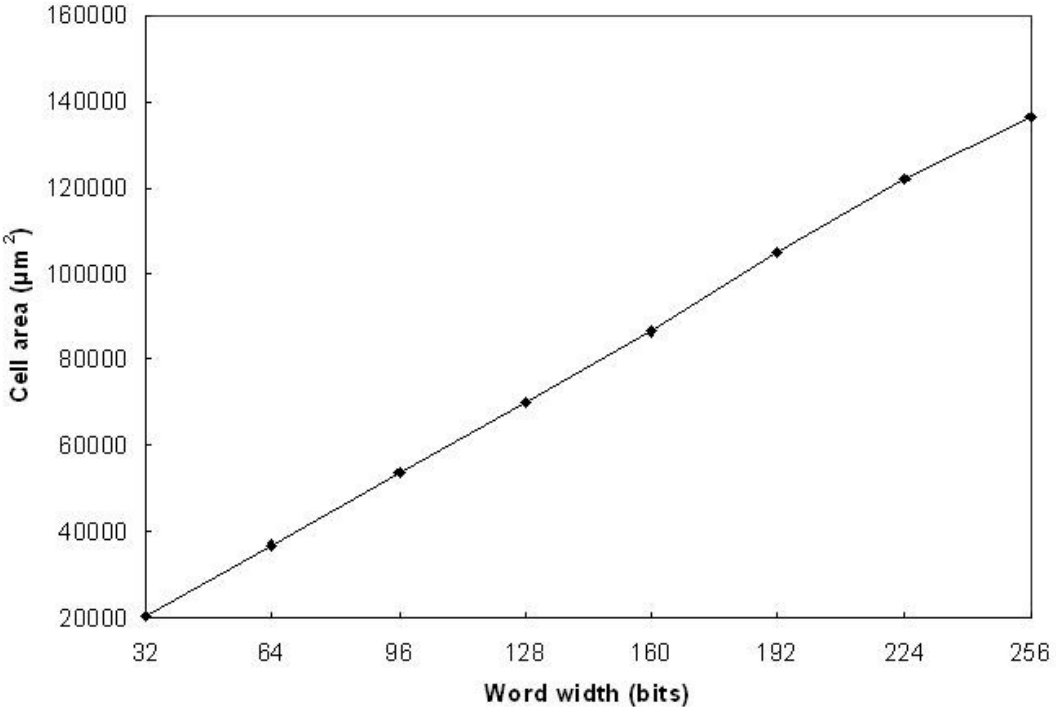

Figure 17: Total Cell area for varying word width

9. Conclusion

In comparison with Aethereal NoC, Aelite NoC offers only guaranteed services which has less complexity with less control signals due to the absence of link level flow control signals. It has better performance 1.2 X the operating frequency and cheaper having 4 X the lesser area. The mesochronous architecture allows the clock phase differences inside the NoC and GALS design supports multiple clock domains on the same chip. This project would be extended for future work accommodating link width conversion in the each component wrapper architecture.

Bibliography

[1] A.Radulsescu et al., “The Efficient on-chip NI Offering Guaranteed Services, Shared memory Abstraction, and Flexible Network Programming”, IEEE Trans. Computer-Aided Design of Integrated Circiuits and Systems,.2005.

[2] I.Cidon, I.Keidar, “Zooming in on Network –on-Chip Architecture”, Technion, CCIT Report, 2005.

[3] K.Goosen et al., “The Aethereal Network on Chip: Concepts, Architectures, and implementations”, IEEE Design and Test of Computers, 2005.

[4] P.Guerrier, and A.Greiner, “A generic architecture for on-chip packet –switched interconnections”, Proc. Design Automation and Test in Europe Conf., 2000.

[5] T.Meincke et al., “Globally asynchronous and locally synchronous architecture for large high performance ASICs”, ”,In Proc.ISCAS, 1999.

[6] A Comparison of Network-on-Chip and Busses,. http://www.arteris.com/ noc_whitepaper.pdf, 2005.

[7] D.Sylveste and K.Keutzer, “Getting to the bottom of deep sub micron.”, Proc. IEEE/ACM ICCAD, 1998.

[8] W.J.Dallyl and B.Towels, “Route packets not wires: on-chip interconnection networks”, Proc DAC, 2001.

[9] K.Goosens et al., “Networks on silicon: Combining best-effort and guaranteed services”, DATE, 2002.

[10] A.Hansson et al., “A unified approach to mapping and routing on a Network-on-Chip for Best-Effort and Guaranteed Service Traffic”, CODES+ISSS, 2005.

[11] L.P.Carloni et al., “Theory of latency insensitive design”, IEEE TCAD, 2001.

[12] D. Fritz, “Synchronous interconnect is hitting the wall,” Silistix Ltd., San Jose, CA, October 2006, http://www.soccentral.com/results.asp?CatID=488&EntryID=20698.

[13] A.Edman and C.Svensson, “Timing closure through the globally synchronous, timing partitioned design methodology”, Proc. 41 DAC, 2004.

[14] A.Hansson et al., "Undisrupted quality of service during reconfiguration of multiple applications in networks on chip", In Proc. DATE, 2007.

[15] T. Ahonen and J. Nurmi, "Synthesizable switching logic for network-on-chip designs on 90nm technologies," in Proceedings of the 2006 International Conference on IP Based SoC Design (IP-SOC '06). Design and Reuse S.A., 2006.

[16] D. Peiliang et al., Multi-clock driven system: a novel VLSI architecture, Proc. 4th Int. Conf. ASIC, 2001.

[17] X.Wu and M.Pedram, Low power sequential circuit design by using priority encoding and clock gating. In Proc International Symposium on Low Power Electronics

and Design, 2000.
[18] J.Mutterbach et al.,“ Practical design of globally asynchronous locally synchronous systems In Proc. ASYNC, 2000.
performance ASICs”, Proc.ISCAS, 1999.
[19] A.Edman and C.Svensson, “Timing closure through a globally synchronous, timing partitioned design methodology”, In Proc. DAC, 2004.
[20] D.G.Messerschmitt, “Synchronization in digital system design”, *IEEE* Journal on Selected Areas in Communications, 1990.
[21] M. Bekooij et al., “Predictable Embedded Multiprocessor System Design”, Proc. SCOPES, 2004.
[22] M. Bekooij et al., “Predictable Dynamic NoC based Multiprocessor System on Chip” In Proc.International Workshop on Software and Compilers for Embedded Systems (SCOPES), 2004.
[23] A.H.Ghmarian et al., “Liveness and boundedness of synchronous data flow graphs”, Tech. Report ESR-2006-04, TU Eindhoven, http://www.es.ele.tue.nl/esreports/, 2006.
[24] P.Poplavko et al., “Task-level timing models for guaranteed performance in multiprocessor Networks-on-Chip”, In CASES, Proc., ACM,2003.
[25] A.J.M. Moonen et al ,“Timing analysis model for network based multiprocessor systems,” in Proc. of the 15th Annual Workshop of Circuits, System and Signal Processing (ProRISC) 2004.
[26] L.P. Carloni et al., “Theory of Latency-Insensitive Design”, IEEE TCAD 2001.
[27] A. H. Ghamarian et al., “Latency minimization for synchronous data flow graphs”, In Proc. Of .the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007.
[28] D. Mangano et al., “Skew Insensitive physical links for Network on Chip”, In Proc. NANONET, 2006.
[29] S.Evain et al.,“NoC Design Flow for TDMA and QoS Management in a GALS Context”, EURASIP Journal on Embedded Systems, 2006.
[30] P.Wielage et al., “Design and DFT of a High-Speed Area-Efficient Embedded Asynchronous FIFO”, In Proc. DATE, 2007.
[31] I.M Panades and A.Greiner,”Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures”, In Proc. NOCS, 2007.

Abbreviations

BE	- Best Effort
CDC	- Clock Domain Crossing
CMOS	- Complementary Metal Oxide Semiconductor
EOP	- End Of Packet
FIFO	- First in First out
FSM	- Finite State Machine
GALS	- Globally Asynchronous Locally Synchronous
GS	- Guaranteed Services
GT	- Guaranteed Throughout
HPU	- Header Processing Unit
IP	- Intellectual Property
NI	- Network Interface
NoC	- Network on Chip
OHE	- One Hot Encoder
PI	- Port Interface
PIC	- Port Interface Controller
PVT	- Process Voltage Temperature
SDF	- Synchronous data flow
SoC	- System on Chip
STU	- Slot Table Unit
TDM	- Time Division Multiplex
WCCOM	- Worst Case Commercial