Scale in Chip Interconnect requires Network Technology

Enno Wein
Arteris
The Network-on-Chip Company
2033 Gateway Place
San Jose, CA 95110
Email: enno.wein@arteris.com

Abstract— **Continued scaling of CMOS has lead to a problem of scale as gates are faster than light travelling across a chip.**

Scalability used to be the hallmark of CMOS. Half the size, double the speed, half the power etc..

Today, transistors can leak as much current as they drive, and wires are no longer "thin film technology" approximated by plate capacitance over ground. Today wires are much thicker than wide and have significantly more capacitance (-coupling) with their neighbors than over ground.

A "short" wire (from one gate to a neighboring one) can be a stub of a few 100nm, while a long wire can connect an IP block with a processor one centimeter away. That is a factor of 100000, which represents a problem of scale and requires fundamentally different solutions. Scalability can be addressed by scaling existing techniques, while problems of scale require new approaches.

We discuss problems of scale in the context of chip interconnect.

I. Introduction

Scalability has always been the hallmark of CMOS. CMOS has replaced bipolar technologies for digital processing (of course by way of PMOS and NMOS) mainly because of its scalability and simple operation. In a new process (-node) almost every parameter changed for the better: Faster, less power ... If you couldn't easily get it to work, one option was often to use the next technology and many problems were solved. Or throw an additional metal layer at it.

Wires which used to be thin film technology and were thus accurately approximated by plate capacitance over ground. Today wires are much thicker than wide and have significantly more capacitance (-coupling) with their neighbors than over ground. In a new process (-node) almost all transistor parameters used to be better than in the one it replaced while not changing fundamentals thus making future performance predictable. As we're moving close to atomic dimensions transistors exhibit statistically significant variations of parameters like drive strength due to a finite number of carriers in the channel and non-linear temperature coefficients.

Continued scaling (usually by a factor of two each second step) over time lead to drastically changed parameters.

A. The Problem Of Scale

For a few percent of change, usually scaling is not an issue as solutions and algorithms can easily be tweaked. Larger percentage (say, 50% to a few hundred%) typically means trouble but solutions can still be adapted with potentially significant effort. Once scaling reaches one to two orders of magnitude, solutions, architectures and methods no longer apply and have to be fundamentally changed.

The beauty of early ASIC technologies (e.g. gate array) was that wires (resistance, even capacitance) were so little a factor that they could simply be ignored in the design process. Scaling technology has only a few years later led to a dominant problem of scale: From being insignificant and ignored, wire delay has moved up to be significant, even dominant by far. Completely new techniques (e.g. timing driven place and route, physical synthesis) were invented to solve the inherent chicken-and-egg problem of not knowing the delay of a wire due to physical design yet needing the information to design the system in the first place. Between 1micron technology and 65nm/45nm the scale is about 20. Since chip sizes have not changed but feature sizes have, the average ratio of chip level wires to short wires has changed 20 fold. Today, a short wire connecting to adjacent cells can be a stub of only a few 100nm, while a long wire connecting an IP block to a processor can be one centimeter long. That is a factor of 100000, which represents a problem of scale and requires fundamentally different solutions for the short wire (e.g. physical synthesis) than for the long wire (communication protocol).

B. Short Conclusion

While local wires and transistors have scaled nicely (faster, lower power), global wires have not. Instead they introduce high amounts of unpredictability in all aspects of design - including schedule. The scale between the two is many orders of magnitude. Synthesis, timing driven place and route flows with physical re-synthesis work very well for local scenarios and achieve reasonable efficiency (in terms of circuit-speed, turn-around time, schedule predictability etc.). For global scenarios these flows do not work at all and usually are not even attempted. For global wiring, entirely different approaches are used, which rely more on actual construction of the interconnect, including length-dependent pipeline stage insertion etc. Global wires are no longer measured in nanoseconds or delay but rather in clock cycles because the delays introduced easily exceed several clock cycles.

1-4244-9707-X/06/$20.00 ©2006 IEEE

This paper suggest that a packet-based switched interconnect offers an architectural solution to the global wire problem.

Enno Wein
Hot Summer of 2006

II. Physical problems of Scale

A. Copper and Aluminum

Only a few years ago, a problem with continued scaling was exemplified by the need to switch from aluminum to copper for interconnect due to aluminum's higher resistance. In the beginning of ASIC resistance was not a factor (typically was not even considered, CMOS was a purely "capacitive" technology). But due to continued scaling, current density increased to the point that the resistance of aluminum interconnect became a significant limitation. Tweaking of parameters (e.g. increasing wire cross-section) no longer helped so a fundamental change had to be made: The industry moved copper wires, despite significant technological challenges and cost (E.g. Tungsten Plugs to prevent copper diffusion and reduce contact resistance). Since the introduction of copper (about .25um node) we continued scaling, we're now at 65nm moving towards 45nm. We've already used up the copper advantage and we're running out of lower resistance metals.

B. Crosscoupling

Crosscoupling (ratio of coupling capacity between neighboring wires to ground capacity of each individual wire) scales between process nodes by the square for the capacitive part of the equation and additionally with increasing resistance. For short wires, cross-coupling is not a factor but for long wires it has become the overwhelming part of the delay equation. A major portion of the actual cross-coupling (delay and noise) is unpredictable because it depends on the switching of neighboring signals (Miller effect or Miller number). With inductance becoming more important, even switching of signals further away than from immediate neighbors becomes a factor.

C. Processor speed and ASIC speed

In the "early days", processors and ASICs ran at similar speeds. It was not untypical to design an ASIC of 25MHz on a workstation with a 40MHz processor. (Timing analysis was done by manually counting the number of elements in a path and keeping all branches of a clock tree similar) Today we have reached 3.4GHz in processors, but ASICs are typically not clocked at more than 200MHz. The reason is not that ASIC is using slower technology (current ASIC technology is 90nm moving to 65nm) than processor (current standard offering processors are 130nm and 90nm). The reason instead is design. Processors use more aggressive datapath designsand avoid long wires. (Of course there's more techniques that make processor fast, but controlling wires is predominant)

D. Block versus global speed

ASICs on the other hand are typically limited not so much by the speed with which individual blocks can be clocked (process technology paired with aggressive, mature synthesis software and good design style from years of experience enable quite fast modules) but by the speed the global interconnect between blocks can run. Long wires are of course the main limiting factor, worsened by the unpredictability of physical effects during the design/synthesis steps (actual length, capacity, resistance, cross coupling, neighbor switching etc.) which are typically overcome by over-design of driver strength, repeater stages etc. . All of those effects typically slow down the top level of an ASIC much more than the individual modules or IP's.

E. EDA industry solves problems of scale

Some specific examples of theses solutions are [1]

- Design and quantization (Mathlab)
- Simulation and modelling (CoWare SystemC)
- Entry, Control and Dataflow synthesis (Bluespec System Verilog)
- Hardware implementation (Synopsys Design Compiler)
- Physical implementation (Magma Blast Platform)
- Verification (Cadence Assura)
- Test (Mentor ATPG)

Each of the above are examples of solving a problem of scale after significant scaling (e.g. synthesis replacing schematic entry when designs became too big) has rendered previous solutions too cumbersome, inefficient or simply no longer feasible. These of course include all aspects of design and manufacturing.

F. Network-on-Chip to solve Interconnect Problem of Scale

The above examples point out the need for a System (or Top-) Level interconnect solution which addresses these problems of scale. A top level interconnect solution must reduce the number and length of global wires while improving the speed from the 100-300MHz range into the 1GHz range.

This can be achieved by a network based on point-to-point, single fanout wires that can be kept short utilizing a hierarchical, distributed, localized switch fabric.

III. System Scale

The point-to-point switch fabric pointed out in the last chapter has system implications which will be analyzed in this chapter.

A. Global Control and Arbitration

Typical ASIC design style is state machine based, which is perfect for block design and due to legacy of scaling and integration of "old" designs has been carried over to top level design as well. Top Level design has until recently not been given special consideration, it has only been an extension of block design at the next hierarchical level. Due to the effects of long wires and top-level timing, arbitration and

[1]Disclaimer: The author is not trying to classify EDA software into "buckets" - rather just giving a few examples of problems of scale and software solution. There are more problems that have been solved and there are more software solutions for the same and other topics. By mentioning a specific solution or vendor, no implication is made as far as quality, preference, market or any other facet is concerned

global control (state memory) have become major limiting factors both in terms of speed and complexity. Since control and arbitration require at least one global signaling process against and orthogonal to dataflow it adds one dimension of complexity to the timing problem. Inserting pipeline stages where needed to break clock cycle problems is not possible without causing significant changes in global control state machines. Typical bus based top level designs suffer from these as busses rely on complex arbitration, control and timing schemes to work properly.

B. Algorithmic Depth

First integrated Circuits contained a single gate, for example an "AND" gate. For packaging and price, several were integrated into a package, for example 3 "AND" gates in a DIL package with 12 connections. The algorithmic depth of this is very low, it is a very predictable, deterministic system as under all circumstances it is very clear what the result is for a given input and even the time it takes the system to respond is very predictable. A complex, current algorithm which can be synthesized and verified in itself contains several ten-thousands of those in equivalent gates and over the last twenty years an entire industry (EDA) has been formed around solving the resulting problems of scale. Synthesis, Layout, Timing etc. software worth many millions of dollars has been developed and sold, many companies have grown large and powerful or failed and disappeared just based on solving very specific, individual parts of the resulting problems of scale (E.g. Synopsys in the case of synthesis over schematic entry). The problem of scale (enabled by process scaling) which has been solved here is the one of algorithmic depth - in other words how many atomic sub-pieces are required to express a single algorithm, for example a deblocking filter for an H.264 decoder.

C. Determinism

By Scaling of the first designs (15 years ago, small ASICs implementing direct algorithms like for example a Reed Solomon chip) to today's complexity by one to two orders of magnitude (E.g. Complete 4th generation wireless base station chip) we have run out of scalability - several parameters have changed so much that a different kind of solution has become imperative. In this case, one major difference is determinism.

1) Simple Algorithms: ASIC/chip technology used to be fully deterministic in the sense that they implemented small algorithms with clearly defined, simple input and output data which was usually very simple in format and content. The algorithms were direct and straightforward and directly implemented in hardware. The clock cycle accurate behavior was exactly predictable due to the directnes of the algorithms.

2) Complex Algorithms: Today's System On Chip typically implement multiple layers of communication, integrating hardware and software components and are usually configurable to span a wide variety of parameters or even functionality. Input/Output data as well as control is usually very complex and moves back and forth between hardware and software portions of the system multiple times. Whether specific sub-algorithms are executed by hardware or software often dynamically varies depending on parameters and context.

3) Input and Output Streams: The input/output stream of a Reed Solomon chip is very deterministic and the algorithm can be expressed deterministically with state-machines, local storage and the input/output behavior can be predicted exactly (including exact clock cycle delay etc.). Often those algorithms relied on a single piece of data being present every single clock cycle and would in turn deliver a single piece of data on every clock cycle at the output.

An entire wireless base station system is fundamentally not deterministic in the same way. It is impossible to predict under all circumstances the exact clock-cycle accurate input/output behavior of the system - nor is that expected or even necessary. (This is not to be confused with cycle-accurate simulation in RTL or gate level, which can of course be done and will be correct, but is not useful for system development as the complexity would necessitate overly long simulations especially when including actual software and operating system running on processor cores)

D. Determinism of the Interconnect

The system designer of today's systems is usually not interested in clock-cycle behavior. Instead it is important for the system to ensure certain requirements (latency, throughput, power consumption etc.) are met and a competitive solution is created (cost, features, functionality). The result is that at hardware level (Top level of the actual chip) the system is not deterministic in terms of actual clock cycles and even order of events. For example, a CPU core can ask for a memory block a little earlier or a little later with respect to a stream of digital video data coming into the device from outside because those processes are fundamentally asynchronous in nature (not necessarily in a clock-domain way, but from the system point of view).

Of course one of the goals of the system operation is to synchronize the processes (the CPU will perform certain operations every time a specific event occurs in the incoming data) but even that synchronization will be at a transactional level. In other words it is not necessary nor desired that the CPU performs a tasks at an exact clock cycle. Rather it has a time window in which tasks need to be performed in order not to disturb system performance.

Specifically, most CPU processes are interrupt driven, which means that their execution is a complex system of interrupt-creation, interrupt masking/handling, context switching, loading of application code, loading of data, execution while loading potential additional data, storage of results etc. .

Each of those steps are systematic operations in themselves often sharing the same resources (e.g. memory subsystem, chip interconnect) with the original process (which likely continues other tasks in realtime). It becomes very clear that those systems are too complex to be deterministic in their exact behavior.

Thus - it seems obvious that each subsystem should not be required to be deterministic in an exact state fashion, rather it is obvious that each subsystem should be expected to guarantee (within reason) that it will be capable under all reasonable cases to perform within the required system parameters.

E. Quality Of Service "QoS"

For the interconnect subsystem (which can easily be the most complex of all subsystems) this performance guarantee is called Quality of Service. A guarantee to execute a certain operation in three clock cycles is not Quality of Service because it is not meaningful from a system point of view. A guarantee that a cpu dataflow has the lowest possible, reliable latency while a realtime traffic (e.g. voice or live camera stream) gets a guaranteed, specific throughput or bandwidth is considered a Quality of Service Guarantee. A few nanoseconds of latency are not important for the camera picture (because it would not be noticed) but the flow must not be stalled as that would be a noticeable loss or freeze of picture.

F. Development cost up, Component cost down

Today's system on chip have reached complexity (in terms of gates, transistors, functions) that exceeds that of former mainframes, yet the selling price is in the ten dollar range, while mainframes were sold for millions of dollars. The development cost of such complex systems is of course in the ten million dollar range, obviously that can only be justified by huge quantities sold, which in turn can only be achieved by many people actually liking and utilizing the product.

1) Features: The bigger the market in terms of potential buyers, the more "opinions" about the essential features of a product and the more unpredictable the required feature set. This results in faster changing product cycles and the changes are more fundamental. A new generation of mainframe had a few new/changed instructions based on the input of a handful of users based on their thorough usage experience. A typical SoC has a feature set trying to capture large markets with very unpredictable behavior, competing products and constant adjustments to emerging standards, hype etc. . For example a typical change from one revision of a mobile phone system to the next is to add a digital camera. That's like saying the next version of mainframe computer should be able to walk and the marketing person in charge would have been walked on the spot.

This results in frequent, fundamental changes of systems which require exchange of entire subsystems with potentially significantly changed system behavior. Standard top level design practices (ad hoc communication design linking one block to another or bus systems) fail to implement the required flexibility in a timely, reliable and predictable manner because each change which can cause changes in the exact timing and order of events can break the global control and arbitration processes/flows and expectations.

IV. Packet Based Network

The packet based network is a solution to the above mentioned problems of scale, mainly due to a single quality:

A. Self-contained packets

Because each packet in a point-to-point packet-switched network carries all required information (sender address, receiver address, quality of service etc.) there is no requirement for global control. No process is required that keeps track of exactly where a packet is, how it is routed, how long it takes, who sent it and so on. The exact (clock-cycle-) time that a packet arrives at the receiver is not important and therefore not required.

B. Local Arbitration

The Quality of Service class of the packet is evaluated at each point in the network for local arbitration. All control for arbitration is distributed to local switches which decide based on quality of service and age of packets their priority with respect to others and thereby determine the arbitration of shared resources.

This eliminates any need for time consuming, error prone and wiring intense global arbitration.

C. Determinism through Quality of Service

By specification of quality of service for traffic classes and appropriate network design, exactly the appropriate level of determinism is achieved without the requirement of individual busses for low latency or guaranteed throughput by over-designing busses.

The system's need for the right data in the right time window (rather than at a specific, but randomly determined irrelevant clock-cycle) is exactly fulfilled as long as the network is designed to actually guarantee the required quality of service parameter.

D. Flexibility in the (re-)design process

A network which does not rely on exact clock-cycle behavior, but rather the expected information to be present in the appropriate time window is flexible for even major changes as long as the system as a whole has the appropriate information sources and sinks and the quality of service expectation and needs are fulfilled.

Even a significant change does not affect a part of the system which is not related to it because no expectation for cycle-accurate behavior (which could be disturbed by the unrelated change) exists.

In other words, entire subsystems can be removed, added or exchanged without upsetting the system. Just like in Internet: A user can come online, an entire country can be added or drop out temporarily - that does not affect unrelated traffic.

V. Scale in real life

A. Kitchen and Living Room

To walk from the living room to the kitchen (assuming a regular size house) takes about 20 seconds. A typical Silicon Valley commute of 30 miles would take all day to walk at 3 miles per hour each direction. Opening the garage door, getting in and out of the car, starting (mirrors, belts...) takes one or two minutes, but the car's speed quickly makes up for the

startup time over walking - even in Silicon Valley commute traffic.

B. California and Europe

A return-trip from California to Europe of about 12000 miles would take four weeks by car (if there was a road), require three oil services and one set of tires. While it takes a few hours to get to the airport, check in, security , wait for the start, potential connecting flights etc. - still the plane beats driving by far. Even the cost is significantly lower than using the car ($1000 for the return ticket), while the car would cost more just for gas. (Not even mentioning value-loss, service, tires and sustain the human life for four weeks)

C. Driving to Europe

This example points out that there's startup (cost and time) for going to the next level of transportation, but due to orders of magnitude of difference in speed (actually efficiency) that cost quickly amortizes if the solution fits the scale of the problem. Staying with the lower form for too long would become prohibitively inefficient very quickly. On the other hand it is easily obvious that using the higher level of transportation for too low a problem would be foolish (flight to shopping center, car to the kitchen).

This also shows that in many cases even a scale mismatch of only one order of magnitude means that there's no solution at all.

VI. Scale in Multiprocessor systems

Cache Coherency is one of the big topics in multiprocessor systems. Many clever protocols have been invented (Berkeley, Firefly, Dragon, Illinois, MOESI etc.) but the main idea is typically bus snooping - in the sense that one processor "listens" to the bus activity that another processor initiates to determine whether its own copy of a datablock is clean (or actual) or no longer valid as another processor has updated the data contained in the cache block. A system like that works well for a few, small processors with few threads, but is not scalable to large systems with many concurrent, asynchronous operations. The system also raises security concerns in systems which are more and more complex, software operated and likely targets for malicious code or at least prone to bugs as data seperation cannot be guaranteed. The multi-faceted communication and state keeping is complex in itself and when scaled to more elements becomes very hard to maintain and verify.

VII. Summary

Today's electronic systems are being integrated on one or a few circuits and contain multiple processor/software elements along with a plethora of hardware, configurable IP, specialized processor, firmware, memories and high-speed connectivity with high level protocols. Trying to hook them all up with simple busses or ad-hoc wiring is like trying to drive a car across the atlantic.

Due to several different effects of scale as explained above, individual busses and wires are way too complex, error prone, unpredictable in terms of signal integrity, slow, hard to verify etc. At the same time the actual system operation has grown in complexity to being very indeterminist in nature. Systems can no longer rely on bus-techniques like snooping therefore the need for the clock-cycle determinism and functionality of direct busses and wires has gone away, at least at the higher levels of SoCs.

A. Quality Of Service

Instead, a new need has arisen - the need for quality of service of communication classes between functional units, without being bogged down by individual protocol or speed mismatches.

Quality of Service (QoS) means that each part or user of the system has specific communication requirements which can potentially be quite different in nature:

- Guaranteed Throughput This means that the traffic class has to transport a certain amount of data in a given time period and it would be very detrimental for the system (e.g. picture dropping out) if this amount could not be guaranteed. Latency is typically not very important in this class as long as the time window of throughput is not violated.
- Low latency This is the "Firefighter" class - the traffic has high priority and needs to get to the desitination as soon as possible. It is important that the amount of traffic for this class (throughput) is fairly low otherwise it would block other traffic heavily (imagine all trucks on the road having firefighter privileges, we'd be standing in traffic even more)
- High Bandwidth Lots of traffic - but no particular latency or throughput guarantee, sometimes called "best effort".

Years ago, this need has come up with the rise of global communication, internet and the growing connection and abstraction of different communication means (telephone, e-mail, high speed data like video etc.) and scalability (Many Millions of users). The only feasible (because scalable and universal) way has shown to be a packet-based network (the internet).

It becomes increasingly obvious that the packet-based network is also the most useful solution for the increasing need for universal, scalable and reliable communication for today's systems, which are typically integrated onto a single chip.

Therefore, the initial cost of packetization in terms of latency (checkin and security lines at the airport) and packet overhead (pilot deck, flight crew) is much less than the increased cost of system maintenance, signal integrity related project dalays, verification overhead (tires, gas, maintenance, cost of living, risk when driving a car across the atlantic).

VIII. Conclusion

The Packet based Network on Chip can be scaled to any size and is therefore future proof.

A. Standard IP socket formats

Standard IP socket formats (e.g. OCP) are a good start to make IP's and processors universally useful and provide

for easy adjustment to individual system's needs rather than standard bus formats and protocols.

1) IP reuse: Often standard bus protocols (e.g. AXI) can be redefined to standard socket and therefore reuse IP that is based on the former bus standards without change.

The packet based network-on-chip enables the mix-and-match of different protocols for attached IP's as it inherently translates the protocols while packetizing and de-packetizing.

B. Timing

Due to the point-to-point nature of a packet switched network (like WAN, LAN) and the abscence of global control schemes (fully self contained packets including all signalling, flow control etc.) the system is resilient against any kind of timing related issues. Therefore, longer wires, higher capacity, higher resistance or any other effect can easily be dealt with by inserting appropriate pipeline stages without disturbing system functionality (of course adding latency, though).

C. Wire Efficiency

Point-to-Point wiring is easy to handle in physical design and can therefore be efficiently driven to much higher speeds (e.g. 1GHz in 90nm) achieving much higher wire effiency than standard techniques. Obviously, a link which runs at 1GHz needs only 20% of the wires of a link which runs at only 200MHz to achieve the same throughput. The actual real-time latency is the same even if a few pipeline stages are added in the higher frequency link.

D. Signal Integrity

Since cross coupling related signal integrity are not frequency related, rather length, resistance and slew rate related a shorter, faster clocked link with fewer (spacing!) wires is less prone to signal deration than the low frequency, more parallel wires standard one.

E. Power Saving Modes

A point-to-point, packet based network lends itself perfectly to advanced power saving techniques like voltage islands and power-off regions since it provides a very structured and clean communication to islands and regions. Outstanding transactions can be tracked easily and hence ensure that regions are only powered down when there's no more outstanding transactions required to complete from the to-be-powered-down domain. The same goes for the case when a domain/IP/region is actually powered down. The network can naturally protect against faulty requests to a powered-down region which would otherwise cause a system to hang or crash.

F. Power Consumption

In terms of power consumption there are several factors to be recognized. The first is that a point-to-point network sends traffic only to exactly the correct target, therefore only the actual target and the link to the target toggle. In a multi-fanout system all attached targets toggle and need to perform at least address checking to see whether a broadcast is for them.

The second factor is frequency. At first glance higher frequency means higher power consumption. While that is correct, it is really the amount of actual information that counts (due to the toggling of wires) that determines the actual current requirement. Therefore, two links with the same throughput and data, but one running at high speed with few wires and one running at low speed with many wires will consume roughly the same amount of power. Since fewer wires have not only proportionally less capacitance (due to them being fewer) but can be spaced further and therefor have less coupling capacitance, the high speed link should actually come out ahead in terms of power consumption.

G. Reliability and yield

Since a wire, which is not needed and therefore not built cannot break, the yield of faster systems with less wires will be higher and with the same argument the long term reliability will be higher.

H. Redundancy

The resources saved by higher wire efficiency can partially used or redundancy. In a packet based network with configurable routing through switches, faulty links (for example after metal migration) can be replaced by a redundent counter-part and therefore the system's reliability and life span can be increased significantly.

I. Graceful degradation

Often redundancy can be achieved not only by additional links but by simple multi-route topologies which use all routes while the system has no faults and in the presence of faults reduce some quality of service classes while still guaranteeing others. This means that a device instead of failing completely after the first breakage or fault could instead only reduce features (e.g. movie camera no longer operational in the cell phone, but the phone function is still operational).

J. Asynchronous operation

Since packet based networks are fully self-contained and there's no need for global control or arbitration they lend themselves naturally for asynchronous operation. That means that synchronization elements (GALS - Globally Asynchronous, Locally Synchronous) can be inserted at any point in the network without loss of control (just single clock-cycle latency).

1) Clock Uncertainty: This enables elimination of clock uncertainty problems related to process/temperature/voltage related local speed differences. Clock uncertainty is a significant design methodology headache in large deep-submicron systems.

K. Latency

A main perceived drawback to packet based networks is increased latency due to packetization and routing.

On a typical chip (12mm or so on a side) long, global connections can easily measure 10mm - that translates to a delay in the range of 10 nanoseconds. At one gigahertz of system operating frequency that is equivalent to 10 clock

cycles. Packetization and de-packetization take about 2 clock cycles each, a network hop (switch) takes about 1-2 cycles each including driving a length of wire. Overall, therefore the additional delay necessarily introduced by a packet based network hardly exceeds the physical delay of a long connection.

Significant latency can be introduced by arbitration (e.g. in bus systems) or routing methodology, see next chapter.

L. Packet Routing Styles

1) Store And Forward: The internet as the premier example of packet switched networks uses store-and-forward routing because the latency introduced by storing an entire packet is in similar order of magnitude of the time it takes the packets to travel across the network anyways (limited essentially by the speed of light or electrical signals over up to several thousand miles).

2) Wormhole Routing: For on-chip application, the store-and-forward routing style would introduce too much latency as a packet is potentially several ten to hundred clock-cycles long but the network is in the centimeter range in size. Therefore on chip, a low-latency routing technique must be employed. In a wormhole routing scheme, the first word of a packet contains the receiver address and the quality of service information. The routing decision is immediately taken based on that information, thus introducing only one clock cycle of latency for the operation.

3) Cut-Through Routing: Since in wormhole routing the packet is typically ”longer” than the network (i.e. the receiver is already receiving the first words of a packet while the sender is still sending the last words of the same packets) all intermdediate, shared network resources (e.g. links) are blocked by that traffic.In order to prevent stalling other traffic, additional memory can be used to enable buffering of intermediate packet contents. That does not change the routing decision, it is still wormhole (the first word determines the routing decision which is immediately taken) but it enables a packet to ”cut through” the network even in the event of intermittent stalls.

REFERENCES

[1] H.B. Bakoglu, IBM Corporation, *Circuits, Interconnections, and Packaging for VLSI* Addison-Weslay Publishing Company 1990.