Delay Insensitive System-on-Chip Interconnect using 1-of-4 Data Encoding

W.J. Bainbridge, S.B. Furber,
Dept. of Computer Science, The University of Manchester,
Oxford Road, Manchester M13 9PL, UK.
{bainbriw, sfurber}@cs.man.ac.uk

Abstract

The demands of System-on-Chip (SoC) interconnect increasingly cannot be satisfied through the use of a shared bus. A common alternative, using unidirectional, point-to-point connections and multiplexers, results in much greater area requirements and still suffers from some of the same problems.

This paper introduces a delay-insensitive, asynchronous approach to interconnect over long paths using 1-of-4 encoded channels switched through multiplexers. A re-implementation of the MARBLE SoC bus (as used in the AMULET3H chip) using this technique shows that it can provide a higher throughput than the simpler tristate bus while using a narrower datapath.

1. Introduction

The major challenge that faces designers of System-on-Chip (SoC) integrated circuits is achieving the required functionality, performance and testability whilst minimising design cost and time to market. The key to achieving this goal is a design methodology that allows component reuse. Such methodologies have been an enabling factor in the success of the fabless vendors of intellectual property, such as ARM Ltd, who license designs of the same processor core macrocells to many competing semiconductor manufacturers.

These design methodologies rely upon the use of a standardised interconnection interface (usually some form of shared bus) for connecting the many component blocks together to form an on-chip system. This system-level interconnect presents increasing difficulties as feature sizes are reduced for a number of reasons:

- small feature sizes allow larger designs, and so more functional units can be incorporated into a single chip placing greater demands on the interconnect;
- wires are taller but narrower and may be placed closer together, leading to increased coupling (crosstalk) between wires;
- narrower wires have higher resistance, leading to slower signal edges and longer interconnect delays.

The increases in wire resistance and capacitance mean that wire delays are becoming more significant compared to gate delays as process feature sizes shrink. This means that isochronic fork or equipotential assumptions are no longer valid for long interconnecting wires, further complicating the design and validation of long, high-performance tristate buses be they synchronous or asynchronous in operation.

Instead, it may be preferable to use a gate multiplexed approximation to a bus as in figure 1, which uses more wire (and control logic) overall, but each wire is shorter and presents less load to its driver. However, even with this scheme the wires can still have significant length and thus incur substantial crosstalk and resistance effects.

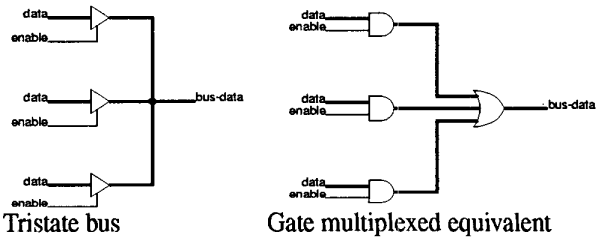

Figure 1: Tristate/multiplexed bus construction

2. Crosstalk

As an example, minimum width wires at minimum separation on the 0.35 µm process used for the AMULET3 chip had a lateral capacitance of 30pF/mm between adjacent wires on the same layer, an interlayer capacitance of 130pF/mm and a resistance of 100ohms/mm in the worst case. Even with this slightly outdated process technology the dimensions and relative positions of long wires can have a significant effect on the signals they carry.

The severity of, and theory behind, these effects is discussed elsewhere [3,8,12] but a simple SPICE simulation of a group of wires such as was performed during the MARBLE bus design [4,5,6] is sufficient to illustrate the problem. Such simulations show that when a wire is surrounded in both horizontal and vertical directions by other wires (as can occur with the most naive wire layout, or with automatically routed wiring) the activity on these wires can have a significant effect on the surrounded wire.

1522-8681/01 $10.00 © 2001 IEEE

The three curves in figure 2 show the nominal case delay when only one wire is switching (middle curve), the worst case delay experienced when 1 wire switches (at the same instant in time) in the opposite direction to all of its neighbouring wires (upper curve), and the best case delay experienced when all of the wires are changing level in the same direction at the same time (lower curve).

The delay can be reduced by increasing the wire width, which gives a linear reduction in its resistance, with a less significant increase in the capacitance [17] since much of the capacitance in deep submicron processes is due to fringing effects. Increasing the separation of wires on the same layer and avoiding running wires on adjacent layers in close proximity allows the capacitance between wires to be reduced.

In AMULET3H [2,9] the processor memory bus (9mm in length) used only one metal layer with wires twice the minimum width and separation, and the MARBLE system bus (5mm in length) used only two of the metal layers, again with a wider spacing. However, layout modifications to reduce wire delay consume considerable silicon area and are only feasible up to a point.

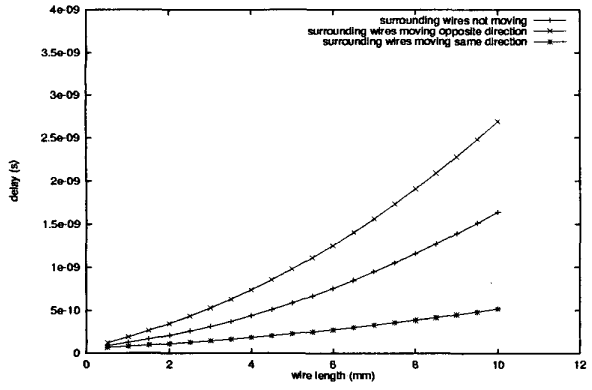

Figure 2: Signal propagation delay in 0.35μm

Possible variations in delays in long wires, as illustrated in figure 2, caused by crosstalk make bundled data systems difficult to design, requiring the inclusion of large margins which seriously affect the performance of the system. More than 1ns of the 12ns MARBLE cycle time is the delay element to allow for crosstalk variations. Synchronous design approaches face a similar timing closure problem since the clock frequency must be chosen such that the receiver is told that data is stable at the correct moment.

3. Improving Interconnect Performance

Techniques applicable to increasing the performance of the interconnect include: inserting repeaters/pipeline latches into the datapath; splitting the datapath into a number of independent, narrower datapaths; using a delay-insensitive signalling scheme to avoid incurring the worst-case delay; and using transition signalling to avoid the penalty of a return-to-zero phase. These issues are discussed in the following subsections.

3.1. Repeaters and pipeline latches

Building the fabric of a shared bus using multiplexers means that all signals are unidirectional allowing careful management of their delay through the insertion of repeater amplifiers (buffers) at regular intervals along their path [3,8]. As an example, (from figure 2), a 10mm wire has a nominal delay of 1.6ns whilst a 2mm wire has a delay of 0.2ns. An inverter with a fan-out of three has a delay of 0.15ns in this technology. Inserting four inverters into the path to amplify the signal every 2mm gives a total nominal end-to-end delay of about 1.6ns.

Amplifying the signal at intervals along the line also has a beneficial effect on the variation in the delay between the slowest case and fastest case, reducing the ±1ns variation to ±0.75ns. This variation in the delay for the whole path must still be allowed for in the choice of clock frequency or delay-margin.

A higher thoughput can be obtained by using a pipeline latch instead of the amplifier/inverter to both amplify the signal and spread the link delay over multiple pipeline stages. Splitting the 1mm wire into five pipeline stages thus gives a 5X improvement in throughput with minimal latency penalty, since we are replacing a forward going amplifier with a latch.

3.2. Data-path width

Unfortunately, adding latches into a wide single-rail datapath causes problems because of the high load on the latch-enable signal. This means that either the circuit runs at low power and low speed, or, to operate at high speed, appreciable power must be consumed to open/close the latch. For a delay-insensitive (DI) encoded datapath, there is no 'wide latch enable' signal, but instead the synchronisation of the acknowledge signals from the many bits of the datapath latch introduces considerable delay. Breaking a wide datapath into a group of narrower datapaths as illustrated in figure 3 (for a 12-bit datapath split into 3 separate channels), and only synchronising their activity at each end of the pipeline, thus offers the possibility of operating at a higher frequency.

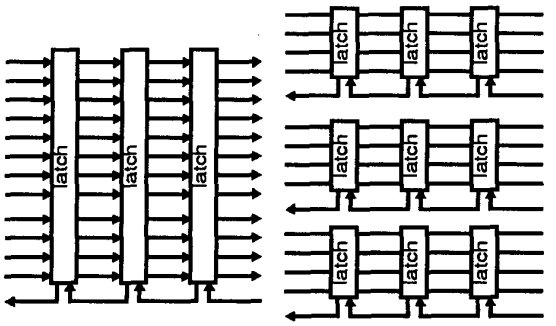

Figure 3: Reducing the datapath widths

Pipelining the datapath and partitioning it into a set of narrower links which run independently also has a beneficial effect on the peak driver supply current as discussed elsewhere [7] since all of the drivers of the outputs of different pipeline stages will not, hopefully, switch at exactly the same time as would occur with a wide datapath. The magnitude of the peak driver current, and hence the correlation of electromagnetic emissions from switching of the interconnect, will thus be reduced.

3.3. Delay-Insensitive (DI) interconnect

Delay-insensitive design methods allow the possibility of exploiting the average case delay rather than the worst case. This could amount to a significant saving with long interconnections.

Furthermore, with conventional single-rail design styles (both synchronous and asynchronous), timing analysis is required to identify the slowest wire in a group, so that the receiver can correctly be told when the data is stable (either using clock or a matched delay path). The use of automated place and route tools to meet short time-to-market further exacerbates these problems since automated layout can lead to a larger distribution of delays in a group of wires carrying related signals.

A delay-insensitive design style avoids the need for this timing analysis, giving designs that operate correctly whatever the delay in the interconnecting wires. Such techniques have previously been used for constructing asynchronous buses [13] and processor systems [11].

3.4. Transition signalling

Transition signalling, which tends to result in large, slow circuits may be a good alternative to four-phase return-to-zero (RTZ) signalling for long interconnects because each communication across the link only requires two link propagation delays, as opposed to four.

Unfortunately, whilst transition signalling offers significant power advantages over RTZ signalling, the parity function required to detect an event is expensive to implement in CMOS technology requiring the use of XOR gates which result in slow circuits with heavily loaded signals. For this reason, we only consider RTZ encodings from here on.

4. The 1-of-4 DI Data Encoding

A 1-of-4 data encoding uses a group of four wires to transmit two bits of information per symbol. A symbol is one of the two-bit codes 00, 01, 10, or 11 and is transmitted through activity on just one of the four wires.

Since it is possible to detect the arrival of each symbol at the receiver (with RTZ signalling, the wires are all low when no symbol is being transmitted) a 1-of-4 encoding is delay insensitive, as are all the other 1-of-N codes [19].

Further advantages of 1-of-4 RTZ signalling are that, if bundles are routed in close proximity:

- the likelihood that two adjacent wires will switch at the same time is much less;
- any crosstalk that does occur will be between wires switching their signals in the same direction.

Crosstalk should therefore not be as detrimental to performance for a 1-of-4 encoded interconnect as it is for a single-rail implementation.

1-of-4 systems are also attractive from a power consumption perspective, as they convey 2 bits of information using only 2 transitions as opposed to the 4 transitions that would be required if a dual-rail encoding were chosen. Given that in a single rail system, assuming random data, an average of 50% of the signals forming a microprocessor data bus change state every time a data value is transmitted, then 1-of-4 encoded RTZ has only a factor 2 worse power cost. There is also the cost of the protocol conversion of course.

5. 1-of-4 Modules

The key to constructing a high throughput interconnect system is a fast pipeline latch structure. True and complement forms of a standard-cell 1-of-4 pipeline latch implementation (without reset inputs) are shown in figure 4. Corresponding STGs describing the behaviour of these latches are shown in figure 5. In brief, the operation of the (leftmost in figures 4 and 5) latch is that:

- initially the input group is low;
- an input symbol is presented by raising one of the inputs, in[1] say;
- since xa is low at reset, in[1] passes to nx[1] causing an acknowledge on nina;

- in[1] may then fall at any time, but this will not be passed to nx[1] until xa has risen, indicating that the next stage has accepted the value encoded on the nx lines;
- after nx[1] rises, so does nina, returning the latch to its original idle state.

A chain of these pipeline latches operates with a cycle time of about 1.5ns (10 inverter delays) in a 0.35 micron CMOS technology [20] giving a throughput of just over 1Gb/s using five wires between successive pipeline latches, or 200Mb/s/wire.

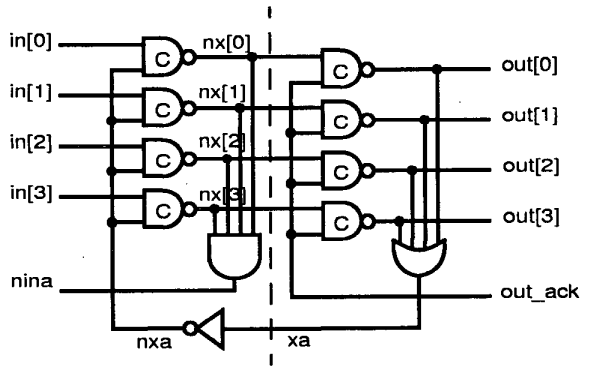

Figure 4: 1-of-4 Pipeline Latch

Ignoring the transmitter and receiver encoders/decoders, two other components (serving similar functions to Sutherland's Micropipeline elements[16]) are required for the creation of a high throughput 1-of-4 transmission system allowing the connection of more than two devices:

- a channel multiplexer (merge)
- a steering mechanism (select)

Implementations of these modules for a 1-of-4 RTZ encoded data stream are introduced in the following sections. These modules all have a 10 inverter delay cycle time (about 1.5ns), the same as the pipeline latch structure, although, to achieve this, a separate acknowledge signal is used for each bit of the datapath in some blocks. These blocks must be connected to a modified version of the latch which accepts such 'single-bit' acknowledges and ensures that they are all inactive before allowing new data to pass through to its output. Forking a data stream into two separate streams simply requires a Muller C-element in the acknowledge path to combine the two acknowledgments into one.

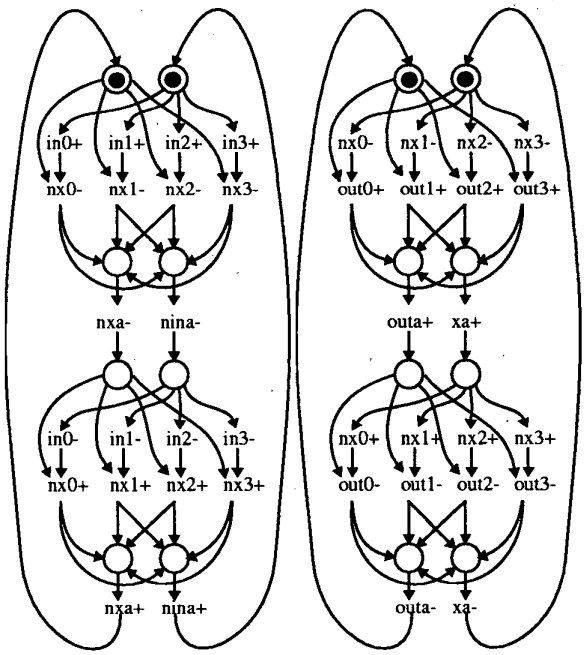

Figure 5: 1-of-4 Pipeline Latch STGs

5.1. Channel multiplexer (merge)

A merge element is used to combine two incoming data-streams onto one outgoing data stream. The datapath logic required to perform this function is a set of OR-gates. These can be seen in figure 6 which shows how pipeline latches can be wrapped around them so that the merge element still has only 10 logic inversions per cycle.

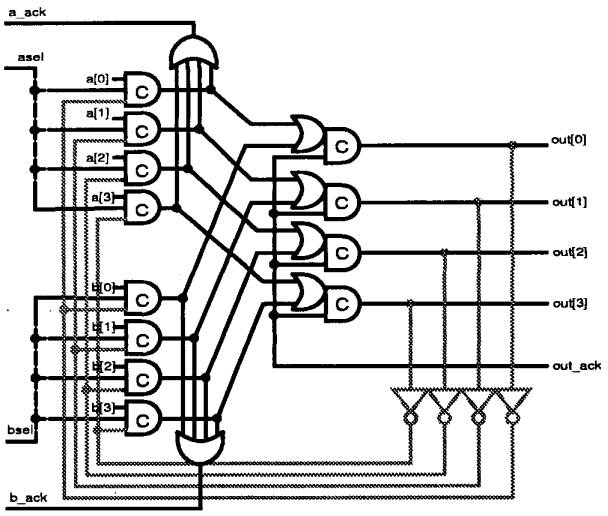

Figure 6: Merge element

The implementation shown in figure 6 has two select line inputs used to control which input (a[3:0] or b[3:0]) is passed through to the output. The select line must cycle once for each handshake on the input port. For correct operation the select inputs must be mutually exclusive. The shaded lines represent the internal bit-sliced feedback path.

5.2. Steering mechanism (select)

To control the flow of data to different destinations, an equivalent of the micropipeline select element [16] is required. The implementation shown in figure 7 has two select line inputs used to control which output the input should be routed to, requiring one cycle on the appropriate select line for each data symbol passed from input to output. For correct operation the two select lines must be mutually exclusive and there should be one handshake on the select line/input-acknowledge pair for each data symbol passed from input to output.

A bit-sliced acknowledge is used here because of the requirement for an OR-function in the acknowledge path.

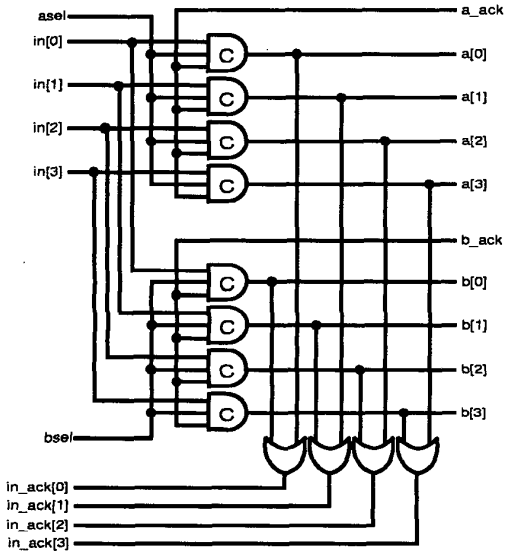

Figure 7: Select block

6. A 1-of-4 Packet Switch

A two-input, two-output switch can be constructed from the select and multiplex elements described above as shown in figure 8. For SoC interconnect needs, a higher level abstraction is required so that the quantities routed are, say 32-bit addresses. Therefore, a format has to be imposed on the symbols flowing through the system so that, for this example, a 32-bit packet is correctly routed, intact from one input to one output, and not split across outputs or interrupted by independent activity on the other switch input.

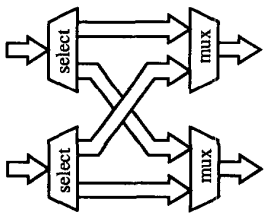

Figure 8: 2x2 Switch

The following subsections discuss:

- a suitable packet format, including a header which can be extracted locally at the switch and used to control the routing of the packet;
- a packet arbiter that allows for independent input streams where the mutual exclusivity of the inputs to the multiplexers in the switch cannot be guaranteed and must be enforced locally;
- a counter to keep track of the start/end of a packet, ensuring that whole packets are switched intact.

6.1. Packet format

Here we assume a simple, fixed length format where, at the input to any switch element, the routing header forms the first symbol of the packet and the payload forms the remainder of the packet.

Each switch node strips off the header symbol that it uses to route the packet to one of its (up to four) outputs. To maintain a constant packet length, the switch node appends the removed header symbol as the last symbol in the packet immediately after the final payload symbol. The format of a packet thus changes after passing through each switch node. This approach also allows the receiver of the packet to identify which sender the packet originated from by inspecting the header.

Figure 9 illustrates the reformatting of the packet in a switch node. The count inputs to the select and mux components are not identical: for a packet of N symbols, the select has to direct the first symbol (the header) to its lower port and then send the N-1 payload symbols to its upper port, whereas the mux requires the opposite count sequence since it must first accept N-1 payload symbols on the upper port and then take the final symbol (which was the incoming header) from the lower port.

A latch stage is required to store the extracted header symbol whilst the payload bypasses it. During this time, the header is used to steer the control inputs of the switch, thus routing the packet to one of the (up to four) outputs.

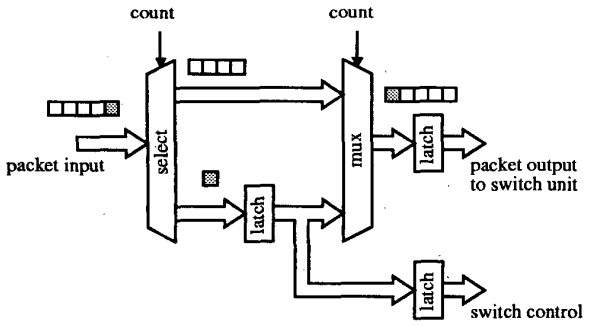

Figure 9: Switch node header extraction

6.2. Detecting the start/end of a packet

It is necessary to determine the start and end of each packet so that the header of a packet can be extracted, allowing the select and merge elements to be switched at the appropriate times. It would be possible to use some of the symbols of the 1-of-4 encoded data stream to signify the start and end of each packet. However, this reduces the effective throughput of the system. An alternative (as used here) is to have packets of a fixed length, using a counter to indicate the start/end and duration of each packet. Van Berkel's systolic counters provide the necessary functionality [18].

6.3. Packet-arbiter

If the inputs to a channel multiplexer are not mutually exclusive an arbiter is required to ensure correct operation. This arbiter must take account of the packet based nature of the transmitted data stream and must ensure that a whole packet is switched in the same direction, without interruption by a packet from a different port. A suitable packet-arbiter is shown in figure 10.

This arbiter takes incoming requests from the input data streams (possibly generated by a 4-input OR function of the data stream signals) on the arbreqn inputs. These then have to compete in a MUTEX [14] to ensure that only one of them is granted the resource for any given period of time.

The arbitration winner is latched by the cross-coupled C-elements so that a train of handshakes can be steered from the count0/count1/countack channel to either the select0/select_ack pair or the select1/select_ack pair. These outputs can then be used to steer a select mechanism such as that in figure 7 above.

The 'latch' is reset by a handshake on the count0/ countack port, after that handshake has triggered a handshake on the select output port. The outputs of the MUTEX are ignored whilst either of the cross-coupled C-element outputs is asserted (low).

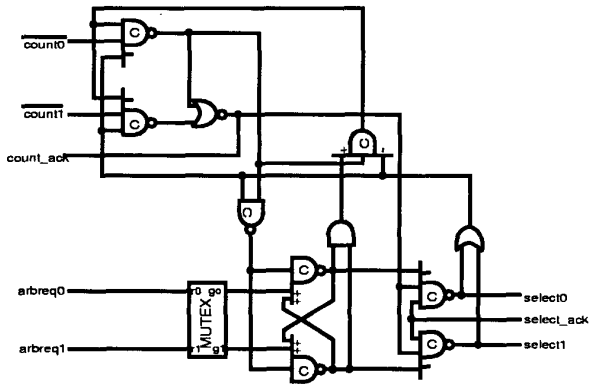

Figure 10: Packet Arbiter

An arbitrated packet multiplexer can thereby be composed of a counter, a packet-arbiter and a channel multiplexer where, for a packet of N symbols, the counter issues N-1 handshakes on select1/select_ack followed by one handshake on select0/select_ack.

6.4. Performance

Averaged over a five symbol packet, EPIC TimeMill simulations of silicon layout show that a 4-input, 4-output packet switch operates at the same 1Gb/s link throughput as the individual 1-of-4 modules. This is a little faster than 2ns/symbol for the payload and a little slower than 2ns/ symbol for the header extraction.

In addition to providing delay-insensitive operation, the 1-of-4 encoded fully connected packet switch allows concurrent connections between inputs and outputs when no conflict for an output arises, and so the total throughput is significantly greater than can be obtained with a shared-bus interconnect. However, the switch does use considerably more transistors.

7. Packet switched Chip Area Network

For small, low performance embedded systems the performance of a single 1-of-4 bundle may be sufficient for the interconnect requirements. An interconnection network using such a system will be used in a prototype asynchronous smartcard system currently under development. Here it will connect a low performance asynchronous processor core to a range of memories and peripherals.

Higher performance systems require greater throughput. This could be provided by chaining a number of these 1-of-4 channels together. For example, four such channels in parallel would require 20 wires, assuming each 1-of-4 group has its own acknowledge signal. Acknowledgement

granularity is an issue that requires further investigation. Such an arrangement provides an aggregate throughput of 4Gb/s.

This figure compares favourably with conventional bus based systems: MARBLE gives $80x10^6$ cycles/second operation with 32-bit datapaths (i.e. 2.5Gb/s) on the same technology; IBM CoreConnect [10] achieves 100MHz with a 64-bit gate-multiplexed datapath (i.e. 6.4Gb/s) on a 0.25 micron process; AMBA-AHB [1] is expected to operate at up to 150MHz with 32-bit datapaths on a 0.25 micron process.

Many on-chip embedded systems have more than four initiators or targets. AMULET3H had 4 initiators with 8 targets all connected through the MARBLE bus. If the packet switched chip area network approach is applied to this application then there are many possible network topologies that could be used to connect the 12 devices. Three are considered here, although only the last of these has been constructed to date.

7.1. Ring network

A ring of 12 two-port switches requires 12 header fields per packet adding a 300% overhead to the 5 symbols per link required to convey a 40bit (control+address/data) payload. However a ring is simple to connect. For this short packet length a ring is not really feasible, but with larger packets (as could be used for, say, a cache line reload) the overhead would be proportionately less.

Two counterflowing rings could be used, one for address traffic and one for data traffic with a split transaction scheme allowing for different ordering of the address and data packet transmission from two different transactions.

The obvious problem with a ring approach is that of avoiding deadlock scenarios when a device, such as a bridge to another network or bus needs to defer a transaction. Conventional buses use a defer mechanism to cause the initiator to back-off and retry the transfer later. A similar scheme can be used here by allowing the target to send the packet around the ring back to the initiator with a completion/rejection tag added. Clearly this represents a loss of available bandwidth since every packet has to pass all the way round the loop, which can interfere with activity from other initiators.

7.2. Switched hub

A fully connected network built around a switched hub approach could use a combination of two 4-input, 4-output switches with two 2-input, 4-output switches connected as illustrated in figure 11.

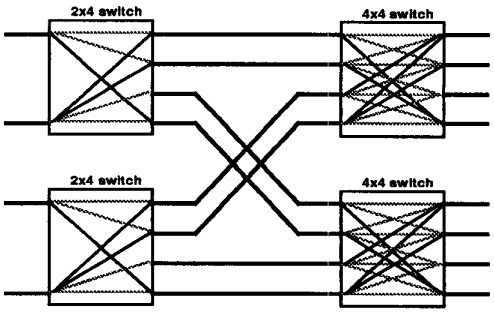

Figure 11: A 4-input, 8-output switch

This arrangement provides full interconnectivity between all initiators and targets and thus avoids the deadlock problem discussed above. As with the application of a ring network, two such fully connected networks could be used, one for the address information and one for the data transfer.

7.3. Multiplexed bus-type arrangement

For many applications, including the AMULET3H chip, a switched hub as discussed above is overkill. A ring network, whilst much simpler, imposes a greater latency due to the switching elements if a packet has to travel around a large part of the ring before arriving at its destination.

A multiplex-demultiplex arrangement, similar to that used by AMBA-AHB and CoreConnect-PLB, but using the 4-group 1-of-4 link approach, is much simpler than the switched hub and provides a lower latency than the ring network.

A replacement for the MARBLE bus used in the AMULET3H chip would require three separate such networks, (as illustrated in figure 12) where each input/output is a 20-wire group of four 1-of-4 channels. The connection from the address-multiplexer to the write-data multiplexer is required to ensure that write traffic arrives at the target in the same order as the addresses are received. Allowing the address and write-data paths to run independently would introduce the need for an (expensive) reordering capability at each target.

In a similar vein, MARBLE operates with a single-outstanding transaction constraint to avoid introducing the need for reordering at the initiator. This approach can be applied here also, although the effect this has on an initiator's ability to saturate the interconnect is more significant because of the increased latency.

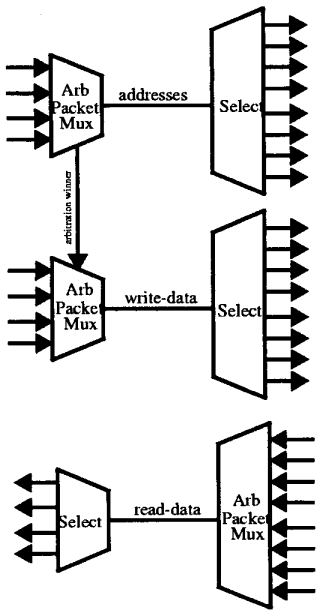

Figure 12: Multiplex-demultiplex arrangement

Simulations show that this network runs at the nominal 4Gb/s throughput (hardly surprising since all pipeline stages have the same number of inversions in them as before). This is a 30% higher throughput than provided by MARBLE in the AMULET3H system, although the latency has increased by about 40% also, primarily because of the large number of pipeline latches.

8. Conclusions

An approach to SoC interconnect has been introduced that offers comparable performance to conventional tristate or multiplexed bus implementations on a 0.35 micron process by using a narrower delay insensitive datapath operated at a higher throughput.

On processes with a smaller feature size it is expected that the performance of this system should not degrade, and may even improve since the switching logic will operate faster, giving a reduced latency. This is in contrast to the expectations for conventional interconnect approaches which use long wide buses. A re-implementation using dynamic gates for the switching units as demonstrated for a very fine grained pipeline by Singh [15] should give significantly smaller circuits and may further improve performance.

Allowing 14 inversions in a pipeline cycle instead of the 10 used here would greatly simplify some of the pipeline stages, and allow the removal of many of the pipeline latches that were added to break the switching control into multiple stages, vastly reducing the latency and allowing the use of pseudo-static C-elements instead of the fully static ones used in much of this work. The cost would be a reduction in throughput to around 700Mb/s for a 1-of-4 group.

The use of a 1-of-4 encoding provides delay insensitive signalling, in theory avoiding the need for extensive verification of the effects of crosstalk on the interconnect wires. Furthermore the use of return-to-zero signalling means that any crosstalk that does occur between interconnect signals should never detract from the performance. However, some timing validation may still be required because of the use of QDI circuits involving isochronic forks for some of the contrl logic in the communications network.

9. Acknowledgments

Aspects of the work have benefited from support from the UK government through the EPSRC, particularly in the form of a Ph.D studentship. The financial support of Theseus Logic is also gratefully acknowledged.

The VLSI design work has leant heavily on CAD tools from Compass Design Automation (now part of Avant!) and EPIC Design Technology, Inc. (now part of Synopsys).

10. References

[1] AMBA, Advanced Microcontroller Bus Architecture Specification, Rev 2.0, Advanced RISC Machines Ltd (ARM), (May 1999)

[2] AMULET3H - 32-bit Integrated Asynchronous Microprocessor Subsystem. AMULET Group, University of Manchester, UK. version 0.8, 1999. URL www.cs.man.ac.uk/amulet/

[3] Bakoglu, H.B., Circuits, Interconnections, and Packaging for VLSI, Adison Wesley Publishing (1990), ISBN: 0-201-06008-6

[4] Bainbridge, W.J., Furber, S.B., "Asynchronous Macrocell Interconnect using MARBLE" Proc. Async'98, San Diego, (April 1998) pp. 122-132

[5] Bainbridge, W.J., Asynchronous System-on-Chip Interconnect, Ph.D. thesis, Department of Computer Science, University of Manchester, UK, (March 2000)

[6] Bainbridge, W.J., Furber, S.B, MARBLE: An asynchronous on-chip macrocell bus., Microprocessors and Microsystems, (July 2000)

[7] Craft, D.J., Improved CMOS Core Interconnect Approach for Advanced SoC Applications, In IP99 Europe, pp233-246, (November 1999)

[8] Dally, W.J., Poulton. J.W., Digital Systems Engineering. Cambridge University Press, (1998), ISBN: 0-521-59292-5

[9] Garside et al. AMULET3i - an Asynchronous System-on-Chip, Proc. Async '00, Israel

[10] IBM Corporation, CoreConnect Bus Architecture, product brief. URL: http://www.chips.ibm.com/news/1999/990923/

pdf/corecon128_pb.pdf

[11] Martin, A.J. et al. The design of an asynchronous MIPS R3000 microprocessor. In Advanced Research in VLSI, pages 164-181, (September 1997)

[12] Matick, R.E., Transmission Lines for Digital and Communication Networks, Mcgraw-Hill, (1969)

[13] Molina, P.A., The Design of a Delay-Insensitive Bus Architecture using Handshake Circuits. Ph.D. thesis, Imperial College of Science, Technology and Medicine, University of London, UK, (1997).

[14] Seitz, C., "System Timing", Chapter 7 of Introduction to VLSI Systems by Mead, C, Conway, L., Addison Wesley. Second Edition, (1980)

[15] Singh, M., Nowick,S.M., High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapaths, Proc. Async '00, Israel

[16] Sutherland, I.E., "Micropipelines", Communications of the ACM, **32** (6), pp 720-738, (June 1989)

[17] Weste, N.H.E., Estraghan, K, Principles of CMOS VLSI Design, A Systems Perspective. Addison Wesley. Second Edition, (1993).

[18] van Berkel, K., Handshake Circuits, An asynchronous architecture for VLSI programming. Cambridge International Series on Parallel Computation. (1993)

[19] Verhoeff,T. Delay-insensitive codes--an overview. Distributed Computing, 3(1):1-8, (1988).

[20] VLSI Technology Inc. 0.35 micron, 3-layer metal CMOS process (VCMN4A3)