An Area-efficient, Pulse-based Interconnect

Simon Hollis, Simon W. Moore
Computer Laboratory, University of Cambridge, Cambridge, CB3 0FD, UK.
Email: {Simon.Hollis, Simon.Moore}@cl.cam.ac.uk

***Abstract*— We present a new style of long-distance, on-chip interconnect based loosely on the asynchronous GasP architecture, with a number of advantages over conventional interconnect. Most significant are a low wire count, a low area requirement, the absence of a global clock and simple composition with existing designs.**

We give some sample throughput and latency figures from simulation on a 0.18μm technology, and show that it is viable for use with modern interconnect requirements, is of low complexity, and has a lower area requirement than parallel interconnect over distances as short as 1mm.

I. Introduction

The International Technology Roadmap for Semiconductors indicates that on-chip interconnect latency and power consumption are increasing with respect to gate speed and power. Current CMOS technologies offer eight or more metal layers, but these incremental increases will never be able to keep up with the exponential increase in connectivity predicted by Rent's Rule [1]. This problem is compounded by issues such as clock distribution, signal integrity and power concerns.

We present a modular interconnection system to manage these problems. Simulations show it has no synchronisation problems and offers a plug-in replacement for parallel interconnect, with data rates close to 1Gbit/s. Communication links use dual-rail signalling which has the side-benefit of balancing power consumption.

Our scheme helps alleviate the problem of low-skew clock distribution [2] by presenting two decoupled, synchronous interfaces. This aids composition, particularly in designs which use multiple clock frequencies. More significantly, it has been designed with a small wire footprint in mind. Reduction in the number of wires required to transmit data at a certain rate not only allows more logic to be added to a device, but also greatly simplifies the routing of troublesome global wires. This can help to speed up the design process.

The requirement for fewer wires gives the ability to route more global signals on top-level metals, increase their width or space them further apart. All enhance performance [3], and the last enables suppression of crosstalk and the use of shielding. These improvements also lead to fewer electromagnetic compatibility problems, reducing development costs. All these benefits are enabled by our design's low wire requirement.

II. Our approach

A. Overview

We now present our area-efficient, pulse-based interconnect. Our basic unit provides an eight-bit, uni-directional interconnect using only two global wires. The number of wires scales linearly with multiples of eight bits. The use of typical synchronous interfaces with FIFO-like signals simplifies integration with standard cells and tools. Between these interfaces, an asynchronous, pulse-driven main interconnect is filled and emptied by trees of multiplexers (MUXs) and demultiplexers (DEMUXs), acting as input and output FIFOs. Operating in a so-called *burst mode* [4], they collect a full word of data before sending it across the interconnect as a single unit. Our system is capable of straddling clock domains and tolerating arbitrary clock skew between its interfaces.

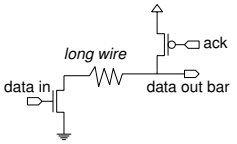


Fig. 1. A Distributed Inverter

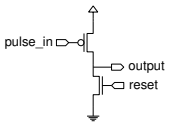


Fig. 2. Our Pulse Latches

B. Our interconnect, based on GasP

The protocol used by our interconnect structure is based on the principle at the heart of the *GasP* system, designed by Sutherland et al. at Sun Microsystems [5]. GasP is an asynchronous FIFO control system, designed to provide very low latency `request` and `acknowledge` signals to micropipeline stages [6]. These signals are contra-flowing, so GasP makes use of a structure called a *distributed inverter* (reproduced for clarity in Fig. 1), allowing a bi-directional flow of events along a single wire. Sutherland et al. call these *state conductors*. Using distributed inverters they transmit events as logic pulses.

GasP is extremely efficient as a control structure, but unsuitable for data transfer since the state conductor is only able to encode events (i.e. request and acknowledge), and not discrete data values. Since we wish our interconnect to be capable of transferring data, we choose to use *dual-rail* encoding [4] of bit values across a pair of wires, each of which follows the distributed inverter scheme.

Dual-rail encoding is an asynchronous signalling protocol where each symbol is encoded on a pair of wires. Of the four states available, two represent symbols `0` and `1`, one is unused, and the final is an idle state. The idle state is used because dual-rail is a return-to-zero (RTZ) protocol. Dual-rail semantics are illustrated in Table I. Dual-rail is often thought of as an area-expensive style of interconnect, but our techniques drastically reduce its area requirement. The other main drawback of conventional dual-rail is that the RTZ phase costs an entire symbol in time, reducing throughput.

By using pulses, which are naturally double-edged, we roll

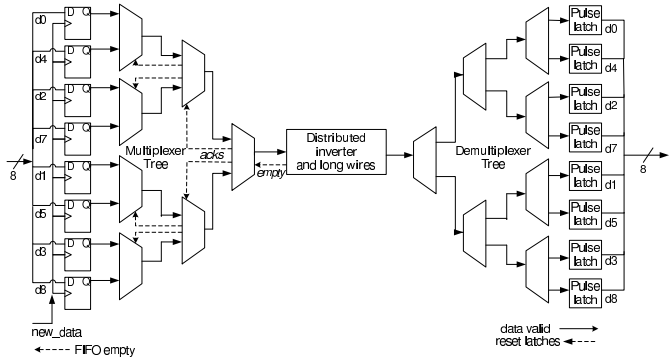


Fig. 3. Our System Block Diagram

TABLE I
DUAL-RAIL SEMANTICS

Wire Values	Meaning
00	Idle/Invalid data
01	0
10	1
11	Undefined/Error

TABLE II
INVERTED DUAL-RAIL SEMANTICS

Wire Values	Meaning
00	Undefined/Error
01	1
10	0
11	Idle/Invalid data

this RTZ phase into the end of data bit symbols, dramatically reducing its impact on throughput. GasP's distributed inverter concept gives us both pulse generation and bi-directional abilities. The latter eliminates all control wires, reducing the wire count and power consumption of our system over that of conventional dual-rail. Since GasP uses logic 1 for the idle state, compared with the logic 0 of dual-rail, we invert dual-rail's normal semantics to make it compatible with our approach. These, inverted, semantics are shown in Table II.

Dual-rail logic runs free of any global clock. Since logic areas a long distance apart are those most likely to suffer from relative clock skew problems, this is extremely useful for long distance interconnect. Our system transcends such matters and provides a reliable channel between them. Implementation is therefore ideal for ASICs, where a designer may not want to have to fully wrest with the idea of exactly equalising global signal paths to all nodes, or where power saving techniques such as voltage islands or multiple clock domains are in use.

III. IMPLEMENTATION

We now discuss the details of our implementation, starting with the main distributed inverter; then the input MUXs; the output DEMUXs; and finally, our pulse latches.

A. Operation of the main interconnect

The heart of our interconnect is extremely simple, comprising a distributed inverter spanning a pair of long wires (see Fig. 1). It operates on an inverted dual-rail protocol, with pulses rather than levels. Input data symbols and output acknowledgments strictly alternate, and further transmissions are blocked until a full cycle has been completed. The wire is long, so both the p- and n-type transistors must be strong.

B. Input MUX tree

The main interconnect is fed with data from a tree of MUX elements responsible for generating and steering pulses of data from a synchronous input environment interface. Due to the varying requirements on MUX operation, dependent on their position in the tree, three flavours of MUX have been designed: Leaf MUXs [Fig. 4(b)], for use next to the input environment; a Root MUX [Fig. 4(c)], for use generating signals for the distributed inverter; and Vanilla MUXs [Fig. 4(a)], for all other situations.

The Root MUX is also responsible for generating master handshaking signals for the MUX tree. These ripple down the tree, clocking those D-flops in MUXs which select a route to the distributed inverter from the input interface. Each new handshake selects a new data bit for transmission, alternating across branches of the tree. Thus, data selection is interleaved in the order illustrated as d0...d7 in Fig. 3. Handshaking is not generated if the output DEMUXs have not emptied sufficiently to accept fresh data. This situation is indicated by the main interconnect not being reset. Hence, our system never loses any data due to a slow output environment, but instead queues data bits in the trees.

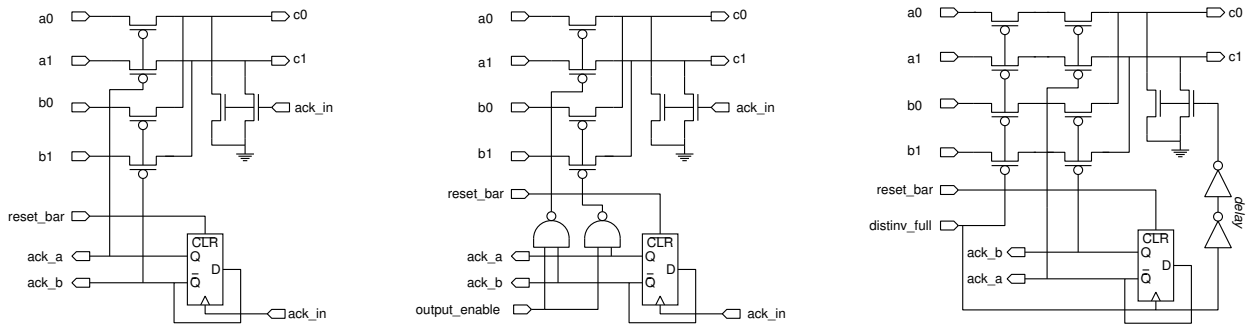
C. Output DEMUX tree

Just as MUXs source data for the main interconnect, demultiplexers (DEMUXs) sink it. Their operation is similar to that of the MUX tree, but with some alterations. First, since no handshaking signals are needed, there is only a single design of DEMUX. Second, we add local state-holding elements, in the form of D-latches, to the outputs of our DEMUX elements. These allow the toleration of arbitrary inter-stage delays, and direct connection to any speed of output environment. A final benefit of latching is that it effectively short-circuits the control path, leading to a cycle time of just one DEMUX element, rather than that of the whole tree. In comparison, our MUXs have no need for state holding since they possess a *push logic* [4] behaviour. This allows us to assume that data is always copied immediately from their outputs by following stages.

Since there is no handshaking, path selection for DEMUXs is controlled by inspection of the actual value on the data path. We know that when we see a transition from the idle state (11) to a valid data symbol (either wire 0) this signifies new data, so we can safely disable the current path, reset the input and configure a new path. The schematic for our DEMUX elements is given as Fig. 5. The pulse generators used in our DEMUXs have been designed to allow inputs straight through to the output, before shaping them as necessary. This presents very little delay, so our cycle time is not adversely affected.

D. Latches and 'Full' signals

Pulse-based systems are transient in nature, and do not lend themselves well to direct connection to static, level-evaluating output environments. Our method of converting from pulses to levels is to introduce a *pulse latch*, which is able to convert a fleetingly-asserted level to a static output. As shown in



(a) The Vanilla Multiplexer Block

(b) The Leaf Node Multiplexers

(c) The Root Multiplexer

Fig. 4. The Three Flavours of Multiplexer

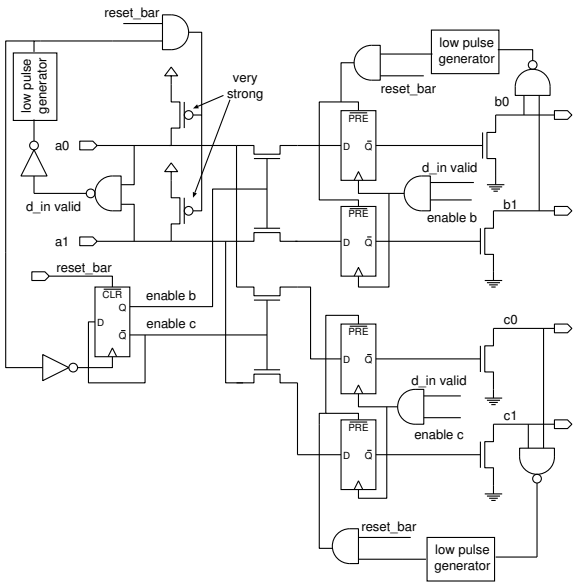


Fig. 5. Our Demultiplexer Block

Fig. 2, it is extremely lightweight: only two transistors are used, and only a single gate capacitance is presented to the input circuit — with positive implications for speed. It is inverting, so its output is suitable for direct connection to an output environment. Valid data is easily detected by checking if either DEMUX line is at logic 0 (i.e. a dual-rail binary value is present). Similarly simple is checking completion of a full output word: see if $\mathtt{d7} \neq \mathtt{00}$.

Since our DEMUXs include latches on their outputs, our latches may simply be used as fast, compact inverters. Conversely, one could omit the leaf DEMUX D-types and use our pulse latches instead, with a more dynamic behaviour.

E. Data interleaving

We believe that the latency of our system may, in a real implementation, be dominated by the turnaround time of the synchronous environment. Therefore we intend to allow our interconnect to perform data interleaving: the successful reception of a partial word will be signalled to the receiving environment, so that it can be read out whilst a previous (partial) word is clearing from the output buffer. Thus, we can ensure that we always have some receive buffer available, preventing the blocking of transmission, and maintaining a high data rate. The input environment and MUXs could operate similarly. Alternatively, an asynchronous FIFO structure could be used to hide synchronous latency.

IV. Methodology

Our results were gained from simulation of the complete system (as shown in Fig. 3) using the *hspice* suite [7] with a UMC $0.18\mu\mathrm{m}$ 1P6M process model. Spice netlists were generated automatically from the schematic tool *Electric*.

To compare directly with Ho et al.'s work (see section VI), we took 3.8mm as the base length of our 'long' interconnect, which was modelled as a pair of parallel transmission lines using physical parameters from our process model. We chose to place these transmission lines on metal layer 6 as it was felt that such a rare, long distance interconnect would most likely be routed on this layer. Our simulations used a worst-case temperature, estimated at $85\,^{\circ}\mathrm{C}$.

V. Results

The following results were gained from our simulations. We believe that, as a well-characterised process, they are reliable and indicative of real silicon performance. Further, worst-case parameters were used, so our results will be conservative.

A. Throughput

1) Distributed Inverter: The main interconnect of our system is the distributed inverter, and system throughput is governed by its cycle time. Our simulations show that each bit takes an identical time to be transmitted, regardless of whether or not it is transmitted on the same wire as the previous one. The cycle time for a single data bit is 1.3ns (so 8 bits take 10.4ns), giving us a main interconnect data rate of 770Mbps, much higher than embedded systems' clock frequencies.

2) Environment: Due to additional control logic, the input buffer's external signals have a longer cycle time than the main interconnect. The good news is how small this overhead is: the time from the input environment signalling new data to receiving a 'input buffer empty' signal is 10.0ns. Note that

TABLE III

AREA REQUIREMENTS (IN μm^2) FOR VARIOUS CONFIGURATIONS

Configuration	Logic Area	Wire Area	Total Area
8-bit, simplex, our scheme, 1mm	5,984	1,320	7,304
8-bit, simplex, std. parallel, 1mm	416	6,925	7,341
32-bit, duplex, our scheme, 1mm	47,872	10,560	58,432
32-bit, duplex, std. parallel, 1mm	3,328	55,440	58,768
32-bit, duplex, our scheme, 3.8mm	47,872	40,128	88,000
32-bit, duplex, std. parallel, 3.8mm	4,560	210,520	215,080

TABLE IV

THE PERFORMANCE OF OUR SYSTEM VS HO ET AL.'S

Parameter	Our Design	Ho et al.
Data width	8 bits	n bits
No. data wires	2	n
No. control wires	0	2
Throughput from 4 wires	1.6 Gbit/s	2.6 Gbit/s
Latency for a 3.8mm wire	1.30ns	0.7ns
Latency for a 5mm wire	1.33ns	1.3ns

this is *less* than the total for 8 cycles of the distributed inverter, because we can signal completion to the output environment simultaneously with transmitting the final acknowledgment (i.e. we need only complete 7.5 cycles). Thus, our system's components are well matched and no one is a critical path.

B. Latency

Throughput is not our only concern. A good interconnect also has low latency. Our simulations show that the latency between a transmitter signalling that new data is valid and the receiver indicating that all data has successfully been received, and is ready to be read out, is 9.7ns. This may seem a little high, but ought to be acceptable for ASIC-based designs, and we believe it still represents good performance over a 3.8mm interconnect with no repeaters.

C. Area

We used the VST $0.18\mu m$ VIP™ Standard Cell Library to provide an estimate of the space required for our complete system. All flip-flops, logic gates etc. used in our design were matched to their equivalents in standard cell and, where this was not possible (e.g. for custom layout), an estimate of $1\mu m^2$ per $1\mu m$ of transistor gate width was used. We believe this to be a reasonable approximation that should give an upper bound on the area requirement. Size estimates for various configurations of our complete system are given in Table III. We compare its area requirements, for both logic and wiring, to that of a standard parallel interconnect; all using minimum wire size and spacing. Full custom layout could give us a more compact design. Our system is competitive for wire lengths as small as 1mm, and an outright winner at the 3.8mm length we have been investigating, even if we omit the area of parallel interconnect's driving circuitry. This is due to the fact that parallel interconnect requires eight wires for every two we use. As wire lengths and widths increase, our approach saves more and more area compared to a traditional parallel approach.

VI. COMPARISON TO RELATED WORK

The most similar design to the one we present here is that presented by Ho et al. [8]. They use two GasP control lines to control a wide data path, enabling two requests and acknowledges to be in flight at once, halving latency. We believe that this is unnecessary and wasteful; the extra control line requires an additional, long wire. We believe we can obtain the same performance with our existing wires.

We illustrate a comparison of their system and ours in Table IV. Our simulations were modified to use identical parameters of metal 5, a $0.18\mu m$ process, and the same wire length of 3.8mm, except where noted. Our design degrades in performance slightly on the thinner metal layer. Our total throughput may not compare favourably to their design but our throughput per wire is much higher. For embedded and low-bandwidth applications, this is clearly desirable. Also, our minimum wire count is two, not four. For lines of 5mm or longer, we break even on latency. Furthermore, our system appears to increase in latency with distance more slowly than theirs after this point.

VII. CONCLUSION

We have presented a high-speed serial interconnect to replace standard parallel on-chip interconnects used in ASICs. In order to source and sink data, we have also demonstrated lightweight multiplexer and demultiplexer tree structures. The system offers good performance, and its asynchronous nature allows it to straddle multiple clock domains. Simulation in a $0.18\mu m$ ASIC technology demonstrates that throughput approaches 1 Gbit/s, and we have seen that it presents a smaller area overhead to a parallel interconnect over distances as short as 1mm. It also compares well with other asynchronous approaches, such as Ho et al.'s, for narrow interconnections. Its modularity and serial nature makes it ideal for inclusion in ASIC designs, where ease of design and simplicity of routing are key.

ACKNOWLEDGMENT

This work was supported by EPSRC grant GR/S42897/01.

REFERENCES

[1] P. Christie and D. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 8, no. 6, pp. 639–648, 2000.

[2] E. Friedman, "Clock distribution networks in synchronous digital integrated circuits," in *Proceedings of the IEEE, Vol 89, No. 5*, May 2001, pp. 665–692.

[3] X.-C. Li, J.-F. Mao, H.-F. Huang, and Y. Lui, "Global interconnect width and spacing optimization for latency, bandwidth and power dissipation," *IEEE Transactions on Electron Devices*, pp. 2272–2279, 2005.

[4] J. Sparsø and S. Furber, Eds., *Principles of Asyncronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.

[5] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, Mar. 2001, pp. 46–53.

[6] I. E. Sutherland, "Micropipelines (the Turing award lecture)." *Comm. A.C.M.*, vol. 32, no. 6, pp. 720–738, June 1989.

[7] Synopsis(R), "hspice." [Online]. Available: http://www.synopsys.com/products/mixedsignal/hspice/hspice.html

[8] R. Ho, J. Gainsley, and R. Drost, "Long wires and asynchronous control," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, Apr. 2004, pp. 240–249.