RasP: An Area-efficient, On-chip Network

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 *RasP*, our asynchronous on-chip-network, which uses high-speed pulse-based signalling techniques. RasP offers numerous advantages over conventional interconnects, such as clock-domain crossing and skew tolerance. Most importantly, it features a very small global-wiring footprint. This compact nature allows a system designer to give priority to link bandwidth or signal-to-noise ratios, rather than being restricted by lane areas.**

We describe our point-to-point link and develop it into a fully-routable system, with a repeater, router, arbiter and multiplexer. Simulations give throughput figures of between 1Gbit/s and 700Mbit/s in a 0.18μm technology, depending on interconnect length. We also show that it compares favourably in performance and area to Bainbridge et al.'s *Chain* interconnect.

I. Introduction

The need for high-throughput, low-latency interconnect has given rise to many advanced designs. Most use many wide and parallel wires, for example as parallel buses. Whilst this approach gives good performance, it is difficult to integrate many copies onto the same die, since a large number of wires need to be routed. Most IC implementations use Manhattan Routing, and so opportunities for track crossover and turns are limited. This is made worse by the fact that all corner turns require an area equal to the square of the bus width.

This crowded environment, found on modern chips, is not conducive to the simultaneous layout of several, wide interconnection structures. Once the issue of how to route so many wires has been solved, traces are often left in close proximity, which causes the problem of crosstalk. The magnitude of crosstalk seen between two wires is inversely proportional to their separation. In Section III-A we will show that, by reducing the number of interconnecting wires, we can dramatically improve signal-integrity. The increase in signal integrity is driven by a reduction in inter-wire capacitance. This also increases throughput, since it reduces the RC product of a wire. Ho quantifies this in his thesis [1], and finds that the bandwidth of a wire is inversely proportional to wire pitch. It is well known that increasing the spacing of wires reduces crosstalk, but we are not always able to do so, due to routing constraints. It is for this reason that area-efficient interconnects are vital to the designer. In the following sections, we will present our area-efficient interconnection system. Its efficiency enables wider wire spacing, and hence a reduction of crosstalk and increase in signal integrity.

First, we will present our point-to-point link, which provides many benefits compared to more conventional designs. For instance, it trivially tolerates clock domain crossing and process variations. This feature come free with our use of clockless, dual-rail logic. A key feature of our system is that it always runs at the maximum possible speed for its environment, and thus many bits can be transferred over our link in a single cycle of its (conservatively) clocked counterparts. At the destination, synchronisation is easily implemented using standard techniques, but we pre-emptively generate control signals, and so performance is not degraded by synchronisation.

All this makes our link an ideal candidate for the basis of a Network-on-Chip (NoC) application, which we call *RasP*, since multiple functional blocks can be connected, regardless of their synchronicity or separation. Later, we will show how multiple instances of our link can be combined to make such a network.

Finally, we will evaluate our system in comparison to Bainbridge et al.'s Chain interconnect implementation. We show that our system is an improvement in throughput and, more importantly, area and energy.

II. Our point-to-point links

Before we consider our full interconnection system, we introduce our basic point-to-point link. Based on the *GasP* system [2] developed at Sun Microsystems, our link requires only a single pair of wires to transmit an 8-bit word of data. Data is first serialised from a synchronous input buffer, before being transmitted bit-by-bit over the two wires, using *dual-rail* encoding [3]. Once at the receiving end, data is de-serialised and placed in a FIFO-like buffer, awaiting a synchronous consumer. Transmission and reception are decoupled in time, the transmitter and receiver being locally clocked. This allows our system to straddle clock domain boundaries with no reduction in efficiency or reliability. This Globally Asynchronous but Locally Synchronous approach is known as *GALS*, and is a common methodology to deal with systems with high clock skew or long distance interconnect.

Our system's GALS nature is produced by a combination of local delay elements (Figs 3 and 5), and a *completion detection* encoding [3] on the global wiring. We will see both of these later, but now we illustrate an overview of our system as Fig. 1. Wider data paths can easily be made by aggregating multiple links together.

The GasP system was originally designed to transmit two contra-flowing control signals, `request` and `acknowledge`, to asynchronous *micropipeline* [4] stages. To do this, it makes use of a structure called a *distributed inverter* (reproduced for clarity as Fig. 2), to allow a bi-directional flow of events along a single wire. Sutherland et al. [2] call these wires *state conductors* and, using distributed inverters as a foundation,

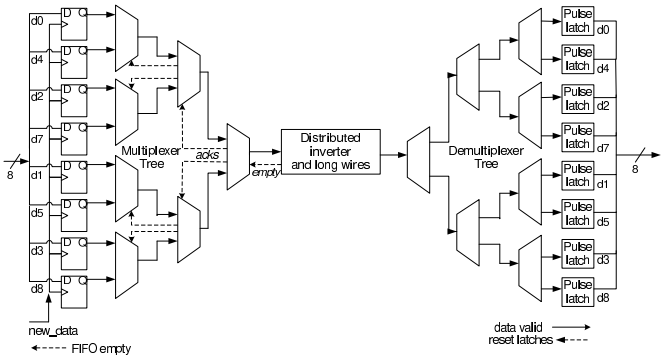


Fig. 1. Our Point-to-Point Link Block Diagram

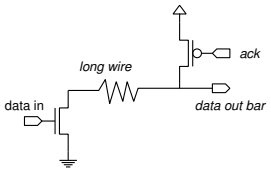


Fig. 2. A Distributed Inverter

they transmit events as logic pulses. Operation is very simple, with self-resetting pulse generators at either end of a global wire being triggered in strict alternation to produce a sequence of data request and acknowledge signals. The two ends only transition in opposing directions, and hence the two signals are distinguishable.

The distributed inverter structure is also know as a *single-track* structure, and previous work has demonstrated logical operations using a 1-of-n encoding technique with single tracks [5]. The issue of serialisation over long wires has, however, not been properly addressed before.

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.

We use dual-rail encoding to give data semantics to our links. It is an asynchronous (lacking a global clock) signalling protocol [3] and is extremely robust, tolerating arbitrary delay at any stage. 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.

We use pulses, which are naturally double-edged, to roll 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

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

wire count and power consumption of our system over that of conventional dual-rail logic. In this way, our approach greatly reduces area requirements. The edge-based signalling means it is sufficient for an edge to be detected, without requiring a full logical level. Therefore, our interconnection wires do not need fully charging to signal information, and this results in a performance increase compared to traditional, level-based signalling. This relaxed approach to voltage levels means that our drivers can be relatively weak given the length of wire. Thus, our pulses are RC shaped, rather than with the sharp edges of traditional circuits. This smoothing reduces peak power consumption, as well a suppressing electromagnetic emission, which helps design for EMC compatibility.

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. Detection of valid data is easy: simply NAND together the inverted dual-rail output. We do so for each receiver stage to produce a `valid_data` signal.

Dual-rail logic is not new, but few attempts have been made to operate it using pulses rather than logical levels. Nanya and Kameda investigated the area some time ago [6], but they used exotic quantum gates, rather than the standard CMOS we use. Ho et al. use a dual GasP control-channel scheme [7], but the data is still transmitted using a *bundled-data* technique [3], rather than being serialised and sent over the same wires as the control signals.

Since dual-rail logic runs free of any global clock and logic areas far apart are those most likely to suffer from relative clock skew problems, our approach is especially useful for long distance interconnect. Transcending skew, it provides a reliable channel. 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.

When combined into a larger system, the lack of need for synchronicity makes composition of our systems elements straightforward. This, combined with its low global wire area requirement render it attractive for NoC implementations, where communication channels are generally pre-reserved on a floorplan — the reduction in area can allow more channels or a higher bandwidth.

III. LONG WIRES

Long wires present a problem for all forms of interconnect. Simplest of all, the finite speed of light causes two signals

TABLE III
COUPLING CAPACITANCES FOR CONFIGURATIONS OF A 3800μm TRACK

Configuration	C_{mutual}	C_{gnd}	C_{total}
Minimally-spaced, no gnd	0.437pF	—	0.437pF
Doubly-spaced, no gnd	0.224pF	—	0.224pF
Triply-spaced, no gnd	0.162pF	—	0.162pF
Minimally-spaced, gnd	0.538pF	0.203pF	0.741pF
Doubly-spaced, gnd	0.350pF	0.237pF	0.578pF
Triply-spaced, gnd	0.078pF	0.252pF	0.330pF
Minimally-spaced, gnd, M4&6 wires	0.346pF	0.293pF	0.639pF
Doubly-spaced, gnd, M4&6 wires	0.174pF	0.327pF	0.501pF
Minimally-spaced, guard wires, no gnd	0.059pF	0.389pF	0.448pF
Minimally-spaced, guard wires, gnd	0.011pF	0.527pF	0.538pF

arriving at two different on-chip locations to experience varying degrees of skew. This can result in data corruption if these signals happen to arrive at an unexpected time, from the point of view of the receiving circuitry (such as near a clock transition). Equally relevant, the longer the wire, the more global wiring metal it takes up, so we wish to minimise the width and spacing of a channel's trace, to allow others to be routed. In our previous work [8], we show how our point-to-point links are much more area-efficient than standard parallel interconnects, and they form the basis of our approach here.

Signal integrity is also affected adversely by increases in wire length. We notice attenuation due to resistive losses, and increases in capacitance between both the wire and ground and the wire and its neighbours (cross-coupling). The first form of capacitance increases the delay of the signal; as does the second which, more seriously, can also introduce a large degree of noise, reducing signal integrity. Similarly, whilst inductance can sharpen signals, it also causes coupling, and impressing additional noise.

Using data from a UMC 0.18μm 1P6M process, a model was created of likely interconnect structures. The capacitance extraction program *Quickcap* [9] was then used to produce capacitance values, both for line-to-substrate and line-to-line coupling. Varying arrangements of line and location, on several metal layers with varying distributions of neighbour wires were simulated. This data was then placed in an *hspice* [10] model of our full interconnection system, and we display the results as Table III. In the table, C_{mutual} is the inter-signal-wire capacitance, C_{gnd} is the wire-to-ground capacitance, and C_{total} is the sum of the two. When we guard, the shielding wires are connected to ground, explaining the non-zero ground coupling, even when there is no ground plane.

We show capacitances for minimally, and doubly-spaced metal 5 lines, with and without ground planes, and with and without minimally-spaced wires in the neighbouring metals 4 & 6. Given these results, we chose to base our system's model on a wire configuration where signal wires were minimally-spaced on metal 5, surrounded by fully-populated metal 4 and metal 6 layers. This gives the worst-case system performance.

All the wires in our system act as transmission lines in the RLC operating region [11], where line propagation delay increases linearly with trace length. Our system is able to adapt to such increases by varying the width of transmitted pulses. Since our forward transmissions only cease once an acknowledgement has been received (see §II), we merely need to vary the width of the acknowledgement pulse: longer for a longer line, and shorter for a shorter one. The optimal number is proportional to signal flight time along the line: δ. This is easily found from the line's length l and its per-unit-length characteristics of resistance, $\hat{R}$, and capacitance, $\hat{C}$, using the simple formula $\delta = l\sqrt{\hat{L}\hat{C}}$. Extraction of these for our line, followed by simulation gave the propagation delay over 3800μm to be 559ps.[†]

Using our model, our system has been shown to operate correctly over unrepeated wire lengths ranging from 750μm to 5000μm. This is exactly the range of distances for which our system is intended since, for distances shorter than 750μm, local wiring is more likely to be used than the higher metal layers we favour. Similarly, few long-distance interconnections run unrepeated for as long as 5mm, and our system is no different.

A. Signal-integrity

A major motivation for producing area-efficient designs is concern over signal integrity. Closely spaced wires couple capacitively (so-called *cross-coupling*), and the closer they are, the stronger the coupling. Cross-coupling produces two effects, both of them unwanted: the retarding of signals, and the creation of noise. As an example of the latter, in our system, a minimally-spaced 3.8mm wire is able to impress crosstalk noise of 44% of `vdd` on a neighbour. This is a dangerous value, and one must take special measures to ensure that it does not cause a problem. This crosstalk is the main reason that our system fails to transmit data correctly at unrepeated wire lengths of over 5mm.

From Table III, we see that simply doubly-spacing the two signal wires in our system gave a decrease in mutual capacitance (and hence crosstalk) from 44% of `vdd` to 26% of `vdd` in the basic case and, in most cases around a 50% reduction. Whilst shielding wires were shown to reduce the crosstalk dramatically, the total capacitance was still high, slowing down signal propagation and degrading potential throughput benefits from reduced crosstalk.

Since capacitive coupling decreases with the distance between two wires, area-efficient designs can offer wider spacing, resulting in reduced cross-coupling through increased separation. Alternatively, this space can be used to insert shielding wires, but we have already seen that we are better off without, and merely increasing spacing, since shielding increases capacitance as well as reducing noise.

B. Repeaters

Signal integrity is not the only thing impeded by long wires. Performance is also decreased due to the impact of an increased delay. The delay of a line is responsible not only for end-to-end latency of a system, but also for throughput in a system where acknowledgements are required (such as ours

[†]This figure may seem large for 3.8mm, but recall that this particular line was simulated with multiple, minimally-spaced neighbours and, thus, has a very high capacitance, and so signals are strongly retarded.

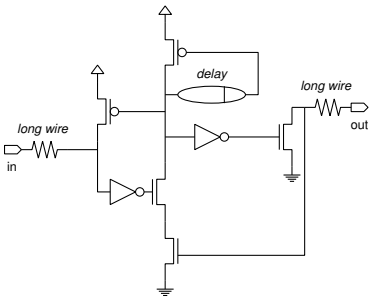


Fig. 3. A Standard GasP Repeater

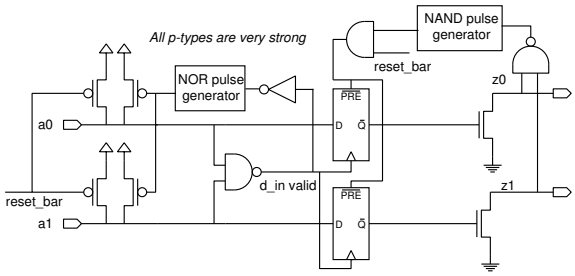
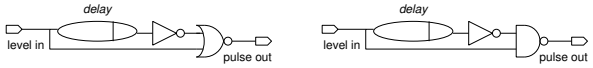


Fig. 4. Our Stateful RasP Stage Repeater



(a) The NOR Pulse Generator (b) The NAND Pulse Generator

Fig. 5. The Two Types of Pulse Generator

— the performance of a computer network using a sliding window protocol with window size of one is a good example of this [12]). Hence, the throughput of a section of our system is, indeed, limited by the delay (and hence length) of its wires. There are two approaches to ameliorating this problem: reduce the delay; or segment the wires, producing multiple sections with lower unit delays. Repeaters do both.

The delay of a line can be minimised by the technique of optimal repeater-insertion [1], [13]. This is a well-known approach, and reduces the delay of a signal travelling down a long wire by placing repeaters of optimal sizes at optimal locations. Repeaters are interesting in our GasP-based approach since we require bi-directional communication, and repeaters traditionally offer only uni-directional operation. To solve this problem, and also that of the critical impact of acknowledgements on the performance of our system, we choose to pipeline, rather than repeat. In this approach, we split a long wire at intervals with repeaters, and also add elements that provide a full signalling-protocol cycle (they consume the data and re-transmit it, but also provide an backwards-propagating acknowledgement signal). Our technique is similar to the one used by Ho et al. to pipeline an asynchronous channel [7]. Our approach therefore increases system performance, as well as boosting signal integrity.

We now describe two varieties of repeater, suitable for use with our interconnect: the first is stateless and is based on a single GasP pipeline control stage, and the second is a stateful design using D-type latches. We illustrate the former as Fig. 3, and the latter as Fig. 4. We use the first for logic-buffering applications, and the second for wire repeating. Note that we choose to add a set of dedicated reset p-type pull-ups on the input of the second, rather than logically ANDing the reset signal into the acknowledge p-types. This choice is made to eliminate an additional gate delay on the reverse, acknowledgement, path; we believe the parasitic delay from the additional p-types to be less than the corresponding logical overhead. The pulse generators both generate low pulses, and are based on the simple edge detector circuit in Johnson and Graham's book [14, p.181]. Depending on the polarity of the input, we use either a NOR-based design, (Fig. 5(a)), or a NAND-based one, (Fig. 5(b)). In our stateful repeater. the latency of the D-latch is less critical to throughput than the fact its inclusion allows local acknowledgements to be generated.

We have already seen that the propagation delay of our unrepeated wire is 559ps. In comparison, if buffered with optimally placed repeaters (in our case two of them), the total propagation delay becomes 186ps [15, p.221]. The forward and backward logic delays of our stateful repeaters are equal, at approximately 400ps. These latencies correspond to just over four fan-out-4 delays in our technology. This is a large figure for a repeater, but recall that our system transfers data in byte-long bursts, and so latency is less important than the fact that throughput is boosted by the repeaters. The total wire time for a bit then becomes around 600ps in the forward case, and 1.2ns in the full-cycle case. Hence, the repeater logic overhead means that repeated and unrepeated lines have similar total delays at the 3.8mm length. However, repeaters do reduce the overall delay for lengths greater than this.

With data from our *hspice* simulations, we show in Fig. 6 how the transmission time for a single byte decreases as we add additional repeaters, up until the optimal number (which, for $3800\mu\mathrm{m}$ is two). After this point, latency increases again because the repeater delay becomes more significant than the wire delay. We clearly see that the minimum transmission time is when we use two repeaters. This also corresponds to the smallest distance between our two curves, their separation indicating the propagation delay of the wires. The 'Logic overhead' plot shows the delay of all the logic in our system, and it is clear that the delay from inserting additional repeaters increases linearly, as would be expected for a line which is almost optimally repeated. Similarly, the time taken to transmit the whole data word increases linearly with wire length.

In Fig. 7 we show how the cycle time of a single bit passing to, and being acknowledged by, a repeater stage varies with the length of the interconnect to be travelled. The linearity clearly illustrates our line's RLC characteristic.

IV. A NETWORK OF POINT-TO-POINT LINKS

A natural extension of a point-to-point link is to make data routable to a number of destinations. This NoC approach is becoming increasingly popular with system designers since it obviates the need for a large number of custom-routed data paths, and has a regular structure and a predictable nature.

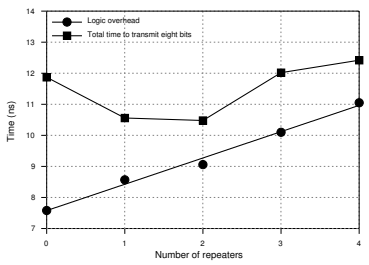


Fig. 6. Transmission Characteristics for a Byte over 3800μm, for Varying Numbers of Repeaters

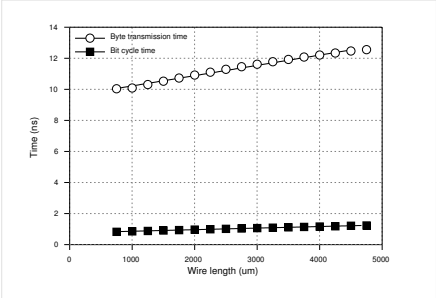


Fig. 7. Bit Delays for Our Repeater with Varying Interconnect Lengths

Our point-to-point links have demonstrated that we can transfer data at high speed, and over large distances, on-chip, without requiring a large area. We will now illustrate how several can be combined into a chip-area network, without compromising the low-area and high throughput benefits.

We model our NoC design on that of Bainbridge's *Chain* interconnect [16]. We chose this basis since we too offer an asynchronous data path, and use similar routing elements. Like them, we present a router, arbiter and multiplexer for our form of interconnect; we have already presented the equivalent of Bainbridge's 'pipe-latch' as our stateful repeater in Section III-B. We name our system *RasP*: a Routable asP-based interconnect. It has cutting-edge performance and, like Chain, is modular, with no composition problems due to its arbitrary tolerance of clock skew due to local handshaking.

A. Our RasP router

Key to our RasP system is a router element, shown as Fig. 8. It takes destination address information and a single data input channel, and chooses one of four output channels to route that data to. For simplicity, we illustrate operation with two address wires indicating one of four routes by their states. In a larger system, the first transmitted data word could be used to indicate one of 256 possible routes; or, we could convert the two address wires to a fully-fledged point-to-point link, serially transferring a long address. We show the schematic for the 'north' output channel. This logic is repeated for each of the four routable directions, with a central route controller shown with dashed lines. Our simplified address wires are steered with AND gates, from the same enable signals, since they do not use our pulse-based protocol. In our small example,

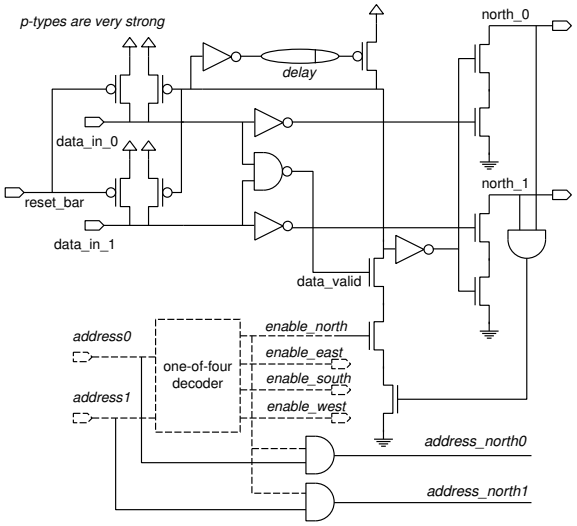


Fig. 8. Our RasP Router

where we have only four possible destinations, the address does not actually need to be routed to the following stage, but we have included the circuitry to indicate how this may be accomplished for larger systems.

B. Our RasP arbiter / multiplexer

Since multiplexing in RasP simply involves connecting the two merging wires together, we can trivially merge the outputs of our arbiter. This eliminates the need for a separate multiplexer, as is used in the Chain architecture.

At the heart of our arbiter (shown as Fig. 9) is a two-way mutual-exclusion (MUTEX) element, which takes in two request signals, `req0` and `req1`, and grants at most one of `gnt0` and `gnt1`. If both requests are asserted simultaneously, one channel will be chosen at random to be granted and, when the request is de-asserted, the other grant will be enabled. This property ensures a degree of fairness: once one input channel has had a packet transmitted, the other one will get to transmit, if it is waiting. Our arbiter's data lines are not latched, and operate by enabling transmission gates on one of the output directions. To ensure signal integrity following these pass transistors, we recommend the immediate use of a GasP-style repeater (Fig. 3), to boost line levels.

Like Chain, we arbitrate on a per-word basis, in order to reduce the arbitration overhead. After arbitration on the first bit of a word, the output path is enabled until the end of the word. A counter tracks the number of bits transmitted since arbitration began, and only releases the MUTEX request after all eight bits of a data word have been transmitted. This approach increases the amount of logic, but removes the need for another global wire, signalling 'end-of-packet'. For wider data paths, where we aggregate multiple point-to-point links, the counter should not be clocked until all data wires are showing `valid_data` signals.

An interesting feature of our arbiter is the operation of the transmission gates: they are driven asymmetrically. Our inverted dual-rail protocol uses low pulses in the forward

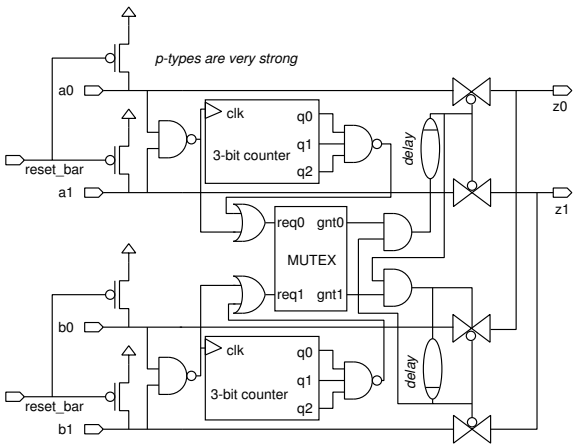


Fig. 9. Our RasP Arbiter-cum-multiplexer

direction, and high ones in the backward. This places the n-type pass transistors on the critical forward path, and the p-type on the equally-critical backward one. Hence, we activate the n-type pass transistors immediately after a grant signal is produced, to allow the fastest possible data propagation. Activation of the p-types is more interesting: since they are not critical on the forward path, we can afford to delay their activation after receiving data. However, they must remain on for a period after the backward-going acknowledge signal is detected, to ensure it has time to charge the input line fully. This involves a period of time where the MUTEX's request line is reset since all wires have gone high locally, and so the `data_valid` signal will be de-asserted. Ordinarily, this would remove the grant, and hence de-activate the transmission line, but we insert a delay to keep it activated for a time sufficient to charge the input line. We were able to do this by delaying its activation for the same time after valid, forward-going data is detected on an input, and performance is unaffected.

C. An introduction to the Chain interconnect system

The Chain interconnection system [16] was developed by Bainbridge et al. as a flexible way of connecting together multiple IP blocks on a single die. By means of comparison, single-track schemes merge elements of Chain and RasP [5]. The Chain approach is to use an asynchronous logic style [3], which allows them to connect together multiple blocks, regardless of whether they share a common clock frequency. It is precisely this advantage, given by asynchronous logic, which makes is so attractive for NoC applications. However, nothing ever comes for free, and asynchronous logic is no different: the price paid for clock independence is area and, potentially, power. Asynchronous logic typically takes around twice the area of more conventional types. Amazingly, this does not always entail a power disadvantage, since asynchronous designs consume switching power only when there is data to be processed.

Chain is no different from any other asynchronous design: it has a high area overhead, mainly because it uses *one-of-four encoding* [3], which requires four data wires to transmit two bits of data. Further, they add two control wires, bringing the total to six wires (for only two bits of data). Chain, unfortunately, suffers performance penalties: asynchronous designs are based on *C-elements*, state-holding multiple-input gates. C-elements are generally slow, and so Chain's latency and cycle times are dominated by these sluggish components. According to our simulations, the cycle time of a Chain stage is a minimum of 1.56ns, compared with only 0.83ns for our design, where we avoid C-elements.

TABLE IV
RASP COMPARED TO CHAIN, UNREPEATED

Characteristic	RasP	Chain
No. of wires	2	6
Bits/cycle	1	2
Bit cycle time, zero-length wire	0.83ns	1.56ns
Bit cycle time, 3800μm wire	1.25ns	3.89ns
Time for 8 bits, zero-length wire	6.64ns	6.24ns
Time for 8 bits, 3800μm wire	11.3ns	15.6ns
Throughput with zero-length wire	1.20Gbit/s	1.28Gbit/s
Throughput with 3800μm wire	707Mbit/s	513Mbit/s
Wire area (metal 5) for 3800μm	1,596μm^2	5,852μm^2
Logic area	5,984μm^2	552μm^2

V. EVALUATION OF PERFORMANCE

For fairness of comparison, we present not only our own results, but those from an implementation of Chain in our process. We chose to re-implement Chain to bring the technology values up-to-date, to give a fair comparison.

A. Our Point-to-point Link

Table IV shows the performance of an unrepeated point-to-point link, which forms the basis of the RasP architecture, against that of a basic Chain link. We show both 3800μm and zero-length links. Since RasP only functions correctly for lengths of 750μm and above, we use this figure rather than zero (for the RasP results only). We see that, for a zero-length path, the Chain architecture performs better than RasP. This is mainly since GasP has a minimum pulse-width limit, but Chain uses a simple `ack` cycle. Once we move on to a long wire, however, RasP's tolerance to delay means that its cycle time increase is only 50%, compared to an increase of 150% for Chain. This naturally impacts upon Chain's performance more heavily, and this is shown with its decrease in throughput over this wire to 513Mbit/s, compared to RasP's 707Mbit/s.

The table also gives area estimates for both RasP and Chain. At the 3800μm point, the total logic and wire amounts are similar, but note that, while Chain is mainly using global metal wiring, which scales with line length, RasP's size is mainly a fixed serialisation / de-serialisation overhead. Thus, at longer distances, RasP's area will be substantially lower than that of a Chain implementation. Since global on-chip interconnects are typically of at least the distance we consider, our system presents a smaller total footprint than Chain for typical applications. Further, it always saves significant quantities of precious global metal. Our previous work [8] has also shown that, for real-world bus widths of 32 bits, and duplex links, RasP occupies just under a third of the area of a standard parallel interconnect; and, even for shorter lengths, RasP is

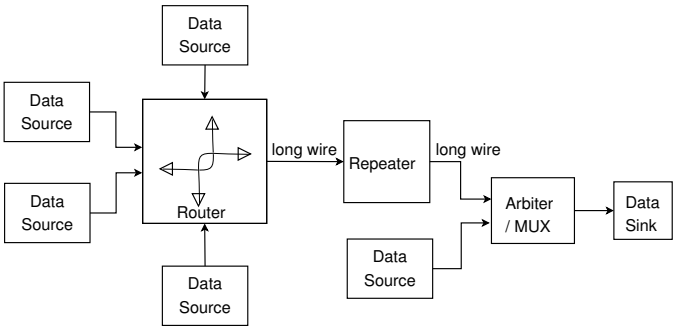


Fig. 10. Our Complex System Test Setup

TABLE V
COMPLEX SYSTEM PERFORMANCE FOR RASP AND CHAIN

Characteristic	RasP	Chain
First bit latency, zero-length wire	3.37ns	1.51ns
First bit latency, 3800μm wire	3.73ns	4.87ns
Byte latency, zero-length wire	7.57ns	6.04ns
Byte latency, 3800μm wire	12.1ns	19.48ns
Throughput with zero-length wire	1.03Gbit/s	1.32Gbit/s
Throughput with 3800μm wire	661Mbit/s	411Mbit/s
Repeater energy for a 3800μm wire	0.67pJ/bit/mm	0.86pJ/bit/mm

always more competitive than a parallel interconnect. All these results show the area-efficiency of our system, enabling the routing of multiple RasP links in the space afforded to just one parallel or Chain interconnection.

B. Our Full RasP System

We now consider a more complex system, using all the components of our RasP system. We take a data source, feeding a router, and being routed out onto a long (3800μm) wire. This wire is split in half by a repeater, before going into an arbiter and multiplexer, and finally being consumed by a data sink. We illustrate this configuration as Fig. 10. Performance for such a system via both implementations is given in Table V. We see a similar performance pattern to that of the point-to-point links: Chain just beats RasP for a zero-length wire implementation, but RasP's performance is better at longer wire lengths. Both systems experience a performance boost over the raw point-to-point link since the long wire is pipelined. Much of Chain's bad performance is as a result of the over-prevalence of C-elements [3] in their design. C-elements are state-holding elements and, even after heavy optimisation, are slow gates. More critically, Chain requires full-swing transitions on wires to signal. RasP is able to function with voltage swings as small as $\texttt{vdd}/2$. This enables a large improvement in performance for RasP over long distances, where transition time is dominated by the RC delay of the wires.

Finally, if we consider a single repeater link, in energy-per-bit performance, RasP is able to transfer bits for only 78% the energy required by Chain.

VI. CONCLUSIONS

We have presented our routable, GasP-based system, which we call *RasP*. We have shown that it constitutes an area-efficient, point-to-point link, featuring high throughput and a low global-metal footprint. We have shown how this link can be extended into a fully-routed NoC system, and have presented a repeater, router and arbiter / multiplexer for this purpose. The full RasP system maintains the area-efficiency of its constituent links, and is a great improvement over traditional parallel interconnects.

Simulations have illustrated the usefulness of repeaters over long distances, and have demonstrated that our system offers a higher throughput-per-unit-area than the similar Chain system. The reduction in metal area provided by our solution greatly helps routing, by offering more freedom to designers; and also enables reductions in crosstalk or increases in throughput. Further advantages of our system over traditional interconnects include arbitrary clock-domain crossing and ease of composition due to clock-skew tolerance. It is therefore ideal for NoC applications, where multiple IP blocks may need to be connected, or for ASICs where a predictable communication structure is needed.

Measured throughput of our system operates over a sliding scale with interconnect length, between just over 1Gbit/s for short wires to around 700Mbit/s over 3800μm, unrepeated, on a 0.18μm technology.

REFERENCES

[1] R. Ho, "On-chip wires: Scaling and efficiency," Ph.D. dissertation, Department of Electrical Engineering, Stanford Universiy, Aug 2003.
[2] 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.
[3] J. Sparsø and S. Furber, Eds., *Principles of Asyncronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.
[4] I. E. Sutherland, "Micropipelines (the Turing award lecture)." *Comm. A.C.M.*, vol. 32, no. 6, pp. 720–738, June 1989.
[5] M. Ferretti and P. Beerel, "Single-track asynchronous pipeline templates using 1-of-n encoding," in *Proc. Design, Automation and Test in Europe (DATE)*, 2002.
[6] T. Nanya and Y. Kameda, "Pulse-driven delay-insensitive circuits using single-flux-quantum devices," in *Proc. International Conference of Computer Design*, 1996.
[7] 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.
[8] S. Hollis and S. W. Moore, "An area-efficient, pulse-based interconnect," in *Proc. International Symposium on Circuits and Systems (ISCAS)*, May 2006.
[9] Magma Design Automation Ltd., "Quickcap." [Online]. Available: http://www.magma-da.com
[10] Synopsis(R), "hspice." [Online]. Available: http://www.synopsys.com/products/mixedsignal/hspice/hspice.html
[11] H. Johnson and M. Graham, *High-speed Digital Design: A Handbook of Black Magic*. Prentice Hall, 1993.
[12] A. S. Tanenbaum, *Computer Networks*. Prentice Hall, 1981.
[13] Y. I. Ismail and E. G. Friedman, "Repeater insertion in RLC lines for minimum propagation delay," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 1999.
[14] R. J. Tocci, *Digital Systems: Principles & Applications, 6th Edition*. Prentice Hall, 1995.
[15] N. H. Weste and D. Harris, *CMOS VLSI Design (Third Edition)*. Wesley, 2005.
[16] J. Bainbridge and S. Furber, "CHAIN: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, pp. 16–23, 2002.