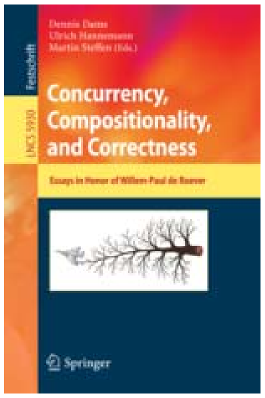
Springer www.springer.com

Computer Science - Theoretical Computer Science - Foundations of Computing | Concurrency, Compositionality, and Correctness


Dennis Dams
Ulrich Hannemann
Martin Steffen (Eds.)
Festschrift
LNCS 5930
Concurrency, Compositionality, and Correctness
Essays in Honor of Willem-Paul de Roever
Springer

Concurrency, Compositionality, and Correctness

Essays in Honor of Willem-Paul de Roever
Series: Lecture Notes in Computer Science
Subseries: Theoretical Computer Science and General Issues, Vol. 5930

Dams, Dennis; Hannemann, Ulrich; Steffen, Martin (Eds.)

1st Edition., 2010, 377 p., Softcover
ISBN: 978-3-642-11511-0

Online orders shipping within 2-3 days.

54,00 €

About this book

- 19 detailed papers by eminent scientists in the field of Concurrency, Compositionality and Correctness. Contains a detailed bibliography of the honoree. Closes with a gallery of photographs.

This Festschrift volume, published in honor of Willem-Paul de Roever, contains 19 detailed papers written by the friends and colleagues of the honoree, all eminent scientists in their own right. These are preceded by a detailed bibliography and rounded off, at the end of the book, with a gallery of photographs.

The theme under which the papers have been collected is *Concurrency, Compositionality, and Correctness*, reflecting the focus of Willem-Paul de Roever's research career. Topics addressed include model checking, computer science and state machines, ontology and mereology of domains, game theory, compiler correctness, fair scheduling and encryption algorithms.

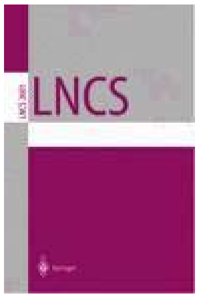
Written for » Research

Keywords » GasP circuits - alternative definitions - automated proofs - bisimulation - co-simulation - complexity - distributed systems - embedded systems - formal semantics - intrusion detection - model checking - modeling - network-on-chip - parallelization - proof systems - refusal equivalence - relative timing verification - scheduling - special applications - specification language - state machines - synchronous message passing - testing - trace equivalence - verification

Related subjects » Foundations of Computing - Software Engineering

SpringerLink

Book Chapter



Timing Verification of GasP Asynchronous Circuits: Predicted Delay Variations Observed by Experiment

Book Series	Lecture Notes in Computer Science
Publisher	Springer Berlin / Heidelberg
ISSN	0302-9743 (Print) 1611-3349 (Online)
Volume	Volume 5930/2010
Book	Concurrency, Compositionality, and Correctness
DOI	10.1007/978-3-642-11512-7
Copyright	2010
ISBN	978-3-642-11511-0
DOI	10.1007/978-3-642-11512-7_17
Pages	260-276
Subject Collection	Computer Science
SpringerLink Date	Friday, February 12, 2010

PDF (488.3 KB) Free Preview

Prasad Joshi[1], Peter A. Beerel[1], Marly Roncken[2] and Ivan Sutherland[3]

(1) Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA

(2) Strategic CAD Labs, Intel Corporation, 5200 NE Elam Young Parkway, Hillsboro, OR 97124, USA

(3) VLSI Research Group, Sun Microsystems, 16 Network Circle, Menlo Park, CA 94025, USA

Abstract

This paper reports spreadsheet calculations intended to verify the timing of 6-4 GasP asynchronous Network on Chip (NoC) control circuits. The Logical Effort model used in the spreadsheet estimates the delays of each logic gate in the GasP control. The calculations show how these delays vary in response to differing environmental conditions. The important environmental variable is the physical distance from one GasP module to adjacent modules because longer wires present greater capacitance that retards the operation of their drivers. Remarkably, the calculations predict correct operation over a large range of distances provided the difference in the distances to predecessor and successor modules is limited, and predict failure if the distances differ by too much. Experimental support for this view comes from the measured behavior of a test chip called "Infinity" built by Sun Microsystems in 90 nanometer CMOS circuits fabricated at TSMC.

Keywords Network-on-chip - relative timing verification - GasP circuits

Prasad Joshi
Email: prasadjo@usc.edu

Timing Verification of GasP Asynchronous Circuits: Predicted Delay Variations Observed by Experiment

Prasad Joshi[1], Peter A. Beerel[1], Marly Roncken[2], and Ivan Sutherland[3]

[1] Ming Hsieh Department of Electrical Engineering,
University of Southern California, Los Angeles, CA 90089, USA
{prasadjo,pabeerel}@usc.edu

[2] Strategic CAD Labs, Intel Corporation, 5200 NE Elam Young Parkway,
Hillsboro, OR 97124, USA
marly.e.roncken@intel.com

[3] VLSI Research Group, Sun Microsystems, 16 Network Circle,
Menlo Park, CA 94025, USA
ivan.sutherland@sun.com

Abstract: This paper uses the method of Logical Effort to reason about the timing of 6-4 GasP asynchronous Network on Chip (NoC) control circuits. Logical Effort provides an analytic relationship between the drive strength of a circuit, the load it drives, and its delay. The important environmental variable in this relationship is the physical distance from one GasP control module to adjacent modules because longer wire connections present greater capacitance that retards the operation of their drivers. Remarkably, this analysis predicts correct operation over a large range of distances provided the *difference* in the distances to predecessor and successor modules is limited, and predicts failure if the distances differ by too much. Experimental support for this analytical prediction comes from the measured behavior of a test chip called "Infinity" built by Sun Microsystems in 90 nanometer CMOS circuits fabricated at TSMC.

Keywords: Network-on-chip, relative timing verification, GasP circuits.

1. Introduction – Network on a Chip (NoC)

Advances in integrated circuit technology have produced chips of amazing speed containing hundreds of millions of transistors. These same advances have forced designers to put more emphasis on communication systems within such chips for four reasons. First, it costs energy to send signals on wires, and an increasing fraction of the energy budget is spent on communication. Second, the wires cost chip area. Modern chips have about a dozen layers of wiring to provide the wiring space required. Third, finding routes for the maze of wires, though highly automated, is becoming an increasingly daunting task. And fourth, the sheer complexity of modern designs requires their partition into more manageable sub pieces.

For these reasons, there has been much recent discussion of system on a chip (SoC) designs constructed from separate parts linked together by a network on the chip (NoC). Such a design approach can partition the system into separate parts that can be designed concurrently or even re-used from previous products. Concentrating the communication needs of such a design into a network with pre-specified properties provides simple interfaces to which each separate part can attach. Moreover, it greatly simplifies the final assembly of the separate designs into the final SoC.

By now there is a diverse family of NoC designs. Many of these communicate the presence or absence of data as well as the data itself, using one of a variety of handshake protocols for the purpose, see e.g. [1]. Such handshake protocols require signals that travel both forward and backward along the network; forward signals to indicate the presence of new data that could progress through the network and backward signals to indicate the presence of space into which such data may fit.

Families of NoC circuits generally include special modules that can branch and merge arms of the network. Such modules may, for example, send an incoming message to several outputs in a "broadcast" mode, or may send the message to only one of several outputs in an "addressable" mode. Other modules may join messages from two or more inputs, again either by concatenation of simultaneously-arriving messages, or on a "first-come-first-served" basis. Such modules form the basis for a wide variety of network topologies. The experimental chip described in this paper uses a two-way addressable Branch module and a two-way first-come-first-served Merge module.

2. Single Track Handshake Signaling

A particularly interesting form of forward and back NoC handshake protocol is called "single track" signaling [3][12][2]. In a single track protocol a single wire carries both the forward and backward handshake signals. A sender briefly drives the wire to one logic level, signaling the presence of data on adjacent data wires. The receiver, noticing that change in the wire's state, copies the corresponding data and then briefly drives the wire to the other logic state to indicate that it has absorbed the data values.

Single track signaling is attractive not only because a single wire occupies less space, but also because single track signaling consumes minimum energy per cycle. A transition must pass in each direction, and the single wire does exactly that with automatic return to the initial state after each handshake. These two advantages are offset by a timing issue inherent in the word "briefly." Because sender and receiver share the signaling wire and drive it in opposite directions, each must take care to cease its drive promptly so that the other has free use of the wire. Proper operation of single-track systems depends on the proper behavior of each participant to drive only briefly. The GasP family of asynchronous control circuits discussed in this paper is such a "single track" system [9].

The 6-4 GasP family gets its name from the six logic gates in the forward direction between each stage and its successor, gates A B C D E F shown in Fig. 1, and the four logic gates in the backward direction, gates A B C X in the figure. The longer delay appears in the forward direction because copying data forward requires action on the part of data latches, whereas moving a “bubble” backwards to indicate that a register has become empty requires none.

Note also that the 6-4 GasP circuit has sets of five inverting gates that form closed loops. One such loop, called the successor loop, involves gates A B C D E. Whenever the inputs to the AND function of gate A cause gate A to act, the successor loop will change the state of the successor state wire in such a way as to discontinue that action. Similarly, the five gates A B C X F form a predecessor loop that serves a similar purpose. The five gates in each loop form, in effect, a pair of five-inverter ring-oscillators coupled to neighbors by the AND logic function inside the GasP module and the state wires through which the module communicates with its neighbors.

2.1. Timing in Single Track Circuits

Each participant in a single-track signaling protocol must cease driving the state wire soon enough to make room for the action of the other participant. Were a participant to drive the wire for too long a time, both might drive it concurrently in opposite directions, consuming unnecessary energy and producing an indeterminate logic signal. How is this to be avoided? Some single-track systems [3][2] make use of the analog properties of the state wire. Each participant drives the wire “long enough” for it to pass some threshold voltage that will alert the other participant. Other single-track systems, including GasP, depend on the relative timing of logic gates to avoid drive conflict at the state wire. Careful choice of the transistor widths in GasP circuits enables the two participants to operate quickly while avoiding conflict.

The transistors in the 6-4 GasP circuits reported here are chosen to be strong enough so that each logic gate has approximately the same delay. This is possible because all but two of the logic gates drive fixed loads. The AND function, called A in Fig. 1, drives only its own output capacitance, the capacitance of the relatively short wire to the inverter called B, and the input capacitance of inverter B. Likewise, B drives only itself, a short wire, and inverter C. In addition to driving both inverter D and the NMOS transistor X and the wires to them, inverter C must drive the rather large load presented by the long control wire to the many latches that will capture the data. The figure labels this load L_1. Thus inverter C tends to be rather large, but its load is the same in every module.

Only the two lone transistors, E and X, drive loads that vary from module to module. Their major load is the capacitance of the state wire between modules, labeled L_2 in the figure. If the neighbor module happens to be nearby, L_2 will be small, but if it happens to be far away, L_2 may be much larger. The module designer cannot know the exact length of the state wire until the module has been placed in the system. Thus the designer of a system of GasP modules faces a choice; either opt for identical modules with variable delay or opt for custom modules with constant delay.

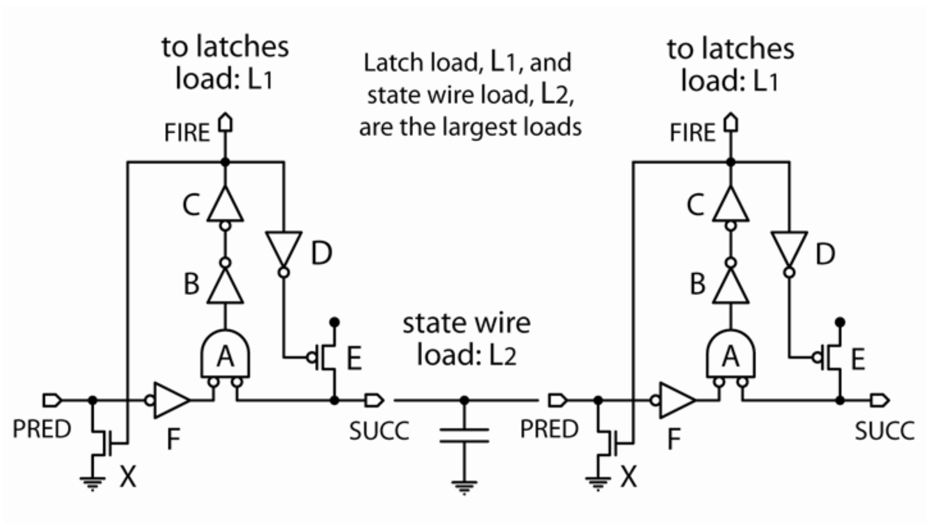


Fig. 1. Two stages of 6-4 GasP. From AND to AND, the forward path is 6 gates: ABCDEF; the backward path is 4 gates: ABCX; the successor loop is 5 gates: ABCDE; and the predecessor loop is 5 gates: ABCXF.

This is an unpleasant choice because both options present problems. The choice of constant delay requires specialized transistor sizing and thus unique layout for each module after establishing the distance to its neighbors. The choice of identical modules causes variation in performance with distance. Although some variation in performance is acceptable in an asynchronous system, excessive variation may cause failures as we shall describe below.

3. The Problem

Faced with the choice between identical modules and constant delay, the VLSI Research Group at Sun Microsystems chose identical modules. The Sun group's earlier efforts with 4-2 GasP [12] had shown them how hard it is to tailor the transistors in each module to the length of the wires between modules. First, such customization requires very late binding of transistor widths after layout is complete, and second, it requires calculation of wire capacitance. Accurate calculation of wire capacitance is very difficult because such calculation requires complete knowledge of what structures lie adjacent to the wire throughout its length. The Sun group now uses 6-4 GasP rather than the faster but more demanding 4-2 GasP precisely to relax the tight constraints on delay presented by the faster 4-2 GasP circuits. The problem remains, however, to understand the conditions under which an identical set of 6-4 GasP modules will operate correctly in the face of variable distance between modules. *This paper addresses only the handshake operations, omitting any discussion of data validity.*

The Sun group did electrical simulations of the behavior of modules in environments with long and short state wires. These simulations used the SPICE electrical simulation tool to simulate the behavior of each and every transistor, wire capacitance, and logic function, a fairly laborious and compute-intensive process. The results of these simulations showed proper operation over the range of state wire lengths actually needed in the test chip. Now that the chip has been tested, we can happily report that it does work. However, to use the 6-4 GasP modules in a more complex design requires a deeper understanding of the conditions under which such a design will be valid. Because the GasP modules in a more complex design will be placed automatically, such understanding could easily be converted into constraints on the automatic placement software.

With this in mind, Sun was eager to enlist the help of the authors of this paper. Together we sought to identify the failure mechanisms possible to the 6-4 GasP design in a much wider range of environments. What can be said about the conditions under which GasP circuits will or will not operate properly? It is always useful to have an outside group, the “red team,” identify flaws in a design. A red team has no vested interest in showing that the design is correct and every motive to show how it will fail. Sun felt that the value contributed by a red team would outweigh any loss of proprietary information that might result.

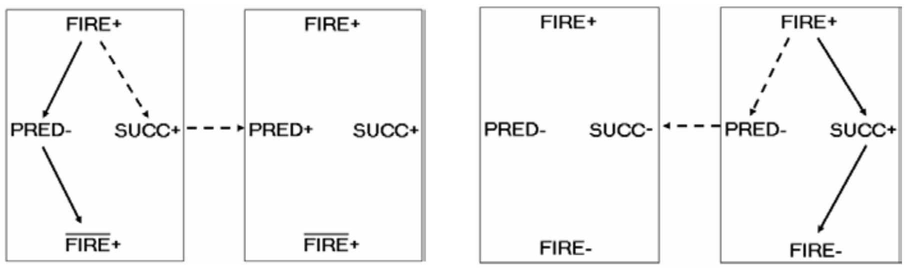
4. Relative Timing

Our timing verification approach is based on relative timing constraints in which we explicitly identify timing constraints in the form of ordering of signal transitions within the circuit [8]. In addition to providing insight into the circuit behavior, we are exploring whether we can verify asynchronous circuits using standard commercial static timing analysis tools, capturing the margins necessary to cover all sources of delay variability that asynchronous circuits may exhibit.

Using this relative timing approach, we identified and formally modeled the intended behavior and circuit implementation of the GasP circuits and identified two important failure modes. Both these failure modes occur because the two loops of logic gates in each GasP module meet at the AND function. Completion of either of the loops shuts off both of them. Failure can result if either of these two loops completes before the other is adequately underway. The timing constraints involve the difference in delay of the two loops.

The constraints are illustrated in Fig. 2 in which the behavior of the GasP design illustrated in Fig. 1 is represented by up- and down-going transitions (+ or -) of key signals to the latches and neighboring GasP stages: FIRE, PRED, and SUCC. The constraints are illustrated by the pair of paths in Fig. 2(a1)-(a2) and Fig. 2(b1)-(b2); the delay of the dotted path must be smaller than the delay of the solid path. These timing constraints help ensure a full rail-to-rail swing on the state wires between stages by turning the drivers on long enough to drive the state wire high or low before turning the drivers off.

The first timing constraint, illustrated in Fig. 2(a1)-(a2), states that PRED of the next GasP stage should go high before PMOS gate E of the current stage stops driving



(a1) (a2) (b1) (b2)

Fig. 2. Relative Timing Constraints on GasP.

it; see Fig. 1. Presence of a large state wire load from SUCC of stage (a1) to PRED of stage (a2) and a small state wire load on PRED of stage (a1) can lead to a violation of this constraint. Moreover, if PMOS gate E in stage (a1) fails to drive the state wire all the way up before the predecessor loop shuts E off, data moving forward may be lost.

The second timing constraint, illustrated in Fig. 2(b1)-(b2), states that SUCC of the previous GasP stage should go low before NMOS gate X of the current stage stops driving it; see Fig. 1. Presence of a large state wire load from PRED of stage (b2) to SUCC of stage (b1) and a small state wire load on SUCC of stage (b2) can lead to a violation of this constraint. Moreover, if NMOS gate X in stage (b2) fails to drive the state wire all the way down before the successor loop shuts X off, bubbles moving backward may be lost, resulting in duplication of data.

Notice that each constraint involves the relative ordering of two actions on the state wire that leads to an adjacent module. When the first of the two actions completes, it turns off the driving action of both loops. Thus a very fast predecessor loop may prematurely terminate the drive of a slower successor loop, and vice versa. Armed with this understanding the question remains: “How much faster?”

5. Applying Logical Effort

The Theory of Logical Effort [13] offers a formal calculus to compute the delay in logic gates. According to the Logical Effort model, the delay in a logic gate depends on the logic function it implements, and is directly proportional to the load that it drives and inversely proportional to the width of its transistors. In general, a logic gate can be made faster by making its transistors wider or by reducing the load that it drives.

One result of this analysis is that fastest overall operation of a three-gate string of logic is obtained by making the “size” of the central gate the geometric mean of the “sizes” of the first and last gate. If the central gate is too small, the first gate will operate faster, but the central gate will be too slow. If the central gate is too large, the central gate will operate faster, but the first gate will be too slow. Logical Effort teaches that the “size” of a gate depends not only on the width of its transistors, but also on its logic function.

According to the Logical Effort model, the delay in a logic gate is given by:

$$d = gh + p \tag{1}$$

where d is the delay in the gate in normalized time units, g is the Logical Effort of the gate, h is the ratio of the load it drives to the input load it presents to its driver, and p is a constant delay that arises from the diffusion capacitance connected to the gate's output. The Logical Effort, g, depends on the logic function implemented by the gate. Table 1 shows the logical effort of some logic gates.

The Theory of Logical Effort ignores the time it takes for signals to change from one logic level to another. More accurate models can take the "rise time," or its reciprocal, "slew rate," into account at the cost of computational complexity. More accurate models are particularly important for estimating the delay introduced by the resistance and capacitance of long wires. A long wire not only adds capacitive load to its driver, retarding the signal at its source, but also delivers any change in logic level with degraded rise or fall time. The available tools that model the effects of rise and fall time degradation in long wires are suitable only for synchronous systems. Peter Beerel and his students at USC seek to adapt such tools to asynchronous systems. The analysis offered in this paper models only how the capacitance of the wires retards the gate that drives them and ignores the delay inherent in the wire itself. The simpler Logical Effort model used in this paper is valid because none of the wires considered here is long enough to present significant electrical resistance.

Table 1. The Logical Effort for some simple logic gates.

Gate type	Number and width of NMOS	Number and width of PMOS	Total transistor width	Width seen per input	Logical Effort per input
Inverter	1 @ W	1 @ 2W	3W	3W	1 (norm.)
2 input NAND	2 @ 2W	2 @ 2W	8W	4W	4/3
2 input NOR	2 @ W	2 @ 4W	10W	5W	5/3
2 input XOR	4 @ 2W	4 @ 4W	24W	12W	4
NMOS transistor	1 @ W	none	W	W	1/3
PMOS transistor	none	1 @ 2W	2W	2W	2/3

The logical efforts in Table 1 are normalized to the inverter whose logical effort is arbitrarily set to one. A NAND gate is worse than an inverter at providing electrical amplification by the ratio 4 to 3, and a NOR gate is worse by the ratio 5 to 3. The difference between NAND and NOR comes from the fact that in most CMOS processes, a PMOS transistor conducts about half as much current as an NMOS transistor of the same width. Except for the single transistor gates, these logical effort numbers are at least one because, unlike inverters, logic gates must use transistors in series or parallel combinations. Transistors in series conduct current less well than individual transistors and transistors in parallel require more input charge than does a single transistor. XOR is a particularly nasty function that involves parallel combinations of series transistors. Another way of thinking about this is that the output of an XOR logic gate will change in response to any single input change. This makes XOR a "more complex" logic function than either NAND or NOR and gives it correspondingly higher logical effort.

In both the NAND and the NOR logic gates, the AND function comes from two series transistors. Both must conduct before current can flow though the series pair to the output. In the NAND logic gate, the AND function appears in NMOS transistors, whereas in the NOR logic gate, the AND function appears in PMOS transistors. The symbol used for logic gate A in Fig. 1 indicates that PMOS transistors provide its AND function and therefore the logical effort of gate A is 5/3.

One of us, Prasad Joshi, applied the Logical Effort model to the 6-4 GasP circuits. He built a spreadsheet much like that shown in Table 2 to calculate the delays earlier identified as critical to correct operation. Table 2 greatly simplifies the sizes and wire lengths in order to offer simple numbers that are, nevertheless, approximately correct. The size S of a gate is a measure of its drive strength and depends on the width of its transistors. Table 2 normalizes wire lengths as the size of an inverter that would present an equal load. Logical effort enters Table 2 as an increase in input load rather than as increased delay.

Table 2. Logical Effort calculations of delays in the 6-4 GasP module of Fig. 1. The three columns with bold numbers are fixed by design: size S and self-delay P follow from the gate implementation and the gate complexity, and wire load WL follows from the wire dimensions. Note that in the calculation of the self delay for E and X we count the diffusion output capacitance of both E and X, because both their outputs connect to the state wire. Using Table 1 for the logical efforts, IL = (S / logical effort of the gate). The formulas for the delays are as follows: $D_1 = GL/S$, $D_2 = WL/S$, Total delay = $D_1 + D_2 + P$.

Gate Name	**Size (S)**	**Load per input (IL)**	**Drives next gates and wire**	**Next gate load (GL)**	**Delay from GL (D_1)**	**Self delay (P)**	**Next wire load (WL)**	**Delay from WL (D_2)**	**Total delay**
A	**18**	30	B+wire	40	2.2	**2**	**9**	0.5	4.7
B	**40**	40	C+wire	100	2.5	**1**	**20**	0.5	4
C	**100**	100	D+X+wire	20+20 +200	2.4	**1**	**100**	1	4.4
D	**20**	20	E+wire	40	2	**1**	**20**	1	4
E L_2=15	**60**	40	A+F+ L_2	30+10	0.67	**with X = 1**	**15**	0.25	1.9
E L_2=150	**60**	40	A+F+ L_2	30+10	0.67	**with X = 1**	**150**	2.5	4.2
F	**10**	10	A+wire	30	3	**1**	**5**	0.5	4.5
X L_2=15	**60**	20	A+F+ L_2	30+10	0.67	**with E = 1**	**15**	0.25	1.9
X L_2=150	**60**	20	A+F+ L_2	30+10	0.67	**with E = 1**	**150**	2.5	4.2

The longer wire that makes L_2 = 150 was the design target. Notice that for this case the gates all have approximately the same total delay, in the range 4 to 5.

Unlike Table 2, which shows only two values of L_2, Prasad's spreadsheet shows all delays for a wide variety of lengths of the predecessor and successor state wires. Although we initially thought that the *ratio* of the lengths of these wires might be important, Prasad's spreadsheet revealed instead that the *difference* in the lengths of these wires matters.

Table 3 summarizes the total delay for several paths of interest. A tenfold change in L_2 results in a change in these path delays of only about 10% to 15% because of the fixed delays of gates A B C D and F.

Table 3. Total delay for various paths.

Path name	Path	Delay, L_2=15	Delay, L_2=150	Difference
Forward latency	A B C D E F	23.5	25.8	9.6 %
Backward latency	A B C X	15	17.3	15 %
Successor loop	A B C D E	19	21.3	12 %
Predecessor loop	A B C X F	19.5	21.8	12 %

6. The Infinity Chip

The VLSI group at Sun Microsystems designed a test chip called "Infinity" to demonstrate 6-4 GasP modules suitable for use in a network on a chip [10]. Infinity tests the operation, speed, and power consumption of the various types of modules required, with special focus on the modules for branching and merging networks. The Branch and Merge modules are electrically more complex than the simple 6-4 GasP circuit shown in Fig. 1, but have many similar elements. Like the circuit of Fig. 1, the Branch and Merge modules each provide 6 gates in the forward direction and 4 in the backward direction. Moreover, each logic gate used in the Branch and Merge modules has a delay in the range of 4 to 5 just like the logic gates in the simple GasP module[1]. Thus we expect performance from the Branch and Merge modules similar to that of the simpler circuits.

Logically, the Infinity chip consists of two rings of 100 GasP modules each, as shown in Fig. 3. Each module passes on a word of 52 bits of data whose latches and the wire to reach them form the large load L_1 in Fig. 1. The rings share a common center section from Merge to Branch inclusive of 50 modules. At the start of the center section a Merge module combines input from the two rings upon demand, using an arbiter to decide which input was presented first. At the end of the center section, a Branch module separates the data elements according to one of the bits that they carry. This bit marks each data element as "belonging" to either the left or the right ring.

[1] The delay in the Merge module will be longer in the rare case of contention at its arbiter.

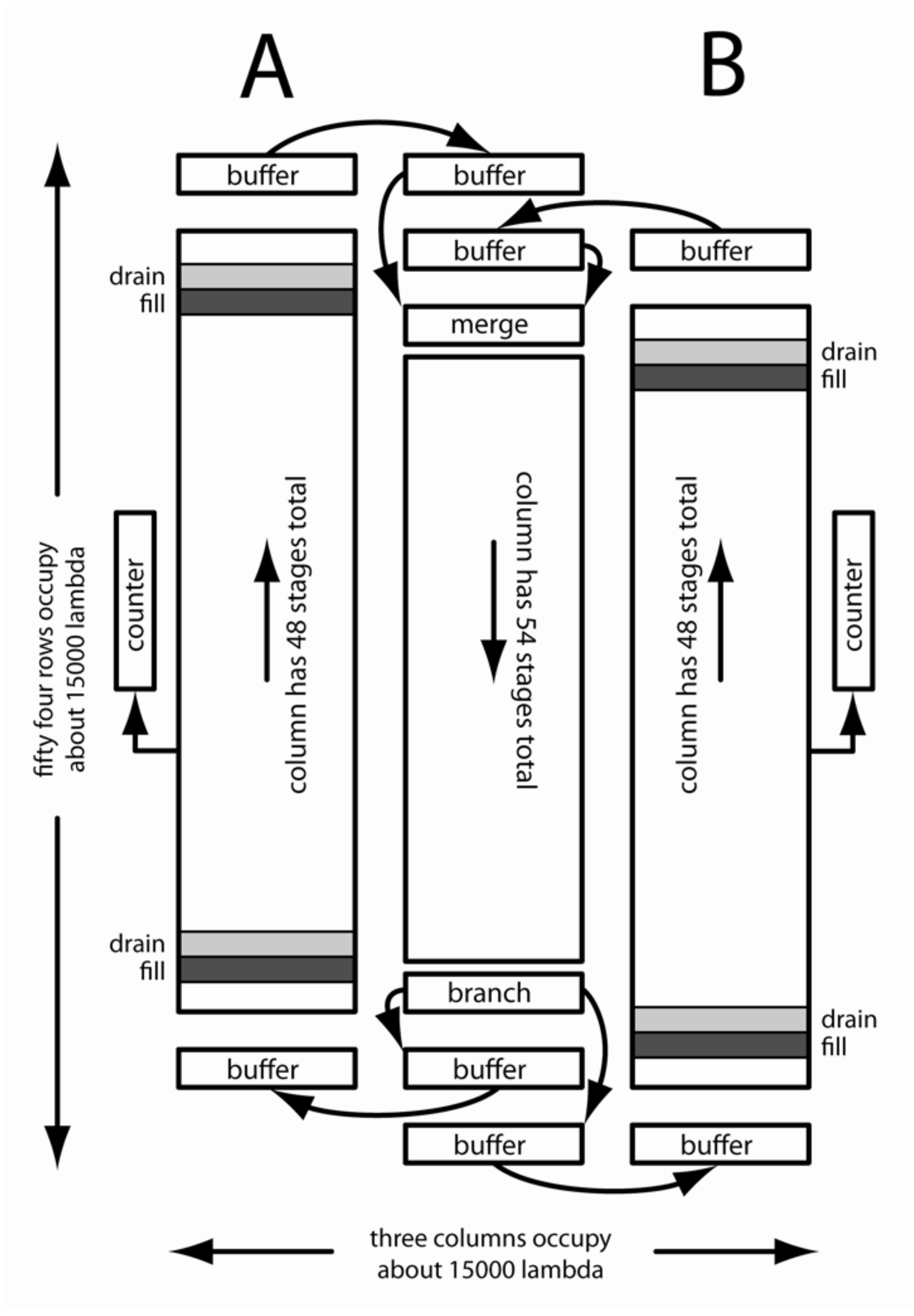

A
B
buffer
buffer
buffer
buffer
buffer
buffer
buffer
buffer
merge
branch
drain
fill
drain
fill
drain
fill
drain
fill
counter
counter
column has 48 stages total
column has 54 stages total
column has 48 stages total
fifty four rows occupy about 15000 lambda
three columns occupy about 15000 lambda

Fig. 3. Layout view of the Infinity experiment. It occupies an area of about 0.5 mm^2.

Geometrically, the Infinity chip is arranged as three columns of modules; see Fig. 3. Each module is 288 lambda[2] high and about 5000 lambda wide. Their width requires that the center-to-center distance of the columns exceed 5000 lambda. The center column is the shared part of the two rings; data flows down through the center column from the Merge module near its top to the Branch module near its bottom. Data flows back upwards through the outer columns, thus completing each of the two logical rings. Each outer column includes a counter to measure how many data elements pass through the column. Each outer column also includes two “proper stoppers” to properly fill and drain data elements from each ring at the beginning and end of each experiment. Low speed scan techniques load data into the rings prior to each experiment and unload the data afterwards to check for errors.

Infinity’s designer, Ivan Sutherland, anticipated that the task of copying data elements from one column to another was “harder” than passing data within the columns because it requires state wires that are approximately 16 times longer, namely about 5000 rather than about 300 lambda long. To avoid placing a large electrical burden on the Branch module in addition to its inherently larger logical burden, he chose to isolate the Branch from the column-crossing section. Thus it is, as Fig. 3 shows, that the pair of buffer stages immediately below the Branch drives data to the adjacent column. A similar pair of buffer stages above the Merge module offers the Merge module similar protection from excess electrical load.

7. Experimental Results

Performance experiments with asynchronous rings typically measure throughput versus occupancy [14][7][5]. It is easy to see that plots from such experiments show a roughly triangular shape. If the ring contains no data, there can be no throughput. Similarly, if every stage in the ring contains a data element, none can move, and again there’s no throughput.

Let us first consider the low-occupancy side of such a plot. A single data element passes once around the ring by passing through each and every module once. Two data elements make the same trip in about the same time, achieving twice the throughput. Three data elements offer three times the throughput, and so forth. Thus for small occupancy, throughput increases linearly with occupancy.

Now let us consider the high-occupancy part of such a plot. If the ring is completely full, there’s no throughput. If all but one module is occupied, the blank space, or bubble, can circulate backwards, just as space for your automobile moves backward when you advance one car length on a very crowded highway. Two bubbles make a complete circuit of the ring backwards in about the same time as one, and so throughput increases roughly linearly with the number of bubbles. The bubble side of the graph is steeper than the data side because 6-4 GasP circuits move bubbles faster than data. A 6-4 GasP circuit moves data forward with a latency of 6 logic gates and moves bubbles backward with a latency of 4 logic gates.

[2] Lambda is a distance measure characteristic of the manufacturing process.

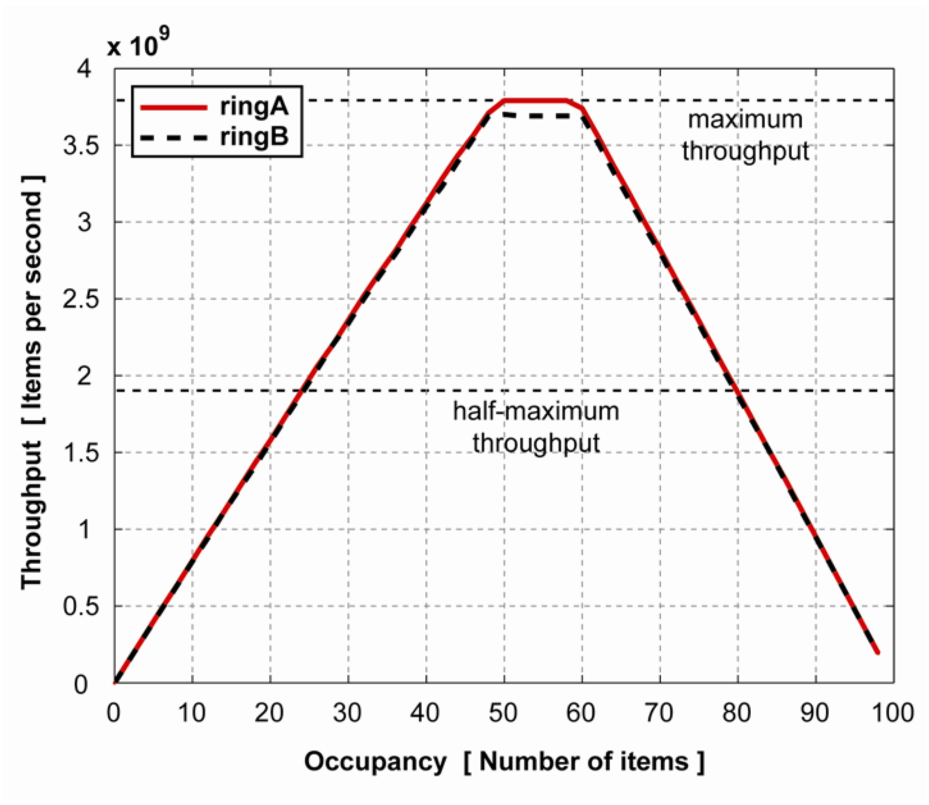


Fig. 4. Throughput versus occupancy for Infinity's rings operated separately.

Somewhere in the middle these two linear regions meet, giving the roughly triangular shape often seen in such plots. One might expect a sharply defined maximum at an occupancy defined by the slopes of the two linear regions, but such a sharp top is rarely observed. The shape at near maximum throughput is defined by the details of the delay in the AND function when both its inputs change at the same time. Real AND gates suffer more delay when both inputs change together than when one input changes well before the other [5]. This effect gives the plot a smooth curved top.

It also sometimes happens that a single module in the ring or a few modules are inherently slower than the others. A slow stage places a maximum limit on throughput, giving the plot a flat top, as can be seen in Fig. 4 which shows throughput versus occupancy for each of Infinity's rings operated separately. We notice from the figure that each of Infinity's rings is limited to a throughput of about 3.8 * 10^9 data items per second. Extrapolation of the two linear portions of the plot, however, suggest that speeds of about 4.2 * 10^9 data items per second might be possible, a 10% increase.

7.1. Single-Column Experiment

A separate and simpler experiment on the same chip as the Infinity experiment and using the same 6-4 GasP modules contains only a single FIFO ring, with all its modules arranged in a single column. Upward bound and downward bound stages alternate in the column, so that no stage is more than two modules away from its logical neighbors. This simpler experiment avoids the column change required in Infinity. The measured peak throughput of this simpler experiment is about $4.2 * 10^9$ data items per second, about 10% higher than either of Infinity's two rings operating alone.

Naturally, we want to know which module in the Infinity design limits its performance. The first suspects, of course, are the Branch and Merge modules because of their greater complexity. Neither of these modules appears in the simpler experiment.

7.2. The Cross-Column Hypothesis

Alternately, the speed limit might be the result of the long state wire required to pass from one column to the adjacent column. Indeed, Infinity includes special buffer modules to drive this wire, but their PMOS and NMOS drive transistors, E and X in Fig. 1 are the same width as those in other modules that drive shorter wires. Moreover, the logical effort calculations summarized in Table 3 suggest that the greater wire length involved might result in a 10% to 15% decrease in speed.

Thus we offer the hypothesis that the long state wires for the cross-column transfer limit Infinity's performance. This hypothesis renders harmless the greater logical complexity of the Branch and Merge modules. Is there an experiment we can do with Infinity to support this Cross-Column Hypothesis?

7.2.1. The Supporting Experiment

Indeed there is. Consider the case where each ring contains data elements that mingle as they pass through the center column. The Branch module delivers data to the two side columns via the two buffer stages immediately below it. If it should happen that the data elements alternate as they pass through the center section, the Branch module will deliver them alternately to the two sides. In this case, the data rate needed to move from the center column to either side would be only one half of the data rate sustained in the center column. Such a reduction in data rate would leave twice as much time for crossing to the adjacent column, much more than the task requires. Any small excess delay of 10% to 15% for crossing from one column to the next will thus be masked. Of course, to achieve its maximum throughput the Branch module must receive a sufficiently fast supply of elements of alternate types.

Similarly, the Merge module at the top of the center column receives data elements from each ring via the two buffer stages immediately above it; see Fig. 3. The Merge module contains an arbiter to decide which element arrives first and dispatch it down the center column. An element observed slightly later at the other input to the Merge must wait for the next cycle of the Merge module. If there is always a data element waiting at the beginning of each cycle of the Merge module, we say that the Merge module is "saturated". In saturation it happens that the electrical properties of its arbiter cause the Merge module to send forward data elements alternately from its two inputs.

When the Merge module is saturated, it will accept data elements from an individual side column at only half of its maximum data rate. This reduction in data rate leaves twice as much time for crossing from the side column, much more than the task requires. Thus, just as alternate delivery by the Branch module masks the cross-column delay at the bottom, so alternate receipt by the Merge module likewise masks the cross-column delay at the top of the center column. Moreover, because the Merge module forwards data alternately from the two sides, the center stream meets the alternating condition assumed for the Branch module.

7.2.2. Confirming the Hypothesis

How many data elements must there be in the side column to achieve at least half the maximum throughput? We can discover the answer by examining Fig. 4 which shows the throughput for any given occupancy. If we need half the maximum throughput, we must have occupancy of at least 24% and not more than 80%. Why is this? With less occupancy there aren't enough data elements to sustain the throughput, and with greater occupancy there is too much congestion. Applying the 24% to 80% numbers to the 50 modules of the side column and its buffers requires them to contain at least 12 data elements and not more than 40, a maximum that leaves enough bubbles, namely 10, to achieve the required throughput.

In addition, of course, the 50 modules of the shared center column at maximum throughput contain a total of about 60% * 50 = 30 data elements. Because these alternate from the two rings, half of them must belong to each ring. The total number of elements in each ring is the sum of its unshared data elements and its shared data elements, a number that lies between 12 + 15 = 27 and 40 + 15 = 55. We expect occupancy in each of the two rings in this range to mask any cross-column delay.

Unfortunately, Infinity lacks a counter able to measure directly the throughput of the center column. However, the throughput of the center column must be the sum of the throughputs of the two side columns. Thus we can use the sum of the two side throughputs to test the hypothesis that the cross-column delay limits the single-ring throughput of Infinity.

Experimental confirmation of the Cross-Column Hypothesis comes from an experiment in which each ring holds 45 data elements. In this condition we measure a combined throughput of about $4.2 * 10^9$ data elements per second. This higher combined throughput is very similar to that of the simpler single-column experiment.

Analytic confirmation of the Cross-Column Hypothesis comes from Table 3 which suggests that the longer state wires required for crossing from one column to another will retard the buffer modules about 10% to 15%.

The near agreement between formal analysis and chip measurements give us confidence that the flat top of Fig. 4 comes from delay in the cross-column buffers and that the Branch and Merge modules can operate at their full design speed.

8. Conclusions

One might be tempted to conclude that the cross-column drivers in Infinity are too small. It would have been possible to use wider drive transistors in the buffer stages to speed up the cross-column transfer. However, examination of Table 2 illustrates otherwise. Notice that for the long wire case in Table 2 the delays of transistors E and X are about the same as the delays of all the other logic gates. In fact, the width of transistors E and X was chosen for the long wire case, but the same E and X widths appear in all stages. The close proximity of stages in a single column makes the state wires between them too short for the chosen width of their drive transistors. Thus, rather than concluding that Infinity's cross-column drivers operate too slowly, we should conclude that the single-column modules, including the Branch and Merge, operate too quickly.

We can improve the design by providing a few additional module types with smaller transistors for driving short state wires. Such a design would have reduced transistor width, and thus would consume less energy. All its stages would operate with about the same delay, albeit with the longer delay of the cross-column stages.

The most important result from this work is that a limited set of module types can provide reliable operation for an extended range of state wire lengths. GasP circuits fail when the delays in the predecessor and successor loops differ by too much. Notice, however, that each loop includes the delay of gates A B and C in Fig. 1. Moreover, the delays of gates D and F can be made approximately the same. Thus for such a timing failure to occur, the delay in gate E or X must exceed the combined delay of the other logic gates, A B C and D or F. It would require an excessively long state wire to delay E or X so much, a wire long enough to span many of Infinity's columns. Moreover such a wire would have too much electrical resistance to be useful. The analysis offered here bolsters our confidence that the 6-4 GasP family of control circuits may prove applicable to a wide variety of NoC conditions.

Acknowledgements. The authors wish to thank Intel and Sun and the sponsors of SRC for their support of this work. Our thanks go also to the members of the VLSI Research group at Sun Microsystems for design, layout, and test of the Infinity chip. Special thanks to Russell Kao and Hesam Fathi Moghadam who provided test results for Infinity, and to Igor Benko and Jon Lexau who helped with its final layout and fabrication. The authors would also like to thank Pankaj Golani, a PhD student at USC, for very insightful discussions on relative timing constraints for more general single-track handshaking.

References

1. Beerel, P.A., Roncken, M.E.: Low Power and Energy Efficient Asynchronous Design. Journal of Low Power Electronics, Vol. 3, No. 3, pp. 234–253 (2007)

2. Beerel, P.A., Ferretti, M.: High Performance Asynchronous Design Using Single-Track Full-Buffer Standard Cells. IEEE Journal of Solid-State Circuits, Vol. 41, No. 6, pp.1444–1454 (2006)
3. van Berkel, K., Bink, A.: Single-Track Handshake Signaling with Application to Micropipelines and Handshake Circuits. In: 2nd IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 122–133 (1996)
4. Coates, W.S., Lexau, J.K., Jones, I.W., Fairbanks, S.M., and Sutherland, I.E.: FLEETzero: An Asynchronous Switching Experiment. In: 7th IEEE International Symposium on Asynchronous Circuits and Systems, pp. 173–182 (2001)
5. Ebergen, J.C., Fairbanks, S., Sutherland, I.E.: Predicting Performance of Micropipelines Using Charlie Diagrams. In: 4th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 238–246 (1998)
6. Golani, P., and Beerel, P.A.: Back-Annotation in High-Speed Asynchronous Design. Journal of Low Power Electronics, Vol. 2, No. 1, pp. 37–44 (2006)
7. Lines, A.M.: Pipelined Asynchronous Circuits. Master's thesis, Caltech CSTR:1988.cs-tr-95-21, California Institute of Technology (1996)
8. Stevens, K.S., Ginosar, R. Rotem, S.: Relative Timing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 11, No. 1, pp. 129–140 (2003)
9. Sutherland, I.: A Six-Four GasP Tutorial. Technical Report, UCIES#2007-is49, see FLEET web site [15] (2007)
10. Sutherland, I.: Infinity: A Proposed Test Chip. Technical Report, UCIES#2007-is46, see FLEET web site [15] (2007)
11. Sutherland, I.: FLEET – A One-Instruction Computer, Technical Report, UCIES#2006-is30, see FLEET web site [15] (2006)
12. Sutherland, I., and Fairbanks, S.: GasP: A Minimal FIFO Control. In: 7th IEEE International Symposium on Asynchronous Circuits and Systems, pp. 46–53 (2001)
13. Sutherland, I., Sproull, B., Harris, D.: Logical Effort: Designing Fast CMOS Circuits. Morgan Kaufmann, San Francisco (1999)
14. Williams, T.E.: Analyzing and Improving the Latency and Throughput Performance of Self-Timed Pipelines and Rings. In: IEEE International Symposium on Circuits and Systems, Vol. 2, pp. 665–668 (1992)
15. Fleet web site, http://research.cs.berkeley.edu/class/fleet/

About the Authors

Prasad Joshi is a graduate student at the University of Southern California. He came to USC after completing a B.E. degree in 2006 at the University of Mumbai, India.

Peter Beerel is an Associate Professor of Electrical Engineering at the University of Southern California and is Prasad's advisor. Peter completed his PhD degree in Electrical Engineering at Stanford University in 1994.

Marly Roncken is employed by Intel Corporation, and assigned as Researcher in Residence at the University of California, Berkeley. She has been involved in asynchronous systems since shortly after receiving her master's degree in Mathematics from the University of Utrecht in 1985. Before joining Intel in 1997 she worked for Philips Research in Eindhoven, The Netherlands.

Ivan Sutherland is a Fellow at Sun Microsystems, assigned by Sun to work at the University of California, Berkeley. He received his PhD degree in Electrical Engineering at MIT in 1963.

The Project

The asynchronous VLSI/CAD group at the University of Southern California (USC) led by Prof. Peter Beerel does research in various aspects of making asynchronous circuits a viable alternative to synchronous design. One of its current goals is to extend the application of standard static timing analysis techniques to non-traditional circuits, including high-speed asynchronous pipelines, see e.g. [6]. Heretofore nearly all timing verification tools have been built to validate the operation of synchronous circuits in which a global clock provides a common rhythm of operation to an entire system. In asynchronous designs, however, separate parts of the system each use their own timing signals, thus making it very hard to apply the standard validation tools. The USC group seeks ways to simplify timing verification for asynchronous systems.

The USC group is sponsored in part by the Global Research Collaboration (GRC) program of the Semiconductor Research Consortium (SRC). GRC provides for a global forum for pre-competitive collaboration among all segments of the semiconductor industry, universities and government agencies. Marly Roncken mentors the SRC work at USC for Intel Corporation, and was seeking additional asynchronous design examples to which the USC timing validation tools and understanding might apply. She found these at the University of California, Berkeley (UCB), where Ivan Sutherland has an ongoing collaborative research project between Sun Microsystems and UCB, called Fleet [4][11][15]. Meanwhile, the VLSI Research Group at Sun Microsystems had fabricated an asynchronous test chip for Fleet, called Infinity, in 90 nanometer TSMC technology.

The fortuitous juxtaposition of these activities brought the four authors together. Marly's earlier association with Prof. Willem-Paul de Roever, as his master's student at the University of Utrecht, leads us to offer this paper in honor of Willem-Paul's "Declaration of Independence" through retirement from the University of Kiel on 4 July 2008. July 4th is also a very important holiday in the United States, being the anniversary of the date when the United States declared independence from England.