GasP: A Minimal FIFO Control

Ivan Sutherland and Scott Fairbanks
Sun Microsystems Laboratories, Palo Alto, California, USA

Abstract

The GasP family of asynchronous circuits provides controls for simple pipelines, for branching and joining pipelines, for round-robin scatter and gather, for data-dependent scatter and gather, and for join on demand through arbitration. The family is designed so that each stage operates at the speed of a three-inverter ring oscillator. Test chips in 0.35 micron technology exhibit throughput in excess of 1.5 giga data items per second (GDI/s).

Between GasP pipeline stages a single wire carries both request and acknowledge messages, also recording the FULL or EMPTY state of each pipeline stage. GasP control circuits rely on careful choice of transistor widths to equalize the delay in logic gates. Assurance of uniform gate delays permits use of self-resetting logic forms that have very low logical effort.

1. Definitions

Let us start by considering the meaning of some terms. In referring to the condition of latches, we use the words "transparent" and "opaque" to avoid the ambiguity of "open" and "closed." In referring to the parts of a pipeline we use the words "PLACE" and "PATH" to distinguish two kinds of circuits: a *PLACE* holds data whereas a *PATH* controls the flow of data between PLACEs. Along an asynchronous pipeline PATHs and PLACEs alternate so that each PATH has a predecessor and successor PLACE. The word "stage" describes one alternation, as in "forward latency per stage," but fails to specify a precise boundary. A stage usually includes a complete PLACE, as in "the stage is FULL," and may include either its predecessor PATH or its successor PATH or part of each.

PATH and PLACE in this context sometimes seem counterintuitive. In common use a path along which we walk takes us from place to place, a notion true in our circuits topologically but not geometrically. The geometric difference comes from the ability of every wire in a CMOS circuit to store information: PLACEs include the wires that hold information, and those wires have geometric extent, whereas PATHs include the tiny CMOS transistor switches through which information flows from PLACE to PLACE. Think of a PATH as the door between corridor-like PLACES, a useful metaphor.

2. Introduction

In the past few years our group has sought speed by reducing the complexity of asynchronous control circuits. The resulting control circuits seem to use the fewest transistors required to move data asynchronously through a series of latches, and they run correspondingly fast. Our circuits depend on the designer's *faith* in correct timing "in the small," faith bolstered by the careful timing analysis described in [7]. Our faith in timing extends not only to the "bundled data convention" that assures correct operation of the data path, but also, and to a much greater extent than ever before, in the control circuits as well. Our circuits use *measurement* very sparingly and are therefore asynchronous only "in the large." This is a marked contrast to Delay Independent circuits that use measurement to accommodate wide variation in component delays.

Some years ago Molnar [4] articulated the basic control requirement for an asynchronous pipeline. When two successive PLACEs have the states FULL-EMPTY, the PATH between them must copy data forward and change their states to EMPTY-FULL. Molnar's "asP*" control system used a flip-flop in each PLACE to record its state and a NAND gate in each PATH to detect the conditions prerequisite to action. When the NAND gate "fired", i.e. when its output went LO, it advanced the data and changed the state of the flip flops in the two adjacent PLACEs. Molnar's asP* circuit was symmetric in form, and so its forward latency and reverse latency were the same. The last three letters in the name GasP acknowledge its asP* ancestry.

Seeking to exceed the speed of Molnar's asP* circuits, others in our group built a FIFO using the "transition" or non-return-to-zero control described in the Micropipelines

1522-8681/01 $10.00 © 2001 IEEE

paper [8]; we reported performance for this FIFO in [5]. In the same technology it operates faster than Molnar's asP* circuit, though, to tell the truth, only because of the dual data-path favored by its transition logic convention. Its important contribution, however, was that its forward and reverse latency differed because of the inversion required on only one input of the Muller C-element. Seeking minimum forward latency, we placed the inverter in the reverse path, relegating the longer two-gate delay to the reverse direction and reducing the forward latency to its minimum: one gate-delay per stage.

We have since learned, however, that in very fast asynchronous circuits it is better to make the forward latency long and the reverse latency short. The reason is that it takes time to copy data forward through a latch, but no time at all to move emptiness backwards. To copy data requires a state change at the output of data latches, but to move emptiness requires only a declaration of willingness to overwrite the old data value. In this regard our early Micropipeline design blundered by relegating the longer delay to the reverse direction. As you will see, the GasP circuits described here have a forward latency of four gate-delays and a reverse latency of two gate-delays. Their cycle time is therefore six gate-delays, or precisely the cycle time of a three-inverter ring oscillator.

One can make a symmetric circuit whose forward and reverse latency are both three gate-delays per stage. Indeed, Molnar explored, but never published, such a circuit form which he called "dynamic asP*." An odd number of gate-delays of latency requires that a rising transition in one PLACE correspond to a falling transition in the adjacent PLACEs and therefore requires two forms of PATH circuit. Molnar had difficulty finding a latch fast enough to keep up with a three gate-delay forward latency. The GasP circuits described here have even numbers of gate-delays of both forward and reverse latency, albeit those numbers differ, and GasP circuits can thus use identical, albeit asymmetric, PATH circuits. At first the lack of symmetry caused us conceptual grief, but this has subsided with growing experience.

Behind GasP lie three useful lessons from the theory of Logical Effort [9]. First, eliminating unnecessary transistors tends to make a circuit go faster because every transistor consumes charge at its input, retarding the action of its driver. Second, conditioning transistors in advance can remove the speed-limiting burden of driving them from components that lie on critical timing paths. And third, calculating transistor widths carefully can balance the effort of successive logic gates, not only to reduce overall delay, but also to give all gates nearly uniform delay. A companion paper [7] shows how we use SPICE to calculate transistor widths.

Our design sequence for GasP control circuits is unusual. At the early stages of logic gate design we assume uniform delay for all logic gates in the control circuits. This assumption simplifies logic design and encourages use of self-resetting logic gates with low logical effort. To equalize the performance of each pipeline stage, all control circuits use the same number of logic gates, usually three or five, in every closed loop.

After finishing the logic design we pick transistor widths to realize the assumed uniformity of gate delay, using the methods described in [7]. The final transistor widths take into account post-layout wire loads. The choice of actual gate delay is arbitrary, but choosing a shorter value for the uniform gate delay gives more speed at the cost of more area and more power. The logic loop with the worst logical effort turns out to establish a minimum achievable gate delay. Surprisingly, even substantial capacitive loads from wires have no impact on the theoretical minimum achievable gate delay, because larger transistors could theoretically drive them as quickly as desired. In practice capacitive load of wires does affect the transistor widths required and careful use of post-layout stray capacitance is important to proper function. We have not explored the impact of wire resistance.

The final result of these design steps is a balanced design in which complex pipeline stages use wider transistors to run as quickly as simple pipeline stages, and simple pipeline stages save power and area with narrower transistors, thus running no faster than complex stages.

3. The basic GasP circuit

Each PATH circuit controlling the flow of data between stages must act only when both its predecessor PLACE is FULL and its successor PLACE is EMPTY. The simplest circuit to detect such a condition is a pair of series transistors, preferably of N-type. Embedding these transistors in a NAND gate adds to them a pair of parallel P-type transistors whose sole function is to reset the logic function's output when it no longer need be active; another reset mechanism may be preferable.

When the two series N-type transistors both conduct, their output will go LO and we say the PATH has *fired*. As Molnar pointed out, a PATH must accomplish three things when it fires: (1) it must make data latches momentarily transparent; (2) it must declare its successor stage FULL, and (3) it must declare its predecessor stage EMPTY. Starting another action may also be useful, namely (4) to reset the output of the series N-type transistors to the inactive or HI state.

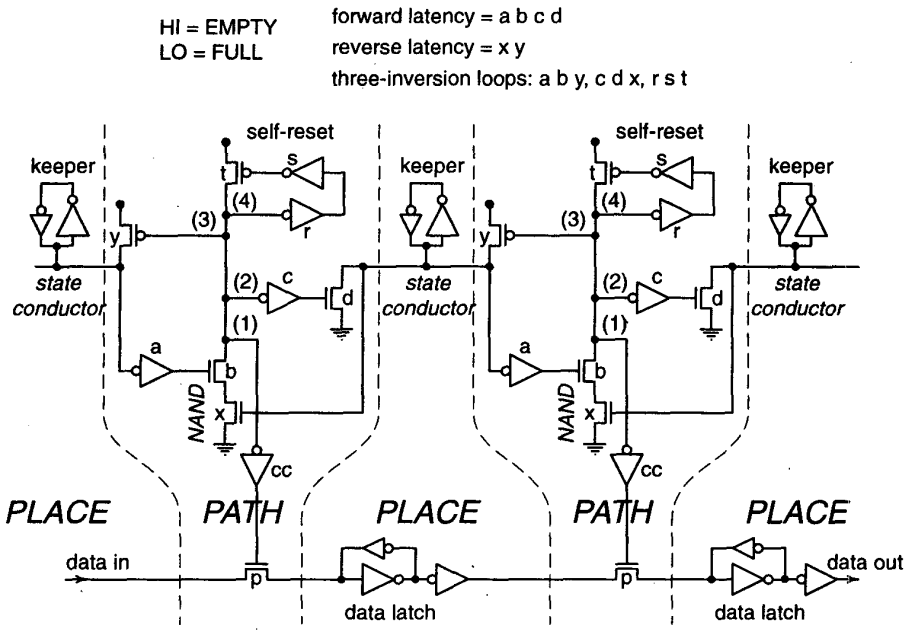

Figure 1. GasP with self-resetting NAND

Instead of storing the FULL or EMPTY state of the predecessor and the successor stages in flip-flops, GasP circuits store each state on a single wire that we call a *state conductor*. This use of a single shared wire is similar to van Berkel's single track handshaking [10]. A master clear signal establishes the initial condition of each state conductor, but the figures in this paper omit master clear. To retain the state for an indefinite period requires also a *keeper*, a pair of small inverters back-to-back, whose opposition to change is easily overcome by the transistors that drive the state conductor.

In a GasP pipeline each PLACE has a state conductor to indicate whether it is FULL or EMPTY. Each PATH attaches an N-type or P-type transistor to its adjacent PLACE's state conductor to force it to the FULL or EMPTY state by driving it HI or LO. Although one may assign either state encoding to any state conductor, it is simplest to understand GasP circuits using the state encoding HI = EMPTY, LO = FULL for all state conductors.

Figure 1 shows three PLACEs and two PATHs using one form of GasP circuit. Each PLACE holds a data item in data latches and holds the FULL or EMPTY state of its data latches on a state conductor with a keeper. The figure omits master clear circuits that set the state of each PLACE to EMPTY. Each PATH contains a GasP control circuit and the pass transistors through which the data flow from PLACE to PLACE.

Given the series stack of N-type transistors and the state encoding, the rest of the PATH circuit is simple. As seen in Figure 1, the output of the N-type transistor NAND stack [b & x] serves the four purposes 1 - 4 stated above, with the circuit connections for each purpose correspondingly labeled: (1) The latch drive signal from inverter [cc] is a short positive pulse suitable for making the N-type transistor pass gates [p] momentarily transparent to copy data forward. (2) Inverter [c] and N-type transistor [d] drive the successor state conductor LO, meaning FULL. (3) P-type transistor [y] drives the predecessor state conductor HI, meaning EMPTY. And (4) delaying inverters [r & s] and P-type transistor [t] reset the NAND function after a short delay.

Notice that the preceding state conductor enters the N-type transistor NAND function at [b] through inverter [a] but the succeeding state conductor enters directly at [x]. This makes the NAND function *detect* the condition LO-HI, which has the meaning FULL-EMPTY.

Notice also that this circuit is built from three loops of three inversions each. The predecessor loop [a b y] involves the predecessor state conductor, an inverter, the NAND stack, and the P-type drive transistor. The successor loop [c d x] involves the successor state conductor, the NAND stack, an inverter, and the N-type drive transistor. The reset loop [r s t] involves the NAND output, two inverters in series, and the P-type reset transistor.

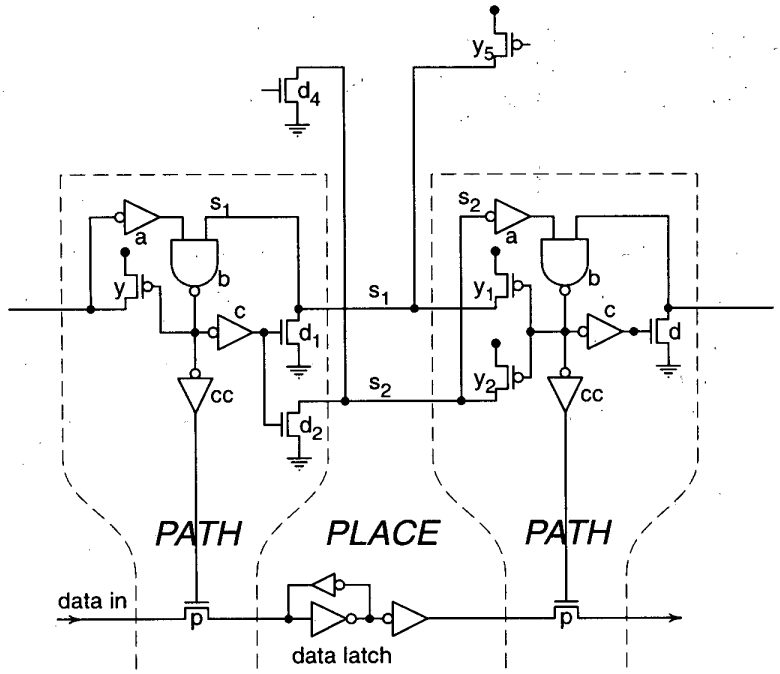

Figure 2. GasP with twin state conductors

The forward latency per stage of this circuit is four gate-delays [a b c d] and its reverse latency per stage is two gate-delays [x y]. In the forward direction, a falling transition, meaning FULL, on the predecessor state conductor travels via inverter [a], the NAND stack [b & x], another inverter [c] and the N-type driving transistor [d] to become a falling transition on the successor state conductor, declaring it FULL. In the reverse direction, a rising transition, meaning EMPTY, on the successor state conductor travels via the NAND stack [x & b] and the P-type driving transistor [y] to become a rising transition on the predecessor state conductor, declaring it EMPTY. The cycle time of six gate-delays is the sum of the forward and reverse latency and is also, as previously mentioned, the characteristic period of the three-inverter loops in the circuit.

As we shall do in some figures, one can draw an ordinary NAND-gate symbol to represent the self-resetting NAND function formed by transistors [x b t] and inverters [r s]. Moreover, one can even replace the self-resetting NAND gate itself with an ordinary NAND gate, depending on the change of state in the adjacent state conductors to reset the NAND gate. Although the self-resetting NAND gate has lower logical effort, an ordinary NAND gate with suitable transistor widths will also serve. In either case, the more compact notation of the standard NAND symbol usually aids understanding.

In complex connections of GasP circuits such as described in [3], a separation of state conductors often proves useful. For example, Figure 2 shows a GasP circuit with two separate state conductors, s1 and s2, in the middle PLACE. The two drive transistors, d1 and d2, in the preceding PATH drive both state conductors LO, and the two drive transistors, [y1] and [y2], in the following PATH drive both state conductors HI. Other path circuits may also establish FULL or EMPTY states as suggested by transistors [d4] and [y5].

Although the PATHs in Figure 2 drive both state conductors from each end, only one PATH monitors the state of each. In this example, only the left PATH monitors the state of s1, via the input to NAND [b], and only the right PATH monitors the state of s2, via the input to its inverter [a]. As seen by the left PATH, s1 has the state encoding HI = EMPTY and LO = not EMPTY; as seen by the right PATH, s2 has the state encoding LO = FULL and HI = not FULL. In a more complex connection several GasP circuits might drive or monitor each state conductor. A companion paper [3] describes many such circuits and a notation for them. It is also possible to make a symmetric form of GasP circuit using twin state conductors to achieve equal forward and reverse delay.

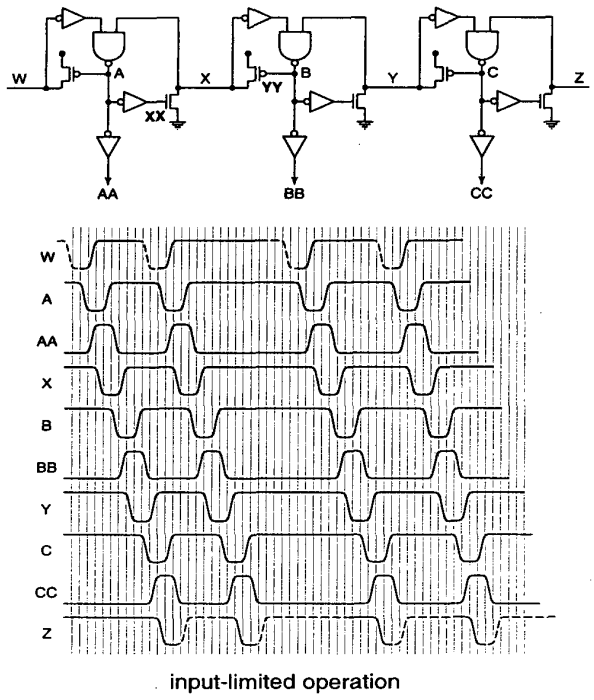

Figure 3a. Input-limited, LO is FULL

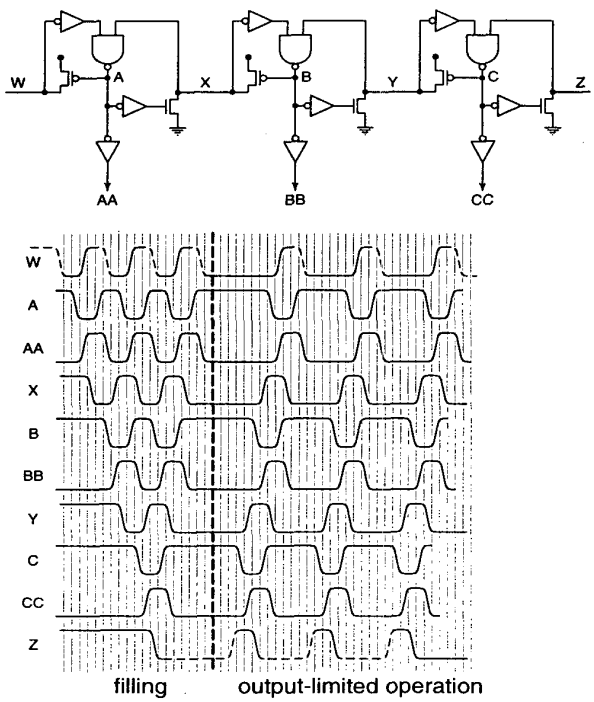

Figure 3b. output-limited, LO is FULL

4. Timing

Careful control of transistor widths, as described in the companion paper [7], makes all gate-delays nearly identical. Indeed, because we give each transistor a width proportional to its load, all logic gates operate not only with nearly uniform delay but also with nearly uniform rise time. Thus the wave-forms seen on different wires have similar shape and differ only in phase, as seen in the idealized output of Figure 3a and 3b. Simulation output from SPICE looks remarkably like these idealized wave forms.

Now let us consider how two PATHs like those of Figure 1 drive the state conductor between them. After the first PATH uses transistor [d] to drive the state conductor LO, meaning FULL, the second PATH will take at least three gate-delays to drive the state conductor HI again using transistor [y]. Moreover, by the time the second PATH turns on transistor [y] to drive the state conductor HI, the first PATH will have turned off transistor [d] and thus will have ceased driving the state conductor LO.

Consider next what happens at maximum throughput, assuming that all gate delays match. At maximum throughput the gate signals on both N-type transistors [b & x] in each NAND stack are identical, because a fresh empty space and a fresh data item arrive at exactly the same time. Moreover, because the gate delays match, the input signal to N-type drive transistor [d] in one PATH matches the input to P-type drive transistor [y] in the next PATH. In effect transistor [d] in one PATH and transistor [y] in the next PATH behave like a single quasi-inverter driving the state conductor, even though they are separated in space.

At less than maximum throughput the input signals to the two N-type transistors [b & x] in the NAND stack differ. One becomes HI before the other, but which one goes HI first depends on whether the FIFO is source-starved and waiting for a data item or sink-starved and waiting for an empty space. Similarly, the quasi-inverter formed by transistor [d] in one PATH and transistor [y] in the next PATH gets two different input signals in this case, but the input to the P-type transistor [y] is never lower in voltage than the input to the N-type transistor [d].

The logical effort of this circuit is remarkably small. In the forward direction, ignoring branching effort, the logical effort is 2/3 for the NAND stack, and 1/3 for the N-type drive transistor, for a 2/9 product. In the reverse direction, again ignoring branching effort, the logical effort is 2/3 for the NAND and 2/3 for the P-type drive, for a 4/9 product. This remarkably low logical effort, reduced to *less than one* in both directions by use of self-resetting circuits,

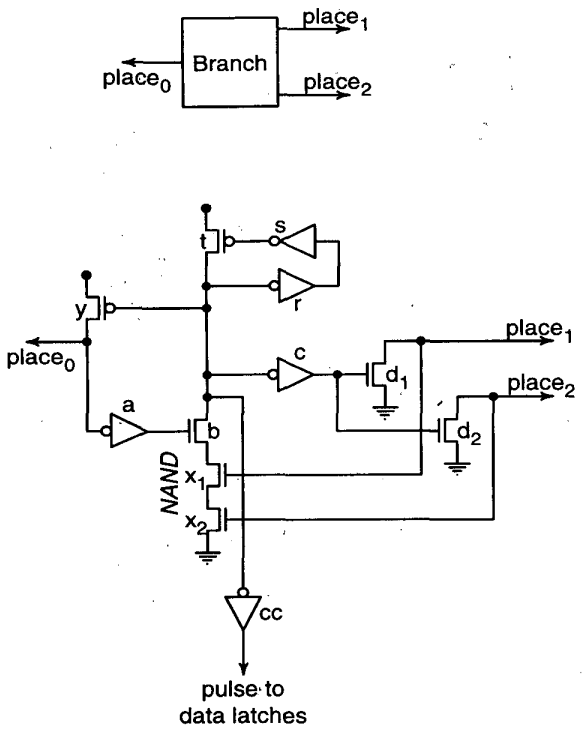

Figure 4. GasP with two successor places

contributes to the speed of GasP circuits. The very low logical effort of GasP circuits also allows them to drive wide data paths via a single amplifier, labeled [cc] in many of the figures. Avoiding multiple amplifier stages is important to retaining an adequate pulse width to activate the latches.

5. The family

Figures 4-7 show other GasP circuits. The unconditional Branch of Figure 4 acts when its predecessor PLACE [place0] is FULL and both successor PLACEs [place1 and place2] are EMPTY. It combines inputs from its two successor state conductors in a three-deep NAND stack [b x1 x2] and uses two N-type driver transistors [d1 and d2] to declare the two successor state conductors FULL. Similarly, the unconditional Join circuit, not illustrated, has separate predecessor state conductors, all of which must be LO, meaning FULL, before it takes action. The additional AND function can appear in the NAND stack, as in the Branch circuit, or as a NOR replacement for inverter [a] of Figure 1. Separate P-type driver transistors, like transistors [y1 and y2] in Figure 2, declare the separate predecessor state conductors EMPTY.

A companion paper [3] describes round-robin

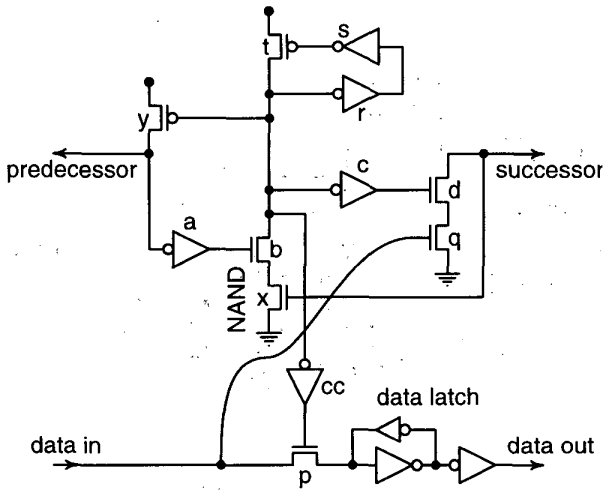

Figure 5. Data conditional GasP

configurations for branching and merging that use additional state conductors to indicate which of several parallel circuits, like those shown here, should be the next to fire.

Data conditional circuits require extra care. They must drive the successor state conductor LO, meaning FULL, only when suitable data are present. Note in Figure 1 that the N-type transistor [d] in the control circuit and the N-type pass gate transistor [p] in the data path are both one-inverter distant from the output of the NAND function. Thus [d] and [p] conduct concurrently. Provided the input data are in the right time relationship for the data latches, any input data bit is also in the proper time relationship with the control to condition a second N-type transistor [q] seen in Figure 5. This extra transistor prevents or permits driving the successor state conductor LO depending on the data input value. Notice that the data input to the circuit, rather than the data output, controls the action.

The circuit of Figure 6 is a form of GasP that uses a different encoding for its state conductors: HI means FULL and LO means EMPTY. This circuit has two N-type NAND stacks [c y] and [cc yy] with parallel inputs. One stack [cc yy] drives the predecessor state conductor directly while the other stack [c y] forms the NAND that activates the remaining functions. Because their inputs are connected in parallel, falling transitions of both stacks always coincide in time. Rising transitions may differ in time because they respond to other inputs.

Figure 6 also shows a P-type transistor driver [dd1] with a self-resetting loop to drive its reset transistor [e]. Such a self-resetting driver accommodates a wired-OR for input from another GasP circuit via transistor [dd2]. Because the duration of the latch pulse depends on the loop delay of the reset loop, its transistor widths must be carefully chosen.

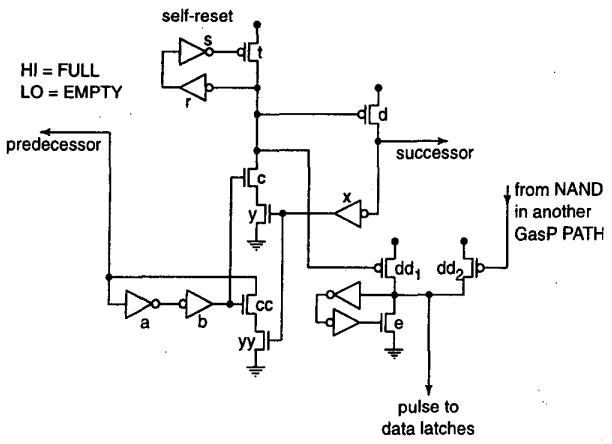

Figure 6. Another GasP form

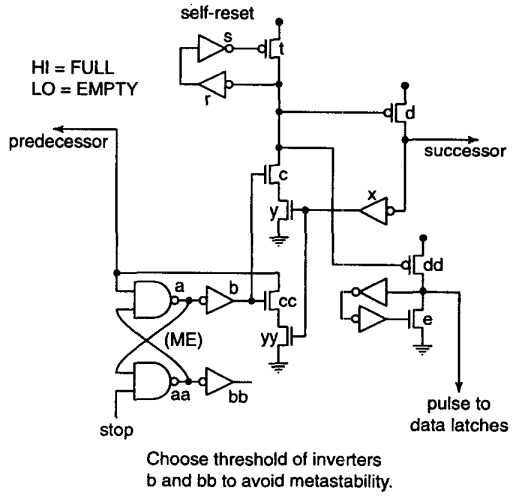

Figure 7. GasP with arbitration

Although its logical effort is slightly higher, the "HI means FULL" form of the GasP circuit seen in Figure 6 is preferable to the "LO means FULL" form of Figure 1 for three reasons. First, it offers two series stages of amplification [a & b] in the forward direction between the predecessor state conductor and the NAND stacks, either or both of which could perform logic. Second, between the successor state conductor and the NAND stacks it places the inverter [x] whose threshold provides extra noise immunity. And third, it has four levels of inversion [a b c dd1] rather than three levels in the forward direction between the predecessor state conductor and the latches. Moreover, in the reverse direction between the successor state conductor and the latches it has three levels of inversion [x y dd1] rather than two. These extra levels of inversion provide enough electrical amplification to drive even the heavy loads imposed by the many latches in wide data paths.

Perhaps most important, two series stages of amplification, [a & b] in the circuit of Figure 6 accommodate arbitration, as shown in Figure 7. Here, a Mutual Exclusion (ME) element [a & aa] and its metastability guard [b & bb] replace the two series amplifiers. This circuit provides a clean choice as to whether the NAND stack will or will not fire, even if the external *stop* input arrives at an unfortunate moment. We use this circuit as a "proper stopper" to interrupt the flow of data without damaging either the existence of, or the value carried by, a data item. A similar Demand-Join circuit gives contending inputs access to a common output.

6. Test chips

Our group at Sun Microsystems has now built several test chips using GasP circuits. One, called "First GasP," demonstrates FIFO rings and the data conditional forwarding shown in Figure 5. Another, called "Vanilla," measures the behavior of GasP circuits near maximum throughput. It includes three variants of GasP called 4/2, 4/4 and 6/4: the two numbers describe the number of gate-delays of forward and reverse latency. The circuits of Figures 1 - 7 are all of the 4/2 GasP form. The 4/4 GasP form adds two extra inverters in the reverse direction to make the circuit symmetric. The 6/4 GasP form adds two extra inverters in both the forward and reverse direction, reducing the care required in picking transistor widths. It is a more conservative design than shown in this paper and operates at the speed of a five-inverter ring oscillator.

Another GasP test chip is the "Square FIFO" described in [7]. It uses many circuits like that of Figure 2 to scatter entries to a number of parallel FIFOs and, later on, to gather them up again in sequence. It is built with ordinary NAND gates, as seen in Figure 2, rather than the self resetting forms. When the circuit drives the state conductors to their new state, they reset the NAND.

We have also built and tested two more elaborate chips. One called FLEETzero is reported separately in [1]. Another elaborate test chip, called "t35," is under test as this is written. The t35 chip includes alternating Join circuits, unconditional Join circuits, a data-conditional switch, as in Figure 5, and proper stoppers, as in Figure 7.

The t35 chip detects mismatched values in two high-speed 35-bit data streams, but after combining two such streams and computing error bits, its data path reaches a width of 105 bits, demonstrating the ability of GasP control circuits to drive large loads. The t35 control circuits operate as expected, but there remain some difficulties with surrounding logic. We expect to report full operation of the chip in the near future.

GasP circuits have proven remarkably resistant to changes in power supply voltage. As reported in [1] the "cargo rings" on the FLEETzero chip run correctly with power supply voltage anywhere between 1.2 and 4.8 volts with a nominal voltage of 3.3 volts. Other test chips experience similar latitude.

7. Conclusion

GasP circuits reduce asynchronous pipeline control to its minimal form. They involve an AND function and a mechanism for changing the state of predecessor and successor stages. Because they are pulse circuits they enjoy very low logical effort, reduced still further by the use of self-resetting logic. In addition to providing high speed, this very low logical effort obviates the need for additional control signal amplification, even in systems with wide data paths.

The GasP family is designed so that each stage operates at the speed of a three-inverter ring oscillator. Test chips in 0.35 micron technology exhibit throughputs in excess of 1.5 giga data items per second (GDI/s). GasP circuits simulated in a 0.18μ technology easily achieve the throughputs reported in [6].

GasP circuits suffer from dependence on time, albeit only in small local areas. Careful balance of transistor widths to match delays is crucial to making GasP circuits work. Devotees of delay insensitivity may claim that this denies the whole point of asynchronous design. We feel, however, that GasP combines the best of both the synchronous and asynchronous worlds. Circuits designed to operate in a fixed known time are logically simpler and faster than circuits that check completion every cycle; through *faith* we gain speed. GasP circuits are based on the faith that circuits designed to operate in a known time will do so, faith bolstered by using logical effort as the basis for design. We prefer to reserve asynchronous *measurement* of completion for higher level functions such as sequencing complex operations, conjunction of data in complex pipeline networks, dealing with metastability in arbitration, and waiting for the arrival of fresh data or empty space in long-distance communication.

8. References

[1] W. S. Coates, J.K. Lexau, I. W. Jones, S. M. Fairbanks, and Ivan E. Sutherland, "FLEETzero: An Asynchronous Switching Experiment," *Proc. of the Seventh International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2001.

[2] W. S. Coates, J.K. Lexau, I. W. Jones, S. M. Fairbanks, and Ivan E. Sutherland, "FLEETzero: An Asynchronous Switching Experiment," *Proc. of the Seventh International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2001.

[3] J. Ebergen, "Squaring the FIFO in GasP," *Proc. of the Seventh International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2001.

[4] C. E. Molnar, I. W. Jones, W. S. Coates, and J. K. Lexau, "A FIFO Ring Performance Experiment," *Proc. of the Third International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 279-289, April 1997.

[5] C.E. Molnar, I.W. Jones, W. S. Coates, Lexau, S. M. Fairbanks and I. E. Sutherland, "Two FIFO Ring Performance Experiments," *Proceedings of the IEEE,* No. 2 Vol. 87 Feb. 1999, February 1999.

[6] S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immediato, and K. Jenkins, "Asynchronous Interlocked Pipelined CMOS Circuits Operating at 3.3-4.5 GHz," *Proc. of the IEEE International Solid-State Circuits Conference,* 2000.

[7] I. E. Sutherland and J. K. Lexau, "Designing Fast Asynchronous Circuits," *Proc. of the Seventh International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2001.

[8] I.E. Sutherland, "Micropipelines," *Communications of the ACM*, Volume 32, No.6, pp. 720-738, June 1989.

[9] I. E.Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufmann Publishers, Inc., 1999.

[10] K. van Berkel and Bink, "Single-Track Handshaking Signaling with Application to Micropipelines and Handshake Circuits," *Proc. of the Second International Symposium on Advanced Research in Asynchronous Circuits and Systems*. 1996.