

# An Asynchronous High-Throughput Control Circuit For Proximity Communication

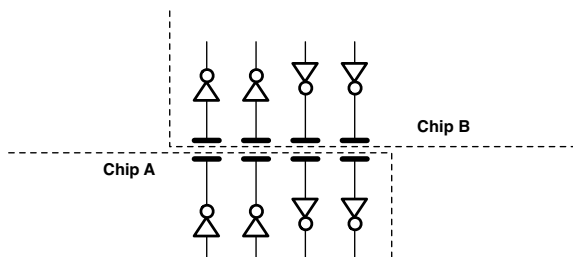
Jo Ebergen, Alex Chow, Bill Coates, Justin Schauer, David Hopkins  
Sun Microsystems Laboratories  
16 Network Circle, Menlo Park CA 94025, USA  
jo.ebergen@sun.com

## Abstract

We describe an asynchronous control circuit for inter-chip communication that enables high throughput proximity communication. The circuit has been fabricated in TSMC 180nm technology and operates over a wide range of coupling capacitances between the chips and over a wide range of supply voltages. At the nominal voltage of 1.8 V and a coupling capacitance of 40fF, this control circuit enables a throughput of 3 Giga token items per second for the data path.

## 1 Introduction

In [1] Drost et al. proposed proximity communication as a way to achieve high-throughput communication between chips. In proximity communication, two chips communicate through capacitive coupling when put in very close proximity. Figure 1 illustrates the idea of proximity communication.



**Figure 1. Proximity communication: communication through capacitive coupling of two face-to-face chips**

In a large array of chips, where inter-chip communication takes place through proximity communication, the dis-

tribution of a high-speed global clock is extremely hard, if not impossible. For this reason we are exploring asynchronous control circuits that enable high-speed proximity communication. One of the challenges in asynchronous proximity communication is to design a control structure that can communicate data items at very high throughput with proper flow control and under large variations of coupling capacitances. Here, proper flow control means that the communication channel never overflows or underflows and that the channel never loses, creates, or corrupts any data items. Our throughput target was 1 data item every  $20\tau$ , i.e., one data item every 4 FO4 gate delays. The goal of this paper is to describe a control circuit that enables this high throughput.

In order to appreciate the challenges, we sketch some of the problems that must be overcome. First there is the problem that the normal duration of a request-acknowledge cycle of a fast asynchronous control is barely fast enough. Our fastest and most efficient asynchronous control, GasP [7] has a cycle time of 6 “gate delays,” where a gate delay may be as small as approximately  $3\tau$ , which yields a cycle time of  $18\tau$ . This cycle time is one of the fastest cycle times among asynchronous control circuits [2]. The cycle time of  $18\tau$  would suffice, if not for the additional constraint that each cycle must include two chip-to-chip communications using proximity communication.

The second problem is minimizing the delay incurred in proximity communication. A chip-to-chip communication using proximity communication must have a large electrical gain and a large voltage gain. The large voltage gain is needed to increase the amplitude of the received signal. The large electrical gain is needed to charge the transmitter plates and to amplify the signal on the receiver chip. These large gains necessarily introduce extra delays. Furthermore, the signal amplitude at the receiver can vary widely, because the coupling capacitances between the plates can vary widely due to variations in the distance between the chips. Experimental data suggests that a typical one-way chip-to-chip communication costs about 3-4 FO4 gate delays.

These properties imply that the cycle time of a GasP control that includes two chip-to-chip communications is much longer than  $20\tau$ . For this reason we have to design a control structure that is significantly different from a simple GasP FIFO control in order to communicate a data item every  $20\tau$ .

A third problem specific to GasP is that the connection wires between GasP control modules are tri-state wires that can be driven from either end of the connection wire. Such bi-directional connection wires cannot be used for proximity communication, which is a uni-directional communication.

Our basic idea consists of combining three GasP FIFOs with special communication channels. Each GasP FIFO has its own control and data path on the sending and receiving chip. Between the chips, there are three special control paths, but only one shared data path. The three FIFOs operate in parallel, and a round-robin ring ensures that one FIFO at a time can send a data item on the shared data path. Each GasP FIFO has a cycle time of about  $60\tau$ , but the control structures operate concurrently such that the aggregate throughput of the shared data path is one data item every  $20\tau$  gate delays.

Related work on high-throughput communication between chips by means of capacitive coupling appears in [5, 6]. In [5], the authors present a synchronous implementation in 350nm technology that achieves a throughput of 1.27Gb/s with a power consumption of 3mW per data pin for the transmitter and receiver in a special data path of three bits. In [6], the authors present a synchronous implementation in 180nm technology for chip-to-chip communication through two 150fF coupling capacitances surrounding a 15cm 50ohm PCB trace and clock recovery at the receiver. Their maximum data rate is 3Gb/s with a power consumption of 15mW per data pin for only the transmitter and receiver in a one-bit data path. The supporting circuitry consumes an additional 110mW. No asynchronous control circuits are discussed in these publications. We present a control circuit for a fictitious 72-bit data path, where the coupling capacitance in the control path can vary between 17fF and 104fF and which achieves a maximum throughput of 3.28Gigatokens/s for a coupling capacitance of 104fF with a worst-case power consumption of 3mW per bit. Other related work on high-throughput communication over long wires with an asynchronous control circuit appears in [4]. We extend the techniques presented in [4] to proximity communication and demonstrate results from a fabricated chip.

## 2 An Implementation

To achieve a throughput of one data item every four gate delays, we created a circuit similar to Figure 2. There are

three identical control structures connected to each other by means of a round-robin ring. Each control structure has a sending FIFO, a receiving FIFO, and two communication channels in between. The control of each sending FIFO consists of three GasP modules. The last GasP module in each sending FIFO is part of the round-robin ring. A short explanation of GasP modules and their notation appears in the Appendix. A more detailed implementation of the GasP modules involved in the communication channel appears later.

All three GasP control structures operate concurrently and share a common inter-chip data path. The idea is that each sending FIFO controls the movements of data items along its own data path. The last GasP stage in each sending FIFO launches one data item at a time in a data path shared by all three FIFOs. The round-robin ring ensures that one FIFO at a time launches a data item in the shared data path. The launches are separated by at least the delay between two GasP modules in the round-robin ring, which is four FO4 gate delays or  $20\tau$ .

At the other end of the communication channel, data items are captured in latches in the receiving FIFOs some fixed delay later. This fixed delay depends on the delay in the communication channel and the GasP modules. The fixed delay must be controllable such that the correct data item will be latched within the time window where the appropriate data is valid.

We have chosen to combine three GasP FIFOs for our control structure, because we expected that each FIFO can operate at a cycle time of approximately  $60\tau$  and three such FIFOs can yield an aggregate throughput of one data item every  $20\tau$ . The cycle time of each FIFO is limited by the chip-to-chip cycle, which is the cycle containing two channels for chip-to-chip communication and three GasP modules. This cycle appears in Figure 3(a).

Although we have shown a data path on our circuit diagrams, in our first chip implementation we have chosen to implement only the control path with comparable data path loads on a single chip. The reason is that we wanted to keep the chip experiment simple and study the behavior of our control circuit under variations of coupling capacitance and supply voltage.

## 3 Complementary Signaling

The communication channels that implement the proximity communication cannot use the bi-directional tri-state wires that are commonly used between GasP modules, because of the presence of the series capacitor. In our implementation we use two-phase complementary signaling on two wires, where the two wires have complementary values. Each connection that uses two-phase complementary signaling is labeled with a 2 in the figures. Figure 4

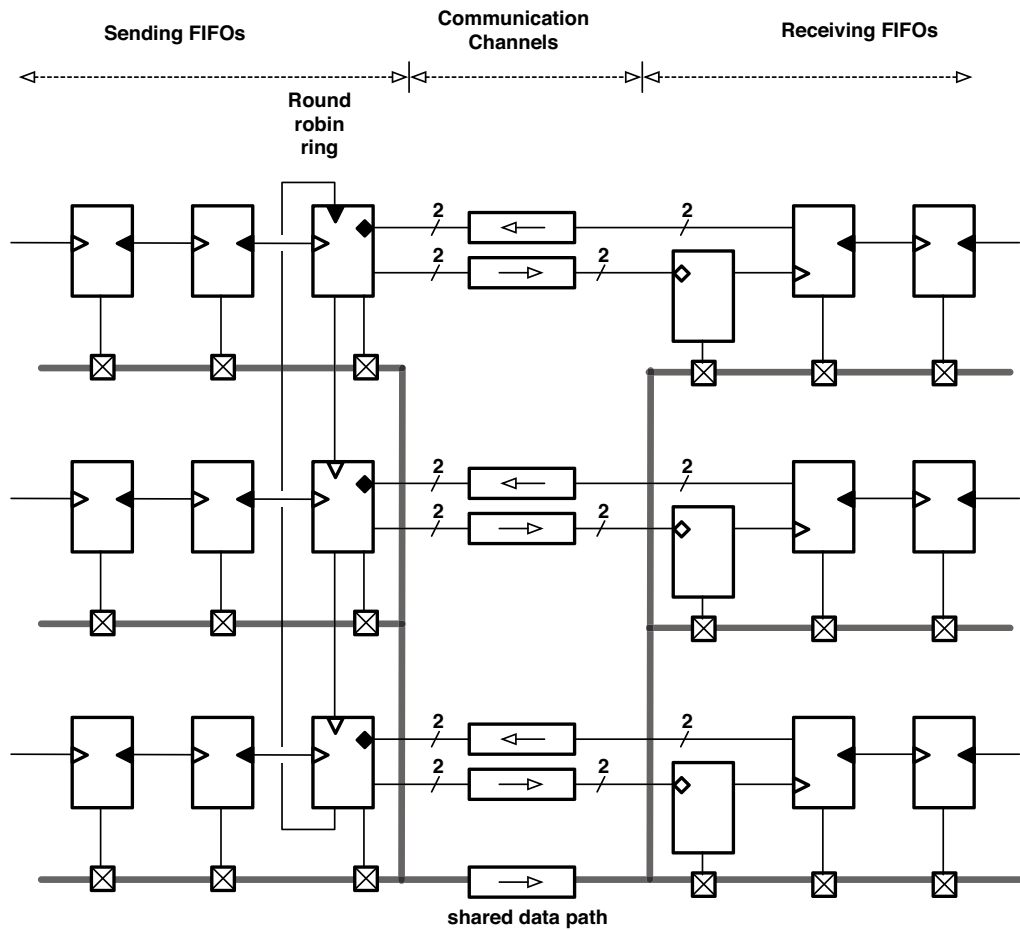
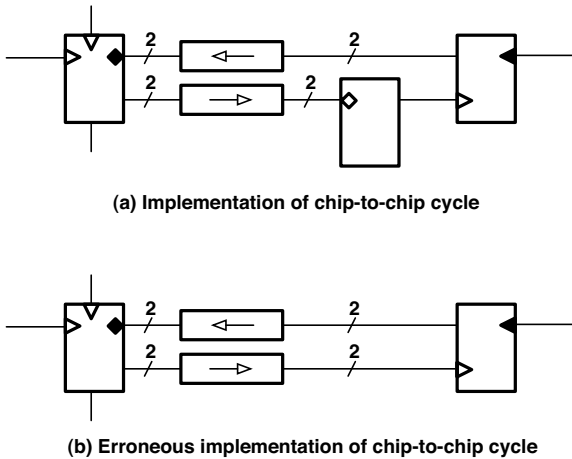
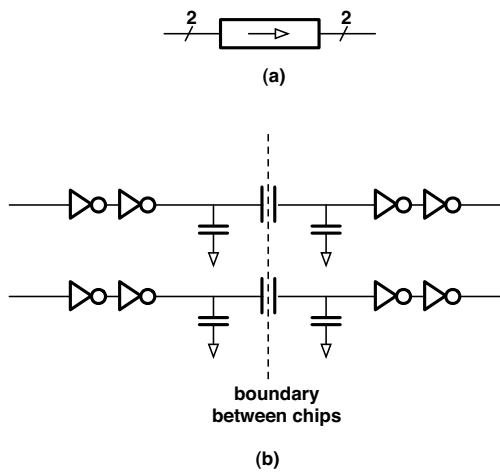


Figure 2. The control and data path



**Figure 3. The cycle containing the communication channels**

shows a simplified implementation of the communication channel. For each signal sent, both wires change state only



**Figure 4. (a) The communication channel and (b) a simplified implementation**

once and at the same time. The reasons for using complementary signaling are greater noise immunity and the presence of a return path for the current. The reason for using two-phase signaling, as opposed to four-phase signaling, is that each wire transitions only once with each communication, whereas in four-phase signaling each communication requires two transitions per wire. The combination of complementary signaling and two-phase signaling also simpli-

fies some of the logic in the circuit and avoids using expensive gates like XOR gates.

In Figure 4(b), the first inverters on the receiving chip are simple continuous receivers [3]. Instead of inverters, one can also use differential amplifiers on the sender and receiver chips. The implementation would change somewhat and may yield even better noise immunity.

#### 4 Control Flow

The control structures have special precautions to ensure proper flow control and to guarantee that no data items are lost or become corrupted. Consider the obvious, but erroneous, implementation in Figure 3(b) for each of the three chip-to-chip cycles. In the case that the three receiving FIFOs are full, each of the three transmitting FIFOs can send a data item on the shared data path, one at a time. Because no receiving FIFO can accept another data item, all three data items will interfere on the shared data path. Thus, the first data item will be overwritten by the second data item, and the second data item in turn will be overwritten by the third data item.

In contrast to Figure 3(b), the chip-to-chip cycle in Figure 3(a) contains an extra GasP module, and thus has an extra stage for storing data items. By means of this configuration we have included the first stage of the receiving FIFO in the chip-to-chip cycle. Thus, any data item sent on the communication channel can and will be captured in the first stage of the receiving FIFO. Because all stages follow the simple semantics of an asynchronous ripple FIFO, this control structure never causes over or underflow, nor does this control structure lose or corrupt any data items. A control structure using the cycle in Figure 3(b), however, may corrupt data items under certain circumstances. Ho et al. use a solution similar to Figure 3(b) for a dual control circuit in [4].

#### 5 Implementation Of Chip-To-Chip Cycle

We have slightly abused the notation of GasP modules in Figure 2. In particular, we failed to show how we can implement the GasP modules that have complementary wires as inputs or outputs. Figure 5 shows the implementation of the chip-to-chip cycle in Figure 3(a).

The left GasP module of Figure 3(a) is implemented by two GasP modules in Figure 5. We also show the round-robin ring connected to these two GasP modules by means of dashed lines. The round-robin ring ensures that the two GasP modules fire alternately. The right GasP module and middle GasP module in Figure 3(a) are also implemented by two GasP modules in Figure 5.

In Figure 5 a GasP module with two outputs labeled 0 and 1 means that upon firing the GasP module sets the out-

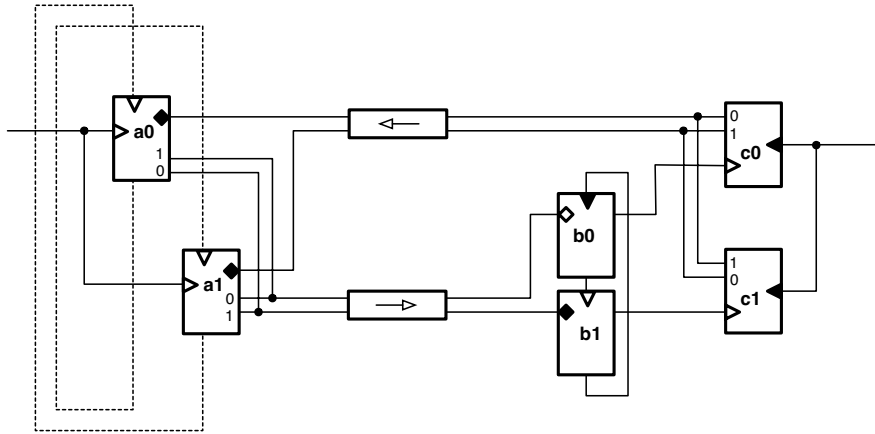


Figure 5. The implementation of the chip-to-chip cycle in Figure 3(a)

put labeled 0 to 0 and simultaneously the output labeled 1 to 1. A more detailed implementation of these GasP modules appears in the Appendix.

Following the firings of the GasP modules in the figure and assuming that module *a0* fires first when enabled by the round-robin ring, we obtain the following firing sequence

$$a0 \rightarrow b0 \rightarrow c0 \rightarrow a1 \rightarrow b1 \rightarrow c1$$

which repeats indefinitely.

## 6 The Channel

Figure 6 shows a simple implementation of a communication channel. The implementation consists of two signal paths and the estimated capacitances on the signal paths. For a specific 180nm chip, we estimate the coupling capacitance between plates of size  $120\mu$  by  $120\mu$  and a separation of  $2.5\mu$  (in air) to be 40fF. We obtained the estimated coupling capacitances by means of 3D field solver simulations. The capacitances before and after the plates are parasitic capacitances and we estimate those capacitances to be 200fF and 120fF respectively. For different channel implementations, the capacitances may vary from the estimated capacitances.

In chip-to-chip communication, the amount of coupling capacitance is strongly dependent on the distance between the two chips. One of the goals of our test chip is to vary the coupling capacitance to investigate the influence of the size of these capacitances on the latency and robustness of the communication channel.

Each line consists of two drivers before the coupling plates and two drivers after the coupling plates. With proper sizing, the two inverters before the coupling plates drive

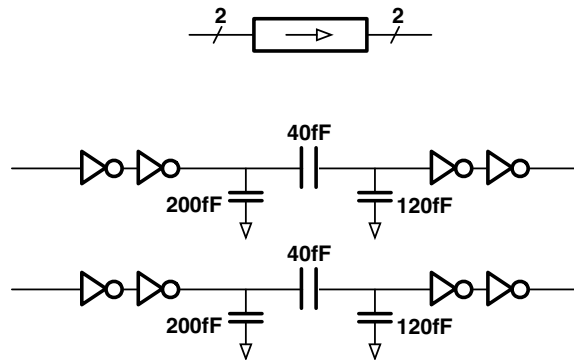
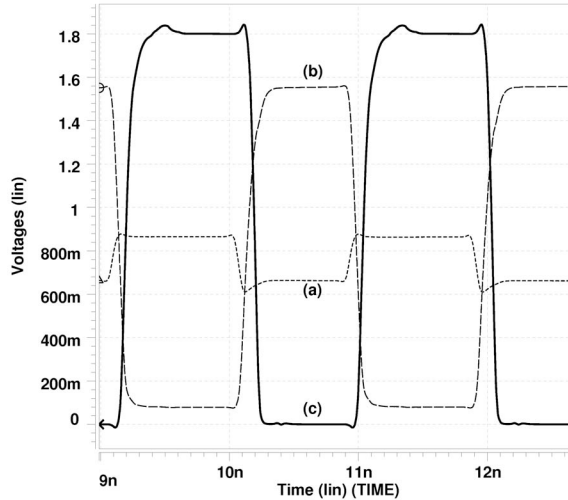


Figure 6. The implementation of the communication channel

the large parasitic and coupling capacitance with little delay, and the two inverters after the coupling plates amplify a small signal swing to a full signal swing with little delay.

Figure 7 shows the waveforms obtained from a SPICE simulation of the signals just after the plates, just after the first inverter, and just after the second inverter. Our bias voltages in the continuous receivers are 659mV and 889mV at 1.8V supply voltage. The receiver inverters contain weak feedback inverters (not shown), which also provide some offset compensation.

Notice that a small signal swing of 200mV has to be amplified to a full-swing signal of 1.8V as well as provide enough drive strength to drive the GasP modules on the receiving chip. Choosing the right sizes for the inverters in the communication channels and the GasP modules so as to obtain an optimal cycle time turned out to be a difficult



**Figure 7. Some waveforms from SPICE simulations of the signal (a) just after the coupling capacitance, (b) just after the first inverter, (c) just after the second inverter**

problem. While the theory of Logical Effort [8] provides an elegant way to calculate transistor sizes for minimizing the path delay in a simple acyclic path with only capacitance amplification, there is no such theory for minimizing cyclic path delays with both capacitance and voltage amplification. We have not been able to find a nice closed-form solution.

## 7 Test Chip and Results

Our test chip, qLite, contained the circuit described in Figure 2, but without the data path latches. For this particular test chip we wanted to study the behavior of the control path in isolation under variations in coupling capacitances and voltage variations. The problems related to the combination of a control path and data path will be studied on a different chip.

On our test chip, the source of each sending FIFO and the sink of each receiving FIFO can be controlled by off-chip signals. The off-chip signals control when each source generates a new control token and when each sink deletes a token from the receiving FIFO. Tokens can be generated continuously or gated one at a time as dictated by an external clock. The same applies for deleting control tokens at the sink. You can set the mode of operation by means of a scan chain, which can also observe the state of each FIFO

stage.

Normally an experiment with capacitive coupling across chips consists of two chips, where each chip contains one half of the control circuit. In order to avoid the complications with the alignment of two chips, we put the complete control circuit on one chip, where the capacitive coupling occurs between plates on metal level 5 and metal level 6. We implemented the variable capacitances for the coupling plates by means of arrays of micro-plates, where the connection of each micro-plate to the communication signal can be switched on or off. Switching a micro plate on or off changes not only the plate capacitance, but also the parasitic capacitance. The variable capacitors allow us to find out how the variation in parasitic capacitances and plate capacitances influence the chip-to-chip cycle times.

A sufficient timing condition for correct operation is that the gates in all GasP modules have approximately equal delay. We sized the transistors to satisfy this condition using a simple sizing algorithm [2]. Apart from this timing condition, there are no other timing conditions. In particular, the channels may have arbitrary delays.

The qLite chip was fabricated through MOSIS in a 180nm, 1.8V, CMOS TSMC technology. The total area for one circuit experiment is approximately 0.5 mm<sup>2</sup>. We tested the chip under many different configurations and used counters to keep track of the number of tokens stored in the sending FIFOs and the number of tokens stored in the receiving FIFOs. Thus we were able to calculate if any items were created or deleted during the emulated chip-to-chip communication. During testing, we noticed a problem with the counters. The counters were consistently off by a few items, independent of how many tokens were sent. We believe that this problem was due to the property that the FIFOs were too fast for the counters during a very small initial period. This counter problem has also plagued us on other chips. Otherwise the chip performed without any errors under many different operating modes of the sources and sinks.

Figure 8 shows the measured throughput versus estimated coupling capacitance of the circuit at three different supply voltages. Notice that above a coupling capacitance of about 40fF the throughput remains fairly constant. Below 40fF, the throughput drops off rather quickly. Figure 9 shows the measured throughput versus supply voltage of the circuit for three different estimated coupling capacitances. Figure 10 shows a schmoop plot of the circuit for different supply voltages and different coupling capacitances and fixed bias voltages. (The optimal bias voltages may be different for different supply voltages.) The white squares indicate the combinations of coupling capacitance and supply voltage for which correct functioning was observed. The black squares indicate combinations for which the chip malfunctioned. The grey squares indicate combi-

nations for which correct functioning was observed, albeit very slowly [10's of MHz].

Power consumption is mostly determined by the capacitances, the sizes of continuous receivers which consume static power, and the throughput of the control circuit. We designed the control path such that it could drive a data path of 72 bits wide through a single amplifier. At 3.28 Giga-tokens-per-second throughput and a worst-case coupling capacitance of 104fF, the total power consumption for the sending FIFOs, receiving FIFOs, amplifiers, counters, and communication channels is 216mW, or 3mW per bit. The dynamic power consumption is about 2.3mW per bit.

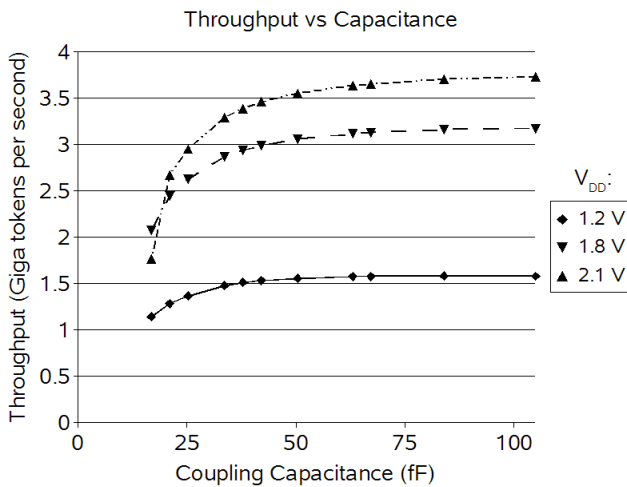


Figure 8. A graph of the throughput as a function of the variable capacitance

### 8 Concluding Remarks

We demonstrated a control circuit for achieving high throughput for proximity communication at a small cost of three chip-to-chip control paths for one shared chip-to-chip data path. Let us summarize the advantages and disadvantages of the chip-to-chip structure of Figure 2. The most important advantage is the high throughput of one data item every  $20\tau$ . A second advantage is that the control circuit was robust under coupling capacitance variations and supply voltage variations. In particular we were pleased to see that the implementation worked at such low supply voltages.

A possible disadvantage is that the data items must be launched in a round-robin fashion. Although the round-robin scheme is a simple scheme, there is a problem if we want to communicate data items in a different order. In such a case, data items may be held up, possibly indefinitely.

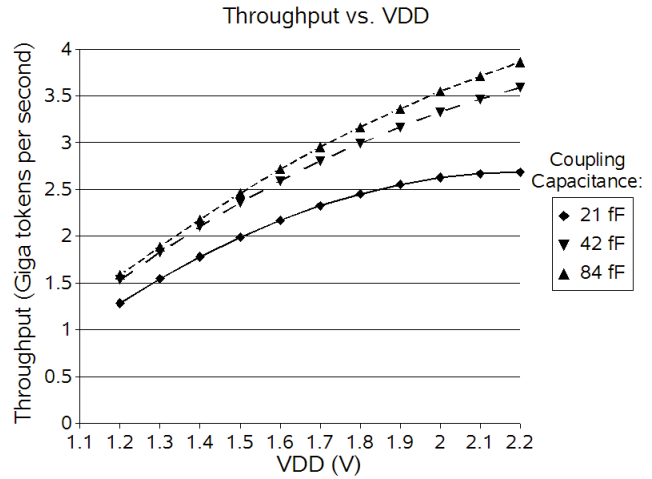


Figure 9. A graph of the throughput as a function of the variable supply voltage

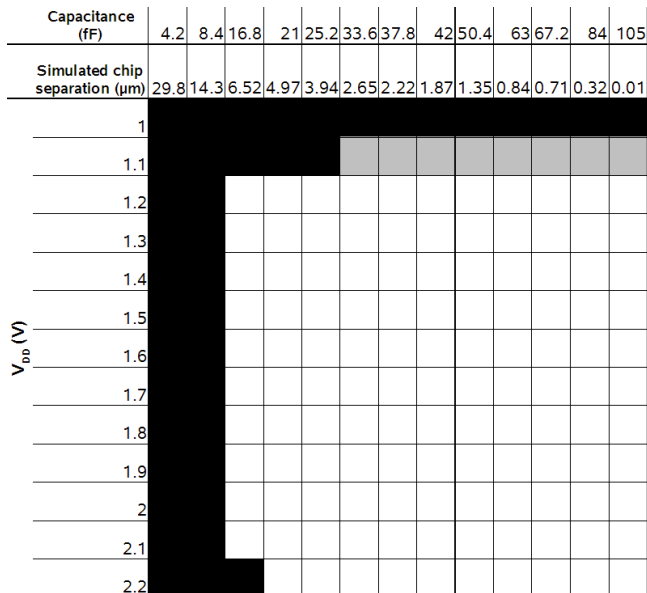
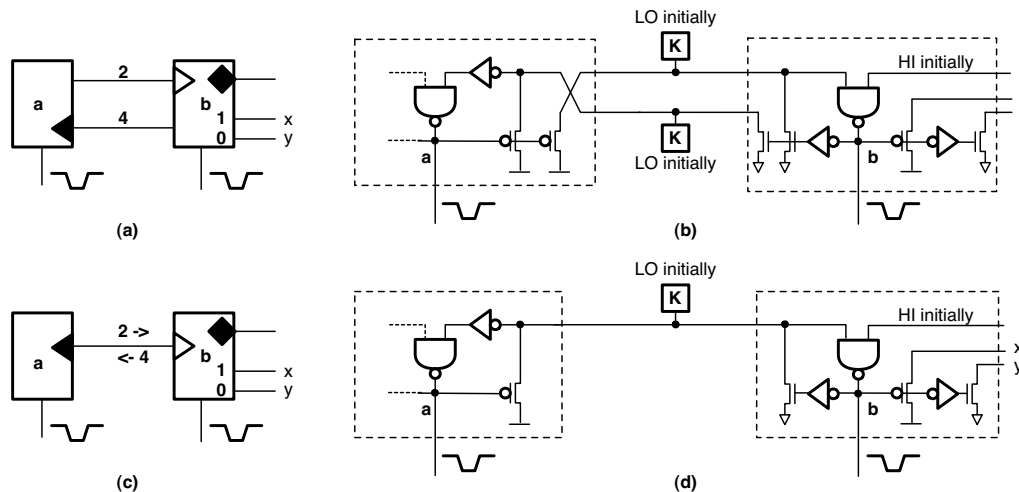


Figure 10. A schmoo plot of the qLite chip as a function of the coupling capacitance and supply voltage



**Figure 11. (a) A connection of 2-4 GasP modules; (b) Implementation; (c) Optimized connection between two GasP modules; (d) Implementation of (c)**

Demonstrating a high-throughput control for proximity communication, which is robust under variations in coupling capacitance and supply voltage, is only one step towards a complete solution for proximity communication. Demonstrating high-throughput communication for a combined control and data path is the next challenge!

## 9 Acknowledgements

We thank Mark Greenstreet for many insightful remarks on a different control circuit that led to the control circuit in this paper. Acknowledgements are also due to Robert Drost, Scott Fairbanks, Ron Ho, Russell Kao, Jon Lexau, Tarik Ono and Ivan Sutherland who helped during the various stages of the chip design. This project was sponsored as part of the DARPA HPCS program.

## References

- [1] R. J. Drost, R. D. Hopkins, R. Ho, and I. E. Sutherland. Proximity communication. *IEEE Journal of Solid-State Circuits*, 39:1529 – 1535, Sept. 2004.
- [2] J. Ebergen, P. Cunningham, and J. Gainsley. Transistor sizing with logical effort: How to control the speed and energy consumption of a circuit. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 51–61. IEEE Computer Society Press, 2004.
- [3] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer. *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, 2001.
- [4] R. Ho, J. Gainsley, and R. Drost. Long wires and asynchronous control. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 240–249. IEEE Computer Society Press, Apr. 2004.
- [5] K. Kanda, D. D. Antono, K. Ishida, H. Kawaguchi, T. Kuroda, and T. Sakurai. 1.27Gb/s/Pin 3mW/Pin Wireless superconnect (wsc) interface scheme. In *International Solid State Circuits Conference*, pages 186–187, Feb. 2003.
- [6] L. Luo, J. M. Wilson, S. E. Mick, J. Xu, L. Zhang, and P. Franzon. 3 Gbps AC coupled chip-to-chip communication using a low-swing pulse receiver. In *International Solid State Circuits Conference*, pages 522–523, Feb. 2005.
- [7] I. Sutherland and S. Fairbanks. GasP: A minimal FIFO control. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 46–53. IEEE Computer Society Press, Mar. 2001.
- [8] I. Sutherland, B. Sproull, and D. Harris. *Logical Effort, Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers, 1999.

## 10 Appendix: GasP

This section contains a brief introduction to GasP modules and their implementation. Figure 11 shows a connection of GasP modules and an implementation of that connection using the family of 2-4 GasP modules. We assume that all gates, including pull-up and pull-down stacks of transistors, have equal delay, achieved by a gate sizing algorithm [2]. Terminals of a GasP module with an arrowhead or diamond are called inputs; all other terminals are outputs. An input is active for a GasP module when the input value leads to a HI input for the NAND gate. Initially ac-

tive inputs have a darkened arrowhead or diamond. When all inputs to the GasP module are active, the module fires. Upon firing, the NAND gate causes three things to happen. First, it makes the inputs with an arrowhead inactive, second, it makes the outputs active with respect to other GasP modules or sets the output to a specific value, and third it produces a falling pulse with a width of three gate delays. This falling pulse can be amplified with an inverter to drive a pulsed latch.

Note that there are two types of inputs: inputs with an arrowhead and inputs with a diamond. Inputs with an arrowhead are called self-resetting inputs, because a GasP module resets this input after firing. Inputs with a diamond are called non-self-resetting inputs, because these inputs are not reset by the GasP module upon firing.

Outputs labeled with a 1 are set high after firing; outputs labeled with a 0 are set low after firing. For example, outputs  $x$  and  $y$  are set high and low respectively after GasP module  $b$  fires.

When two GasP modules are connected, the minimum time separation between the firing of two GasP modules can be two or four gate delays. For example, the minimum time separation from the firing of module  $a$  to the successive firing of module  $b$  is 2 gate delays. The minimum time separation from the firing of module  $b$  to the successive firing of module  $a$  is 4 gate delays.

The connection wires between GasP modules are called state wires. One GasP module drives the state wire HI, a different GasP module drives the state wire LO. State wires are tri-state wires, and consequently each state wire has a keeper to keep the state. To avoid clutter, we usually omit keepers on state wires in GasP circuits.

In the case of the connection in Figure 11(a+b) between two GasP modules, we can perform a small optimization. The two state wires can be combined into one state wire, resulting in the schematic and implementation of Figure 11(c+d).

Although there are various sufficient timing conditions under which GasP modules operate properly, a simple sufficient timing condition is that all gates in all GasP modules have approximately equal delay. We satisfy the timing conditions by sizing each gate in a GasP module for equal delay [2].