University of Pennsylvania
ScholarlyCommons

Departmental Papers (BE) Department of Bioengineering

7-1-2004

A burst-mode word-serial address-event link--III: analysis and test results

Kwabena A. Boahen
University of Pennsylvania, boahen@seas.upenn.edu

Copyright 2004 IEEE. Reprinted from *IEEE Transactions on Circuits and Systems--I: Regular Papers*, Volume 51, Issue 7, July 2004, pages 1292-1300.
Publisher URL: http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=29094&puNumber=8919

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/be_papers/5
For more information, please contact repository@pobox.upenn.edu.

A Burst-Mode Word-Serial Address-Event Link—III: Analysis and Test Results

Kwabena A. Boahen

***Abstract*—We present results for a scalable multiple-access inter-chip link that communicates binary activity between two-dimensional arrays fabricated in deep submicrometer CMOS. Capacity scales with integration density because an entire row is read and written in parallel. Row activity is encoded in a burst: The row address followed by a column address for each active cell. We predict the distribution of burst lengths when transmission is initiated by active cells and access is arbitered using a two-level queuing model. Agreement with the experiment is excellent for loads over 50% but not for lighter loads, where our assumption that service time is exponentially distributed breaks down. We also quantify the throughput–latency tradeoff. The price of an n-fold increase in throughput is an n per N_{col} timing error in a cell's inter-event interval, where N_{col} is the number of cells per row. Links implemented in 0.6, 0.4, and 0.25 μm are compared; the highest burst-rate achieved was 27.8 M events/s.**

***Index Terms*—Asynchronous logic synthesis, event-driven communication, fair arbiter design, neuromorphic systems, parallel readout, pixel-level quantization.**

I. Performance Criteria

ALL-OR-NONE voltage transitions in two-dimensional (2-D) arrays may be communicated to other arrays using event-driven multiplexer-demultiplexer links. Event-driven access, whereby the transition, or event, is transmitted as soon as it occurs, has clear advantages over clock-driven access, whereby each cell is polled regularly. In particular, performance is better when activity is sparse (e.g., spatial or temporal filtering occurs) and timing is critical (e.g., time encodes analog information). Consequently, event-driven communication links have been explored for silicon retina [1]–[4] and silicon cochlea [5] chips. Interest in these links is increasing, driven by the growing trend of quantizing signals inside the array (e.g., active pixel sensors [6]–[8] and pulse-coded neural networks [9]).

Performance of event-driven communication links, or any system that communicates all-or-none signals for that matter, is measured by capacity, throughput, and latency [9]. **Capacity** is defined as the reciprocal of the minimum transmission time; this is the maximum rate at which events can be read, multiplexed, demultiplexed, and written. **Throughput** is defined as the usable fraction of capacity; the maximum rate is rarely sustainable in practice. **Latency** is defined as the mean delay; this wait time may be several transmission slots. Latency depends on the fraction of transmission slots that are filled; this fraction of the link capacity that is actually being used is called the **load**.

Link designers strive not only to maximize throughput, but to minimize latency as well. High throughput allows large numbers of event generators operating over a broad range of rates to be serviced. Whereas low latency preserves the timing of each individual event, capturing information over and beyond that carried by the generator's mean event rate [10], [11]. Throughput is optimized if collisions are prevented through arbitration; it is then limited only by the increase in latency with activity due to queuing [12]–[14]. To achieve a specified **timing error**, defined as the percentage error in a cell's interevent interval, throughput must be capped at a level somewhat less than 100% of the link's capacity.

We have recently developed an event-driven link that boosts capacity by reading (and writing) the state of an entire row of cells in parallel, as shown in Fig. 1 [15], [16]. Communication speed is not compromised when we convert from parallel to serial (and back) because we use devices much larger than those in the array. As capacity is boosted without sizing up devices inside the array, our design can better exploit the high integration densities deep submicrometer processes offer. However, the throughput attainable depends on the load, as the number of active cells read from (or written to) a row increases with the average event-rate. This fraction goes up both because the probability per unit time of an event occuring is higher and because more time is spent waiting when there are more requests.

In this paper, we analyze the tradeoff between latency and throughput in the parallel-read-write link and validate our analytical results using measurements from fabricated chips. We judge our design's success by comparing it to previous event-driven link designs, which read (and write) events from (and to) the array serially (reviewed in [14]). A mismatch in the level at which queuing occurs (entire rows versus single cells) leads to rather different tradeoffs. In particular, we address the questions: How much can you increase throughput by going parallel? What price do you pay in latency? Given your timing-error specification, is it worth it?.

The paper is divided into six sections. In Section II, we identify the timing parameters that determine the parallel-read-write link's performance and state the assumptions that underly our model. In Section III, we analyze the link's ability to increase its throughput with demand and quantify the longer latency incurred in doing so. In Section IV, we present capacity measurements for links fabricated in 0.6-, 0.4-, and 0.25-μm CMOS technology and compare their signaling rates. In Section V, we

Manuscript received January 3, 2002; revised November 2002. This work was supported in part by the Whitaker Foundation and in part by the National Science Foundation's LIS/KDI and CAREER programs under Grant ECS98-74463 and Grant ECS00-93851. This paper was recommended by Associate Editor G. Cauwenberghs.

The author is with the Department of Bioengineering, University of Pennsylvania, Philadelphia, PA 19104-6392 USA (e-mail: boahen@seas.upenn.edu).

Digital Object Identifier 10.1109/TCSI.2004.830701

1057-7122/04$20.00 © 2004 IEEE

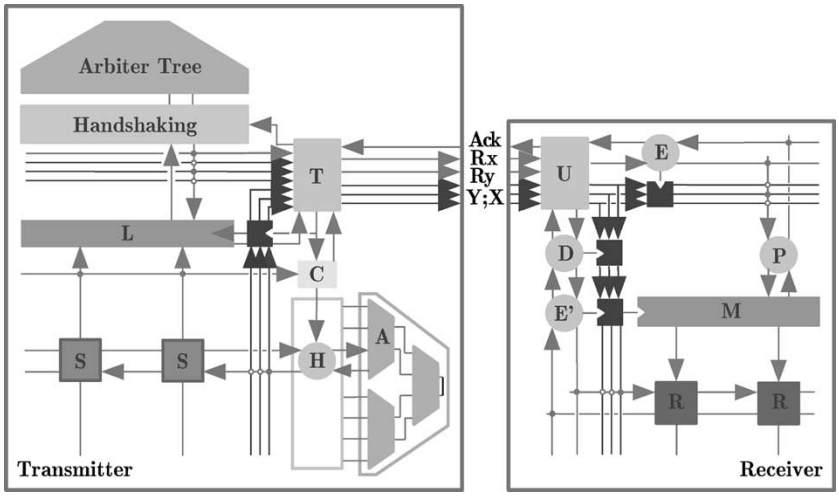


Fig. 1. Link architecture. *Transmitter*: An interface circuit (H) relays requests to the row arbiter (A) and permits that row to output its address and its events (S) when the arbiter acknowledges. The events' column addresses are generated similarly, after latching the row's state (L). A two-way multiplexer (T) sequentially outputs row (Y) and column (X) addresses using separate request lines (Ry,Rx); they share a single acknowledge line (Ack). Meanwhile, a controller (C) cycles the array to another row. *Receiver*: A two-way demultiplexer (U) directs row addresses to one latch (D) and column addresses to another (E). When the burst ends, the row's address and its decoded column addresses (P) are written to a second set of latches (E' and M). As the row address is decoded and the column data is written to that row (R), the next burst is received and its column addresses are decoded.

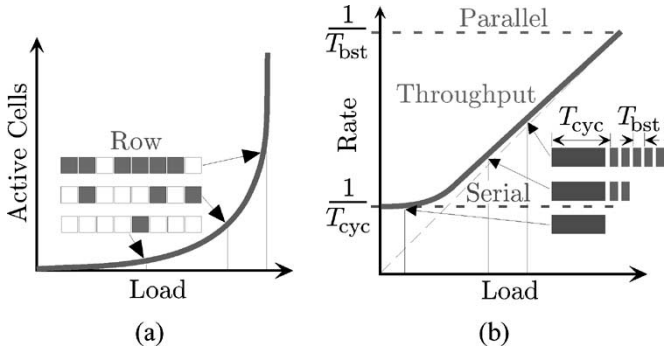


Fig. 2. Throughput on demand. (a) Number of active cells in a selected row increases with load. Boxes represent a row with eight cells; active cells are gray. The three rows are for three different loads. (b) Throughput increases with load (unity-slope line) as an increasing number of active cells are read in parallel. Boxes represent how long it takes to service each active cell. Again, the three rows are for three different loads.

present experimental measurements of the mean number of events read in parallel (i.e., burst length) at various load levels and compare them with our theoretical predictions. Section VI concludes the paper. The parallel-read-write link's transmitter and receiver designs are presented in companion papers [15], [16].

II. Timing Parameters

The parallel-read-write link's performance is determined by two timing parameters, $T_{\rm cyc}$ and $T_{\rm bst}$, as illustrated in Fig. 2. $T_{\rm cyc}$ is the time it takes to read (or write) an event from (or to) the array. We call $1/T_{\rm cyc}$ the **cycle rate**, because this is the rate at which we cycle between rows. $T_{\rm bst}$ is the time it takes to read (or write) an event from (or to) a latch on the array periphery. We call $1/T_{\rm bst}$ the **burst rate**, because this is the rate at which we transmit multiple events read from the same row. We presume that the transmitter and the receiver have the same values for $T_{\rm cyc}$ and $T_{\rm bst}$, and that they are pipelined such that the next event can be read while the previous one is being written. Both of these requirements are met by the architectures we proposed previously (see Fig. 1).

Users are interested not in how much you boost the capacity of the link but in how much you increase its absolute throughput. The parallel-read-write link's capacity is equal to the burst-rate, as this is the maximum rate at which it transmits events. Whereas the serial-read-write link's capacity is equal to the cycle-rate, as this is the maximum rate at which events can be read from (or written to) the array individually. Therefore, the parallel link has $b = (1/T_{\rm bst} - 1/T_{\rm cyc})/(1/T_{\rm cyc}) = T_{\rm cyc}/T_{\rm bst} - 1$ times more capacity than the serial link. The question is: Does this b-fold boost in capacity translate to a b-fold increase in absolute throughput? We relate the **throughput gain** r to the **boost factor** b in Section III by analyzing queuing in the parallel link. For this analysis, we make the simplifying assumption that event-generation and event-service are Poisson.

Event generation and event service may be assumed to be Poisson under most—but not all—conditions of interest. For event generation, the independent behavior that underlies Poisson statistics certainly applies to the array's cells when they are not responding to a stimulus. Independence holds for stimulus-driven activity as well when decorrelation (i.e., whitening) is performed to encode sensory information efficiently (e.g., silicon retinae [14]). For event service, Poisson statistics' exponential event interval distribution does not pertain for light loads, where the transmission time is more or less fixed at $T_{\rm cyc}$. Nevertheless, the exponential distribution is appropriate for heavy loads because burst lengths are geometrically distributed, as we show shortly.

III. Queuing Model

We have developed a two-level queuing model for the parallel-read-write link's transmitter. The model is shown in Fig. 3.

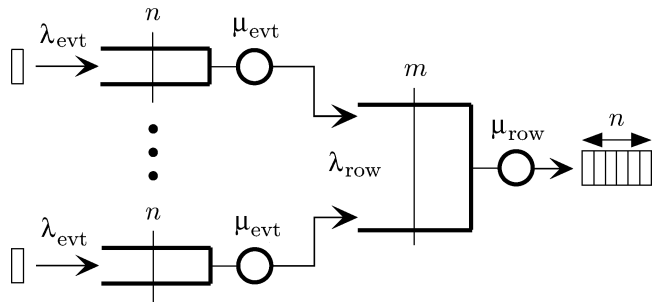


Fig. 3. Queuing model. Events occur at the same rate $\lambda_{\rm evt}$ in each of $N_{\rm row}$ rows; the overall rate at which rows make requests is $\lambda_{\rm row}$. Each row is serviced at the same rate $\mu_{\rm evt}$; the overall rate at which rows are serviced is $\mu_{\rm row}$. The number of rows waiting to be selected is m and the number of events a selected row has is n. These events are transmitted in a burst of length n.

Level 1 has as many queues as there are rows in the array. Each one gives the number of events n that a particular row has when it is serviced. Level 2 has a single queue; it gives the number of rows m waiting for service. The event count n depends on how often events are read from each row $\mu_{\rm evt}$, which in turn depends on the row count m. The row count m depends on how rapidly rows are transmitted $\mu_{\rm row}$, which, in turn, depends on the event count n. This circular behavior in analyzed in this section, keeping in mind that the average event-transmission time drops as n rises.

Under the Poisson assumption, the rate $\lambda_{\rm evt}$ at which events occur in a row and the rate $\mu_{\rm evt}$ at which they are read from a row give the probability of generating and servicing an event in that row, respectively. These probabilities are simply $\lambda_{\rm evt}\Delta t$ and $\mu_{\rm evt}\Delta t$ for a short time interval Δt. Similarly, $\lambda_{\rm row}\Delta t$ and $\mu_{\rm row}\Delta t$ give the probabilities of generating and servicing row requests in the array, respectively. Unlike $\lambda_{\rm evt}$ and $\mu_{\rm evt}$, which are in events per second, $\lambda_{\rm row}$ and $\mu_{\rm row}$ are in rows per second.

In equilibrium, the number of events in a row is obtained by equating the rates at which events are generated and serviced [17]. This count increases from n to $n+1$ with probability $p(n)\lambda_{\rm evt}\Delta t$, where $p(n)$ is the probability of being in state n, while it decreases from $n+1$ to n with probability $p(n+1)\mu_{\rm evt}\Delta t$. Equating $p(n+1)\mu_{\rm evt}$ to $p(n)\lambda_{\rm evt}$ and assuming that events are serviced faster than they are generated (i.e., $\lambda_{\rm evt} \le \mu_{\rm evt}$), we obtain

$$p(n) = (1-p)p^n, \qquad n = 0, 1, 2, \ldots \tag{1}$$

This is a geometric distribution with parameter $p = \lambda_{\rm evt}/\mu_{\rm evt} \le 1$. For intuition, think of p as the fraction of event-transmission slots that are filled at the level of a single row. Similarly, the number of row requests also is described by a geometric distribution, $q(m)$, with parameter $q = \lambda_{\rm row}/\mu_{\rm row} \le 1$. For intuition, think of q as the fraction of row-transmission slots that are filled at the level of the entire array.

To calculate p, we need to know the rate $\lambda_{\rm evt}$ at which events occur in a row and the rate $\mu_{\rm evt}$ at which any particular row is serviced. We can easily infer $\lambda_{\rm evt}$ from the number of cells the row has, $N_{\rm col}$, and the average cell activity. But $\mu_{\rm evt}$ is more involved because it depends not only on how many rows are serviced per second, $\mu_{\rm row}$, but also on how many rows request at the same time. To calculate $\mu_{\rm row}$, recall that the transmitter takes $T_{\rm cyc}$ seconds to send the first event in a burst and $T_{\rm bst}$ seconds to send each of the remaining ones [see Fig. 2(b)]. Therefore, we have

$$\mu_{\rm row} = \frac{1}{T_{\rm cyc} + (n_{\rm evt}-1)T_{\rm bst}} \tag{2}$$

where $n_{\rm evt}$ is the average number of events read from a row. Dividing $\mu_{\rm row}$ by $m_{\rm row}$, the average number of rows requesting, and multiplying by $n_{\rm evt}$, to convert from rows per second to events per second, yields

$$\mu_{\rm evt} = \left(\frac{n_{\rm evt}}{m_{\rm row}}\right)\mu_{\rm row}. \tag{3}$$

We are now left with the tasks of calculating $n_{\rm evt}$, the average number of events read from a row, and $m_{\rm row}$, the average number of row requests. To calculate $n_{\rm evt}$, we make use of the geometric distribution $p(n)$

$$n_{\rm evt} = \frac{\sum_{n=1}^{\infty} p(n)n}{\sum_{n=1}^{\infty} p(n)} = \frac{1}{1-p}. \tag{4}$$

We do not include $n=0$ because an empty row would not be read. To calculate $m_{\rm row}$, we make use of the geometric distribution $q(m)$

$$m_{\rm row} = \sum_{m=0}^{\infty} q(m)m = \frac{q}{1-q}. \tag{5}$$

We now have expressions for every variable we need to calculate p—but we still need to calculate q.

To calculate q, which is required to calculate $m_{\rm row}$, we need to know the total rate $\lambda_{\rm row}$ at which the rows make requests. We already know the total rate $\mu_{\rm row}$ at which they are serviced [see (2)]. We obtain $\lambda_{\rm row}$ by observing that, on average, a row makes a request every time it accumulates $n_{\rm evt}$ events, as that is the mean number of events read. Therefore, we divide the rate $\lambda_{\rm evt}$ at which events occur in each row by $n_{\rm evt}$ to obtain the rate at which a single row makes requests. Then, we multiply the result by $N_{\rm row}$, the number of rows, to obtain

$$\lambda_{\rm row} = \left(\frac{N_{\rm row}}{n_{\rm evt}}\right)\lambda_{\rm evt}. \tag{6}$$

We are finally ready to derive a system of simultaneous equations that can be solved for the geometric parameters q and p. For q, substituting the expression for $n_{\rm evt}$ given in (4) into (6) and (2) to obtain $\lambda_{\rm row}$ and $\mu_{\rm row}$, respectively, gives

$$q = \frac{(1-p)T_{\rm cyc} + pT_{\rm bst}}{T} \tag{7}$$

where $T = 1/N_{\rm row}\lambda_{\rm evt}$ is the inter-event interval for the entire array. For p, calculating $\lambda_{\rm evt}$ from (6), dividing the result by the expression for $\mu_{\rm evt}$ in (3), and substituting the expression for $m_{\rm row}$ in (5) gives

$$p = \frac{1}{N_{\rm row}}\frac{q^2}{1-q}. \tag{8}$$

We can solve these equations for p and q as a function of T, the inter-event interval, given $T_{\rm cyc}$, $T_{\rm bst}$ and $N_{\rm row}$. Instead of presenting these expressions, which are messy and uninsightful, we derive excellent approximations for light and heavy loads in the next section.

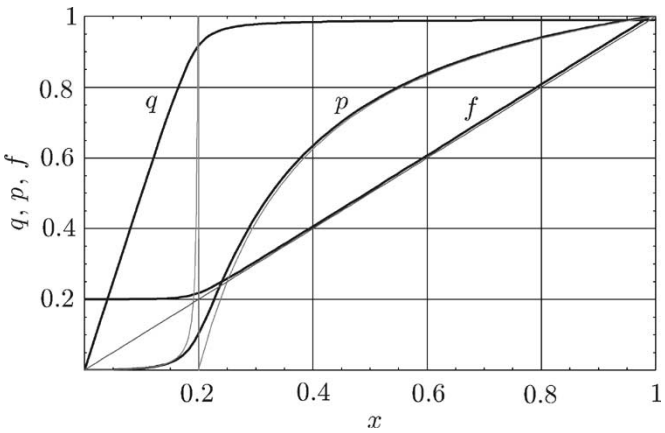


Fig. 4. Throughput versus load. Row load q, burst probability p, and attainable throughput f, versus link load x (unity-slope line). Load and throughput are expressed as a fraction of the link's burst-rate, which is five times higher than its cycle-rate. Light- and heavy-load approximations for p are plotted in hairlines. There are 100 rows.

A. Light and Heavy Loads

To find approximations for the geometric parameters, we draw on two insights from the preceding derivation. First, the fraction of events that are transmitted at the burst-rate (i.e., $(n_{\rm evt}-1)/n_{\rm evt}$) is equal to the fraction of event-transmission slots that are filled at the row level (p). This identity is stated by (4). Second, the fraction of attainable throughput that is used is equal to the fraction of row-transmission slots that are filled (q) at the array level. This identity is stated by (7), where attainable throughput is precisely defined as the reciprocal of the weighted average of $T_{\rm cyc}$ and $T_{\rm bst}$. Hence, we refer to p as the **burst probability** and to q as the **row load**, which is not to be confused with the link load defined earlier (see Section I).

Solutions for row load, burst probability, and attainable throughput are plotted as a function of the link load in Fig. 4, with $T_{\rm bst}/T_{\rm cyc}=0.2$ and $N_{\rm row}=100$. The row load (q) starts increasing early, even when the link load (x) is a small fraction of the cycle-rate (i.e., $x \ll 0.2$). By the time the cycle rate is exceeded, the row load is almost 100%, indicating that virtually all row-transmission slots are filled. At this point the burst probability (p) shoots up, shifting the transmission time from $T_{\rm cyc}$ to $T_{\rm bst}$. Hence, attainable throughput (f) starts to increase, a hair ahead of the load. The row load increases before the burst probability does because the 1:1 load-to-service ratio for rows is higher than the $1/N_{\rm row} : 1/m_{\rm row}$ load-to-service ratio for individual cells. Hence, the fraction of row-transmission slots filled at the array-level (q) is $N_{\rm row}/m_{\rm row}$ times higher than the fraction of event-transmission slots filled at the row level (p). Or $p=(m_{\rm row}/N_{\rm row})q$ [this identity also follows from (3) and (6)].

The insight that burst probability is almost zero when the load is light yields a light-load approximation for p. With $p \ll 1$, (7) simplifies to $q \approx T_{\rm cyc}/T$. Substituting this value into (8) yields

$$p \approx \frac{1}{N_{\rm row}} \frac{\left(\frac{T_{\rm cyc}}{T}\right)^2}{1-\frac{T_{\rm cyc}}{T}}. \tag{9}$$

This approximation is also plotted in Fig. 4. It is excellent for loads lower than $1/2T_{\rm cyc}$ (i.e., $x<0.1$) because $p<1/2N_{\rm row}$ for $T>2T_{\rm cyc}$, which keeps $q<1/2$. Thus, our light-load approximation is good until half the attainable throughput is used up.

The insight that row load is almost 100% when the load is heavy yields a heavy-load approximation for p. With $q \approx 1$, (7) gives

$$p \approx \frac{1-\frac{T}{T_{\rm cyc}}}{1-\frac{T_{\rm bst}}{T_{\rm cyc}}}. \tag{10}$$

This approximation is also plotted in Fig. 4. It is excellent for loads higher than $2/(T_{\rm cyc}+T_{\rm bst})$ (i.e., $x>0.33$) because $p>1/2$ for $T<(T_{\rm cyc}+T_{\rm bst})/2$, which keeps $q>1-1/(1+N_{\rm row}/2)$.

B. Wait Time

We now use our approximations for the geometric parameters to estimate the mean wait time. The wait time is given by the average amount of time required to transmit each event times the number of events queued. To calculate the number of events queued, we use (4) to obtain the expected number of events in each row $n_{\rm evt}$ and solve the identity $p=(m_{\rm row}/N_{\rm row})q$ to obtain the expected number of rows waiting, $m_{\rm row}$. Thus, we obtain

$$n_{\rm wait}=n_{\rm evt}m_{\rm row}=\frac{\frac{p}{q}}{1-p}N_{\rm row}. \tag{11}$$

To find a light-load estimate for the wait time, we substitute the appropriate approximations for q (i.e., $T_{\rm cyc}/T$) and p (9) into (11). After multiplying by the mean transmission time, which is close to $T_{\rm cyc}$ in this regime, we obtain

$$T_{\rm wait} \approx \frac{\frac{T_{\rm cyc}}{T}}{1-\frac{T_{\rm cyc}}{T}}T_{\rm cyc} \tag{12}$$

assuming $T_{\rm cyc}/T \ll N_{\rm row}$, which is very reasonable since $T_{\rm cyc}/T<1/2$ in the light-load regime.

To find a heavy-load estimate for the wait time, we substitute the appropriate approximations for q (i.e., $q \approx 1$) and p (10) into (11). Observing that the mean transmission time virtually equals the mean inter-event interval T in this regime, we obtain

$$T_{\rm wait} \approx Q\frac{\frac{T_{\rm bst}}{T}}{1-\frac{T_{\rm bst}}{T}}T \tag{13}$$

where Q is given by

$$Q=\frac{N_{\rm row}(T_{\rm cyc}-T)}{T_{\rm bst}}. \tag{14}$$

Q attains its maximum value of $N_{\rm row}$ times $b=T_{\rm cyc}/T_{\rm bst}-1$ when $T=T_{\rm bst}$.

How do these wait times compare with the serial-read-write link's? In the light-load regime [see (12)], the wait time is identical to that in a link with a fixed transmission time $T_{\rm cyc}$ [14], [17], hence the two links behave the same. We would not expect any different, as bursts are rare when the load is light. In the heavy-load regime [see (13)], the wait time is Q times longer than in a link with a fixed transmission time of $T_{\rm bst}$. And this queue-multiplier Q can be as large as the boost factor times the number of rows. The reason is that virtually all the rows are

waiting when the burst probability p is high (based on the identity $p = (m_{\rm row}/N_{\rm row})q$ with $q \approx 1$). And the number of events each row has is more or less equal to the boost factor, as that is the degree of parallelism required to achieve the boost. It is this escalation in the wait time that ultimately limits how much of the parallel-read-write link's capacity we can use, as we show next.

C. Throughput Gain

We are now in a position to answer the question: Given your timing error specification, how much can you gain in throughput? As wait time increases with load, we must ease off to achieve a desired timing error, ϵ, specified as a fraction of the mean inter-event interval for a cell, $T_{\rm cell}$. Thus, we have the constraint $T_{\rm wait} < \epsilon T_{\rm cell}$, or $T_{\rm wait} < \epsilon N_{\rm col} N_{\rm row} T$ if the interevent-interval for the entire array is T and it has $N_{\rm col}$ columns and $N_{\rm row}$ rows. Applying this timing constraint to (13) and solving for $T_{\rm bst}/T$ yields a throughput of

$$\hat{c} = \frac{1}{1 + \frac{b}{(\epsilon N_{\rm col}+1)}}. \tag{15}$$

This is the fraction of the link's capacity (i.e., the burst-rate $1/T_{\rm bst}$) we are allowed to use if we wish to keep normalized timing error less than ϵ. Conversely, we must tolerate timing errors of $(b-1)/N_{\rm col}$ if we wish to use 50% of the capacity, for example. Note that $\hat{c}$ must be greater than $2T_{\rm bst}/(T_{\rm cyc} + T_{\rm bst})$, or $1/(1 + b/2)$, to satisfy the heavy-load assumption used to derive (13), which requires that $\epsilon > 1/N_{\rm col}$.

The last result (15) implies that throughput gain, r (see Section II), cannot exceed the desired timing error times the number of columns. To see why, assume that most of the serial-read-write link's capacity, $1/T_{\rm cyc}$, is usable, which is reasonable enough [14]. In comparison, the usable amount of the parallel-read-write link's capacity is $\hat{c}/T_{\rm bst}$. When we substitute the expression for $\hat{c}$ from (15), divide by $1/T_{\rm cyc}$, and then subtract one, we obtain

$$r = \frac{b\epsilon N_{\rm col}}{b + \epsilon N_{\rm col} + 1} \tag{16}$$

for the throughput gain, where $b = T_{\rm cyc}/T_{\rm bst} - 1$ is the boost factor.

Throughput gain r is plotted as a function of boost factor b in Fig. 5, for various levels of timing error multiplied by the number of columns $\epsilon N_{\rm col}$: r saturates at $\epsilon N_{\rm col}$ when $b \gg \epsilon N_{\rm col} + 1$. Therefore, we cannot increase throughput more than $\epsilon N_{\rm col}$ times $1/T_{\rm cyc}$—no matter how small we make $T_{\rm bst}$. This limitation arises because the timing error limits how many events can be read in parallel. Basically, we have to read the row no latter than $\epsilon T_{\rm cell}$ s after the first event occurs. We cannot expect more than $\epsilon N_{\rm col}$ events to occur in that time, as the row has only $N_{\rm col}$ cells.

In summary, throughput can be enhanced substantially by going parallel if the acceptable timing error is on the order of a percent, assuming that the array has several hundred columns. Essentially, the timing specification sets how long we can wait for other cells in that row to become active, and therefore it limits how many events we can read in parallel. For example, for $T_{\rm cyc} = 200$ ns and $T_{\rm bst} = 20$ ns (which gives $b = 9$) and with $N_{\rm col} = 500$ and $\epsilon = 0.01$, we get $r = 3$ [from (16)].

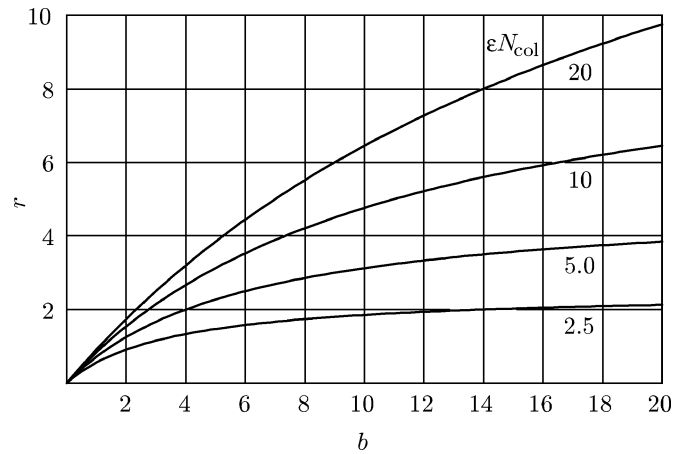


Fig. 5. Throughput increase. Throughput gain r obtained by increasing the boost factor b for various values of $\epsilon N_{\rm col}$, the product of timing error—expressed as a fraction of a cell's inter-event interval—and column count.

That is to say, even though the burst rate boosts the link's capacity nine-fold, the timing spec caps the gain in throughput to three-fold. This is a substantial enhancement, though smaller than expected. These conclusions follow from our assumption of Poisson-like behavior, which is indeed validated by test measurements (Section V). Before discussing them, we present measurements of the capacity of links implemented in three submicrometer technologies.

IV. Signaling Rates

The parallel-read-write link has thus far been used in five parallel image-processing chips fabricated in 0.6-, 0.4-, and 0.25-μm CMOS technologies. The first two were 0.6- and 0.4-μm imagers whose pixels converted photosignals into pulse frequency at the focal plane [3], [18]. The third was a 0.4-μm inverse imager whose pixels converted pulse-frequency back into analog current, which was video-encoded using an analog multiplexer (i.e., scanner) [19]. The fourth and fifth were 0.25-μm orientation-selective chips whose pixels received pulse trains from a silicon retina [18] and encoded their outputs also as pulse trains [20], [21].

In terms of the overhead in cell area, nine transistors are required for transmission whereas the prior serial-readout technique requires just four (reviewed in [14]). At five transistors, the number used for reception is unchanged. However, whereas serial-readout requires two lines per column, parallel-readout requires just one because cells are not selected individually. Trading a metal line for five transistors is worth it when metal lines are at a higher premium than transistors, which is the scaling trend.

Before taking a look at signaling rates, we review the communication protocol, which is based on a four-phase handshake [22]. As a concrete example, logic-analyzer traces captured from a 0.4-μm implementation are presented in Fig. 6. In this example, a single event is followed by a burst with two events. Communicating a single row-column address pair involves a sequence of eight transitions in the row request (Ry, active-high), column request (Rx, active-low), and acknowledge (Ack, active high for Ry but active low for Rx) signals.

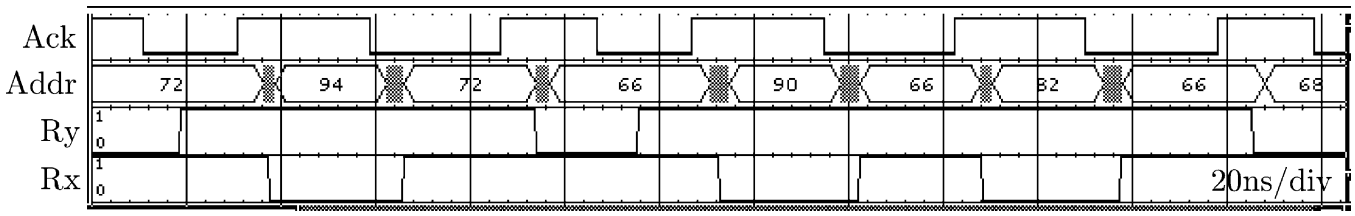
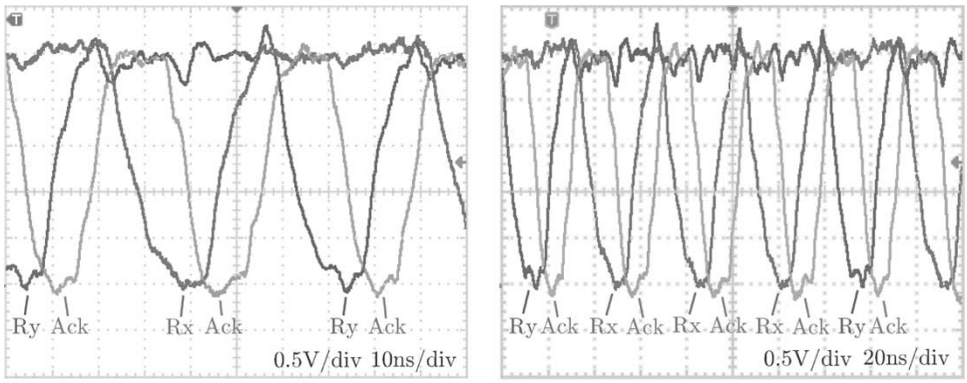


Fig. 6. Word-serial communication protocol. Transmitter outputs row address (Addr) and asserts row request (Ry, active-high). Receiver latches address (72) and asserts acknowledge (Ack, active-high for Ry). Transmitter then outputs column address and asserts column request (Rx, active-low). Receiver latches address (94) and asserts acknowledge (Ack, active-low for Rx). Now Rx goes high, followed by Ack, and Ry goes low, followed by Ack. A burst of two address-events is transmitted next (row address 66 and column addresses 90 and 82). These logic analyzer traces were captured from a 0.4-μm link.



(a) (b)

Fig. 7. Word-serial link signaling rates. (a) Eight transitions transmit a row–column address pair sequentially; it takes 70 ns. (b) Four transitions transmit each additional column address; it takes just 36 ns. There were three events in this burst. These scope traces were captured from a 0.25-μm link.

However, each additional column address requires only four transitions, involving Rx and Ack.[1] The address lines revert back to the row address when Rx is inactive (i.e., high), so the row address can be reread at any point during the burst. We added this feature to support receivers with limited buffering capability, such as microcontrollers, since a burst can be as long as an entire row [15].

To demonstrate the parallel-read-write link's signaling rate, we present scope traces captured from a 0.25-μm implementation in Fig. 7. These traces indicate that it takes 70 ns to send a row-column address pair but only 36 ns to send each additional column address, which gives a capacity of 27.8 M events/s. The pads and the ribbon cable (plus PCB trace) account for 6.7-ns delay per transition, as we measured rise and fall times 7.4 $\pm$ 1.4 ns long [see Fig. 7(a)] and round-trip propagation delays 6.0-ns long.[2] These off-chip sources would have added a total of 26.8 ns to the transmission time, but pipelining splits this delay half-and-half between the transmitter and the receiver. Therefore, these measurements lead us to conclude that the transmitter can issue a new column address every 22.6 ns. This value agrees with an estimate of 20.5 ns that we calculated using transistor sizes and capacitances extracted from the layout of the transceiver's 192 $\times$ 48 array, based on fabrication parameters supplied by the MOSIS service for this 0.25 μm process. Hence, we conclude that off-chip limitations increased $T_{\rm bst}$ from 22.6 to 36 ns.

TABLE I
BURST-MODE LINK PERFORMANCE

Tech (μm)	*Array*	*Pixel* (λ^2)	$T_{\rm cyc}$ (ns)	$T_{\rm bst}$ (ns)	*Word Serial*
0.60	60×80	107×100	98	43	No
0.40	60×96	200×173	77	37	Yes
0.25	48×192	250×173	72	36	Yes

Measured link timing for transmitters and receivers fabricated in three different technologies [transmitter array (rows $\times$ columns) and pixel sizes are given]. λ is half the minimum feature size given under *Tech*. $T_{\rm cyc}$ and $T_{\rm bst}$ are durations of inter-row and intra-row transmissions, as defined earlier, while *Word Serial* refers to multiplexing row and column addresses.

Table I summarizes the timing parameters we measured for the three submicrometer technologies that the link has been fabricated in. Array sizes and pixel areas listed are for the transmitter.[3] The array size of the 0.4-μm receiver was 176 $\times$ 132; that of the 0.25-μm receiver was identical to the 0.25-μm transmitter. Since we did not fabricate a 0.6-μm receiver, we used the 0.4-μm receiver's delay as an estimate for the 0.6-μm implementation.[4] The 0.4- and 0.25-μm implementations have similar timing, which supports our conclusion that performance is

[1]The transmission times are 10–20 ns longer than those listed in Table I because of off-chip logic added to decode cell type. There were four cell types tiled in 2 $\times$ 2-pixel blocks.

[2]Although we tried to match the 100-Ω impedance of the ribbon cable and PCB trace, our pad design was conservative. We estimated its impedance to be 150 Ω from the 1.0-V reflection ($\mathrm{Vdd} = 2.5\,\mathrm{V}$) we observed at the output pad.

[3]The cell drove the column lines with a 31λ wide p-type field-effect transistor (pFET) in the 0.6-μm design and with a 21λ wide n-type field-effect transistor (nFET) (u-shaped) in the 0.4- and 0.25-μm designs.

[4]This delay was added to the timing measurement made by tying the 0.6-μm transmitter's request signal directly to its acknowledge signal. This transmitter had a single request, as it output row and column addresses in parallel, unlike the other two transmitters.

limited by off-chip signaling. This limitation also explains why $T_{\rm bst}$ is half $T_{\rm cyc}$, as the former handshake involves half as many transitions. However, it is likely that doubling the number of columns in the 0.25-μm design also contributed to the lack of improvement; this is addressed in Section VI.

The transmission rate was also limited by an increase in crosstalk with activity. The 0.25-μm transceiver chip sustained a maximum rate of 22.7 M events/s between its transmitter-port and its receiver-port. This load represents 81.6% of its 27.8-M events/s capacity. Turning up the silicon neurons' activity further caused a step change from 22.7 to 25.0 M events/s, the peak transmission-rate we observed. We had to turn down the activity well below the trigger point to get the chip out of this state. Crosstalk mediated by the supply rails, the substrate, or the row and column lines can generate spurious events when it is amplified by the event-generators, which have high gain, achieved through positive feedback [3]. The mean number of spurious events triggered per event increases as the activity level increases, as it becomes more likely to find a generator close enough to threshold. The event rate explodes when each event triggers more than one event.

In addition to minimizing crosstalk by isolating low-level analog circuitry through circuit design and layout techniques [14], we also packaged the dice carefully. We took advantage of the 50% reduction in link-width (number of address lines) realized by multiplexing row and column addresses to intersperse power pads between both input and output pads in the 0.25-μm transceiver. Thus, every other digital pad was either $V_{\rm dd}$ or Gnd. This practice, common in high-speed processor design, reduces LdI/dt noise by dividing I among several parallel paths. We also minimized L by using a thin-quad-flat-pack (TQFP) package with just 3-nH lead inductance.[5] These measures yielded the 81.6% utilization figure quoted above, a much higher fraction of capacity than we have previously been able to use [23].

V. Theory Versus Practice

We verified that the 0.25-μm link behaves as theoretically predicted for loads up to 81.6% of its capacity, that is 22.7 M events/s. We calculated the fraction of events transmitted at $T_{\rm bst}$ (i.e., $(n_{\rm evt}-1)/n_{\rm evt}$) to determine the burst probability p. We obtained an excellent fit to this data when we solved (7) and (8) for p, as shown in Fig. 8(a). The transmission times that gave the best fit are almost identical to the times where peaks occur in the event-interval histogram (see Fig. 9). These peaks appeared at 73 and 37 ns whereas $T_{\rm cyc} = 68$ ns (-6.8%) and $T_{\rm bst} = 37$ ns (0%) yielded the best fit. Conversely, the observed transmission times yielded a theoretical burst probability that exceeded the measured value by 3.8% at 22.7 MHz (0.834 versus 0.803).

Although the theory fitted nicely at heavy loads, our measurements were up to 65% less than it predicted for light loads, as Fig. 8(b) shows. Lower than expected burst probability is not surprising since we assumed an exponentially distributed service time when, in actual fact, the service time is fixed. Such deterministic service times can reduce queuing to 50% of that in the exponentially-distributed model (Poisson assumption) [17], [24]. However, the row-service time distribution does become exponential as we transition from sending a single event from each row to sending bursts of several events, because burst-

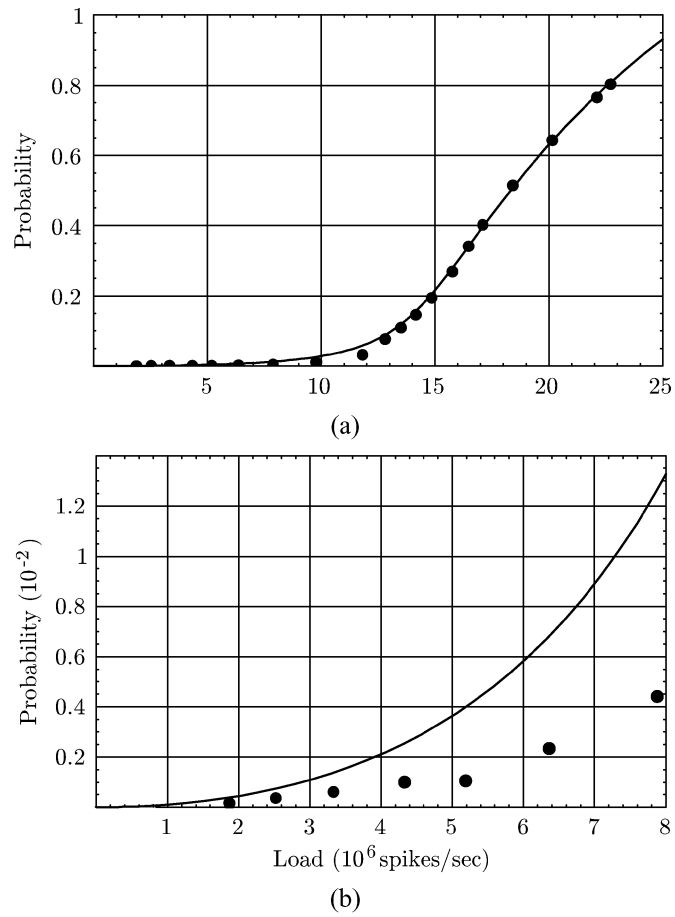


Fig. 8. Burst probability measurements. (a) Fit of the analytical solution for p [(8) and (7)], with $T_{\rm cyc} = 68$ ns, $T_{\rm bst} = 37$ ns, and $N_{\rm row} = 48$. (b) Zooming in on the light load range to show poor fit of low burst-probabilities. The data is from the 0.25-μm link.

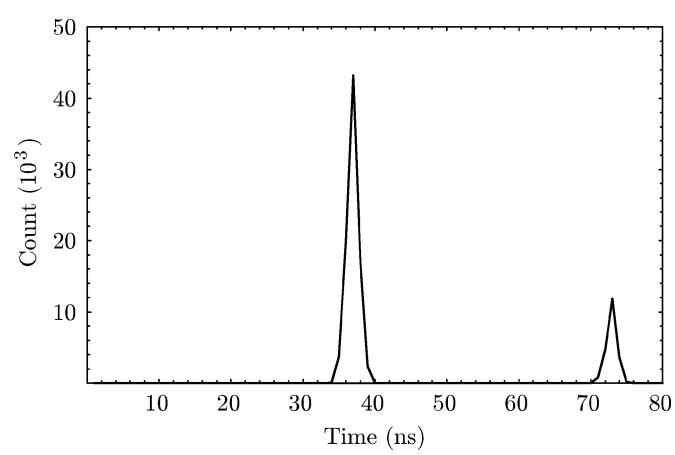


Fig. 9. Event-interval histogram. Distribution of intervals between 107 317 address events recorded in 4.72 ms—a 22.7-MHz mean rate—and time stamped at 0.5-ns resolution. Bin size is 1 ns. The first peak is at 37 ns; it corresponds to $T_{\rm bst}$. The second is at 73 ns; it corresponds to $T_{\rm cyc}$. This data is from the 0.25-μm link.

[5]This 120-pin package was 14 mm × 14 mm in size, with a 7 mm × 7 mm paddle size. The 2.1 mm × 4.8 mm die was placed near the top of the paddle and bonded on just three sides to minimize bond-wire length.

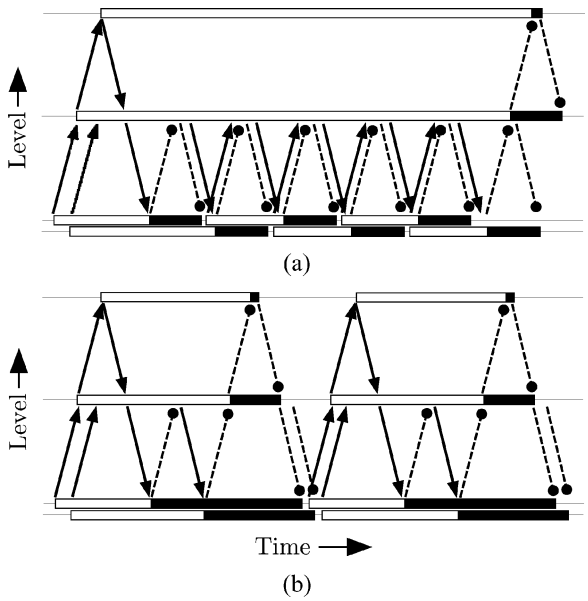


Fig. 10. Ping-pong in arbiter tree. White and black rectangles delimit set and clear handshake-phases for a cell (middle), its parent (top), and its two daughters (bottom); requests travel up while acknowledges travel down. These signals are set and cleared sequentially, as indicated by the solid arrows and the dashed pendulums, respectively. (a) Cell ping pongs from one daughter to the other one if that daughter makes another request while its sister is being serviced. (b) Cell avoids ping ponging by waiting until its parent's acknowledge clears before it clears its own acknowledges.

lengths are distributed geometrically [see (1)]. These variable-length bursts explain the excellent agreement between theory and experiment in the heavy-load regime.

The highest burst probability we measured in the 0.25-μm link was 0.9944, which occured at a load of 25.0 M events/s. This probability is pretty close to the maximum possible, as it corresponds to 180-event burst-lengths, whereas the chip has only 192 cells per row. However, we did not include this data point in Fig. 8 because crosstalk introduced significant correlations between event-generators at such high levels of activity, as explained earlier. These artificial correlations enhance the burst probability as they make it more likely that neighboring neurons will fire. Indeed, the measured value was significantly higher than the theoretically predicted burst probability for this load (0.9944 versus 0.9309). These high burst probabilities confirm that the fair arbiter design [15] introduced in our 0.25-μm transceiver chip successfully eliminated load shedding, which plagued previous link implementations [14].

A fair arbiter design is crucial to achieving the theoretically predicted throughput-gain. As shown in Fig. 1, a 1-in-m arbiter is built by connecting $\mathsf{m}-1$ 1-in-2 arbiters in a binary tree. In the greedy arbiter design presented in [14] and [25], the top cell in the tree ignores one side completely when more than half of the arbiter's clients are waiting for service. This behavior first occurs in the row arbiter, and it migrates down the tree as the load is increased further, locking out more and more rows. Such load shedding is detrimental in a parallel-read design because it prevents the burst probability from exceeding 0.5, which corresponds to an average of just two active cells per row. This value of p follows from the identity $p = (m_{\mathrm{row}}/N_{\mathrm{row}})q$ with the number of row requests $m_{\mathrm{row}} = N_{\mathrm{row}}/2$ and the row load $q \approx 1$. When the burst probability fails to increase, the average transmission time does not decrease, and hence empty transmission slots needed to send the locked-out rows' events do not materialize.

Load shedding happens when a daughter that has just been serviced makes a new request while the other daughter is being serviced—and gets selected again. Selection continues to ping-pong back and fourth, as illustrated in Fig. 10(a), where a possible fix also is presented [Fig. 10(b)]: Do not clear your acknowledge until your parent clears her's. This ordering, which mirrors the set sequence, requires requests at all levels of the tree to be cleared, starting at the bottom and propagating all the way up, before the acknowledges are cleared, starting at the top this time and propagating all the way down. Traversing the entire tree like this makes this reshuffling painfully slow [compare Fig. 10(a) and (b)]. An alternative sequencing that is optimized for speed is presented in the companion paper [15]. The results presented here confirm that that fair arbiter design can indeed achieve burst probabilities that exceed 0.5.

VI. Summary and Conclusion

We analyzed and tested a 2-D address-event communication link that was designed to take advantage of high integration densities available in deep submicrometer processes. For scalability, it exploits a linear increase in the number of active cells per row with increasing array size by reading and writing events in parallel. Our analysis revealed that the resulting throughput-gain can be severalfold if a timing error as large as this gain divided by the number of columns is acceptable. For instance, a two-fold throughput gain is achievable with subpercent timing errors in a 200-column array. This conclusion holds if event generation and service are Poisson. This assumption is validated by the excellent agreement between predicted and measured burst probabilities in the heavy-load regime.

In practice, the parallel read-write technique did deliver on its promise of boosting absolute throughput—breaking the array's row cycling limit—but we achieved only a twofold gain because off-chip signaling limited the burst-rate. Whereas it took 36 ns to transmit a column address, it appears that the 0.25-μm transmitter can issue one every 23 ns, but the pads added 7 ns while the 15 cm-cable added 6 ns. Thus, off-chip signaling limitations cut capacity from 43 to 28 M events/s. Sophisticated asynchronous off-chip signaling techniques will be required to achieve substantial throughput gains, in addition to optimizing critical timing paths on chip.

The critical path in the transmitter (receiver) is now the column encoder (decoder). The encoder drives capacitance proportional to the column count when it outputs the column address, and when it feeds the acknowledge signal back to the column-arbiter interfaces. Similar considerations apply to the decoder. Hence, the transmitter (receiver) takes more time to encode (decode) events as the number of columns increases. This linear scaling can be avoided by using a hierarchical organization just like the arbiter does—its two-input cells are connected in a binary tree. This approach yields a logarithmic delay scaling; the price is a logarithmic increase in wiring [22].

ACKNOWLEDGMENT

The author would like to thank B. Taba for his insightful suggestions about analyzing and testing the link design and P. Merolla for help in obtaining data from his 0.25-μm transceiver chip.

REFERENCES

[1] M. Mahowald, *An Analog VLSI Stereoscopic Vision System*. Boston, MA: Kluwer, 1994.

[2] K. A. Boahen, "The retinomorphic approach: Pixel-parallel adaptive amplification, filtering, and quantization," *Analog Integr. Circuits Signal Process.*, vol. 13, pp. 53–68, 1997.

[3] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, pp. 281–294, Feb. 2003.

[4] J. Kramer, "An on/off transient imager with event-driven asynchronous read-out," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, 2002, pp. II-165–II-168.

[5] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie, "Silicon auditory processors as computer peripherals," *IEEE Trans. Neural Networks*, vol. 4, pp. 523–528, May 1993.

[6] W. Yang, "A wide-dynamic range low-power photosensor array," in *Proc. Int. Solid-State Circuits Conf.*, vol. 37, 1994, p. 230.

[7] B. Fowler, A. E. Gamal, and D. Yang, "A cmos area image sensor with pixel-level A/D conversion," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC'94)*, vol. 37, San Francisco, CA, 1994, pp. 226–227.

[8] L. G. McIlrath, "A low-power low-noise ultrawide-dynamic-range cmos imager with pixel-parallel A/D conversion," *IEEE J. Solid-State Circuits*, vol. 36, pp. 846–853, May 2001.

[9] A. Murray and L. Tarassenko, *Analogue Neural VLSI: A Pulse Stream Approach*. London, U.K.: Chapman & Hall, 1994.

[10] A. Mortara, E. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE J. Solid-State Circuits*, vol. 30, pp. 660–669, June 1995.

[11] A. Abusland, T. S. Lande, and M. Hovin, "A VLSI communication architecture for stochastically pulse-encoded analog signals," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 3, May 1996, pp. 401–404.

[12] M. Sivilotti, "Wiring considerations in analog vlsi systems, with application to field-programmable networks," Ph.D. dissertation, Dept. Comp. Sci., California Inst. Technol., Pasadena, CA, 1991.

[13] S. R. Deiss, R. J. Douglas, and A. M. Whatley, "A pulse-coded communications infrastructure for neuromorphic systems," in *Pulsed Neural Networks*, W. Maass, Ed. Boston, MA: MIT Press, 1999, ch. 6, pp. 157–178.

[14] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address-events," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 416–434, May 2000.

[15] ——, "A burst-mode word-serial address-event link—I: Transmitter design," *IEEE Trans. Circuits Syst. I* , vol. 51, pp. 1269–1280, July 2004.

[16] ——, "A burst-mode word-serial address-event link—II: Receiver design," *IEEE Trans. Circuits Syst. I* , vol. 51, pp. 1281–1291, July 2004.

[17] M. Schwartz, *Telecommunication Networks: Protocols, Modeling, and Analysis*. Reading, MA: Addison-Wesley, 1987.

[18] K. Zaghloul and K. A. Boahen, "Optic nerve signals in a neuromorphic chip—II: Testing and results," *IEEE Trans. Biomed. Eng.*, vol. 41, pp. 667–675, Apr. 2004.

[19] C. A. Mead and T. Delbruck, "Scanners for visualizing analog vlsi circuitry," *Analog Integr. Circuits Signal Process.*, vol. 1, pp. 93–106, 1991.

[20] T. Y. W. Choi, B. E. Shi, and K. Boahen, "An orientation-selective 2D AER transceiver," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, May 2003, pp. 800–803.

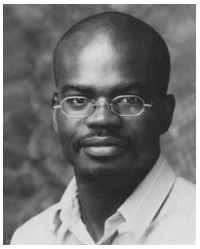
[21] P. Merolla and K. Boahen, "A recurrent model of orientation maps with simple and complex cells," in *Advances in Neural Information Processing*, S. Thrun and L. Saul, Eds. San Mateo, CA: Morgan Kaufman, 2003, vol. 15.

[22] C. A. Mead, *Introduction to VLSI Systems*. Reading, MA: Addison Wesley, 1980.

[23] K. A. Boahen, "A retinomorphic chip with parallel pathways: encoding ON, OFF, INCREASING, and DECREASING visual signals," *Analog Integr. Circuits Signal Process.*, vol. 30, no. 2, pp. 121–135, 2002.

[24] L. Kleinrock, *Queueing Systems*. New York: Wiley, 1976.

[25] K. A. Boahen, "Retinomorphic vision systems II: communication channel design," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. suppl., May 1996, pp. 14–17.



Kwabena A. Boahen received the B.S. and M.S.E. degrees in electrical and computer engineering from The Johns Hopkins University, Baltimore, MD, in the concurrent masters-bachelors program, both in 1989, and the Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, in 1997.

He is an Associate Professor in the Bioengineering Department at the University of Pennsylvania, Philadelphia, where he holds a secondary appointment in electrical engineering. His current research interests include mixed-mode multichip VLSI models of biological sensory and perceptual systems, and their epigenetic development, and asynchronous digital interfaces for interchip connectivity.

Dr. Boahen was awarded a Packard Fellowship in 1999, a National Science Foundation CAREER Grant in 2001, and an Office of Naval Research YIP Grant in 2002. He is a member of Tau Beta Kappa and has held a Sloan Fellowship for Theoretical Neurobiology at the California Institute of Technology.