Programmable Connections in Neuromorphic Grids

Joseph Lin, Paul Merolla, John Arthur, and Kwabena Boahen
Department of Bioengineering, University of Pennsylvania, Philadelphia, Pennsylvania 19104–0250
Email: {linjh,pmerolla,jarthur,boahen}@seas.upenn.edu

***Abstract*— We describe asynchronous circuits that can relay spikes between multiple chips in a grid. These circuits interface with an on-chip SRAM to implement programmable connectivity among chips. We introduce a packet format that is compatible with updating the SRAM. From a high level specification, we synthesized and fabricated these circuits in an area of 0.206mm^2 in 0.18-μm CMOS technology. Test results that measure performance and demonstrate correct function on first silicon are presented.**

I. Layered Neuromorphic Systems

Neuromorphic engineers are building single-chip systems with ever-increasing complexity, at both the neuronal and microcircuit levels. At the neuronal level, the conductance-based neuron is displacing the integrate-and-fire one [1]–[3]. At the microcircuit level, the canonical microcircuit—stereotyped local connectivity among specialized cell types—is displacing the free-standing all-purpose neuron [4,5]. This trend has capped the number of pixels and bands in neuromorphic vision and auditory systems—despite on-going miniaturization—bumping up against the single-chip limit.

Faced with the problem of finite area, the neocortex took advantage of the third dimension, organizing its various neuronal types into six layers and transforming its canonical microcircuit into a columnar circuit [4]. Inspired by this architecture, neuromorphic engineers have looked to 3D microfabrication techniques [6,7]. Though impressive systems have been built, 3D-integration remains a niche solution, used mainly in memory devices [8], where structure is regular, functionality fixed and bandwidth enormous—features that do not apply to neuromorphic systems.

Here we propose a solution that supports and exploits neuromorphic systems' unique features. On the one hand, bandwidth is discounted: bandwidth-per-neuron is a millionth of a wire's. This favors packet-switching, where wires are shared by numerous users (as in the Internet), over circuit-switching, where a wire is dedicated to a single pair of users (as in the archaic phone system) [9,10]. On the other hand, flexibility is priced: functionality comes from reconfiguring connections (through developmental and learning mechanisms). This also favors packet-switching: packets may be rerouted on the fly simply by relabelling them [11] whereas circuits must be set-up in advance.

In our multichip architecture, layers of different neurons reside on different copies of the same chip (differentiated by electronically adjustable parameters) and the columnar circuit (CC) spans corresponding locations on these chips (Fig. 1). To restrict the size of the look-up-table that specifies its connectivity, we assume the CC is translation-invariant, thus there are only as many entries as there are chips. If there is a connection, the entry also specifies its properties, physiological (e.g., excitatory or inhibitory) or, in the case of a multicompartment neuron, anatomical (e.g., basal or apical). In this way, any desired CC may be realized simply by programming the appropriate bits into a small SRAM that fits snugly on the periphery of the chip.

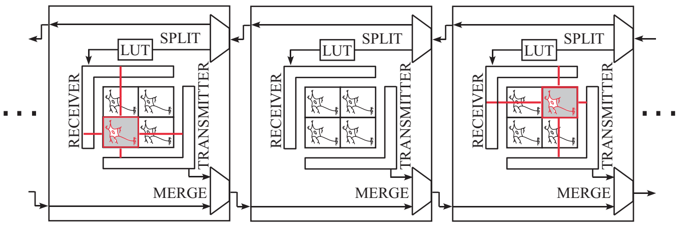

Fig. 1. Programmable Columnar Connections. Packets travel through each chip rightward along the bottom path and leftward along the top path. The addition of an SRAM based look-up-table allows programmable connectivity; packets can be remapped to other locations, or can skip chips.

Section II describes our architecture's connectivity network. Section III presents our network's high-level specification and Section IV decomposes this into smaller, concurrent processes. Section V presents synthesized circuits and Section VI presents test results.

II. Grid Networks

Capitalizing on existing chip-to-chip links [12], grids were developed to broadcast a packet consisting of a neuron's chip, row and column addresses whenever it spikes; these packets are relayed from chip to chip [13]. A packet may have multiple column addresses (following the chip and row address) corresponding to multiple spikes read from the same row. More recent work added the capability to selectively deliver or filter a packet based on its chip address; a particular chip may be either targeted or excluded [14]. The excluded mode provides the functionality we desire for broadcasting spikes; we do not wish to send them back to the same chip. The targeted mode provides the functionality we desire for programming; we wish to write a byte to a particular SRAM on a particular chip.

Targeted and excluded delivery capitalize on the relative addressing scheme grids use. To support relative addressing, a packet's chip address is incremented or decremented as it hops from chip to chip. Thus, we can arrange for an overflow or underflow to occur at a particular chip by modifying the

TABLE I

CHP NOTATION

Operation	Notation	Explanation
Process	P_i	A composition of communications
Guarding	G_i	$\equiv B_i \rightarrow P_i$. Execute P_i if B_i is true
Sequential	$P_1; P_2$	P_1 ends before P_2 starts
Concurrent	$P_1 \\| P_2$	There is no constraint
Repetition	$*[P_1; P_2]$	$\equiv P_1; P_2; P_1; P_2; \ldots$. Repeats forever
Selection	$[G_1 [\!] G_2]$	Execute P_i for which B_i is true
Arbitration	$[G_1 \mid G_2]$	Required if B_i not mutually exclusive
Input	$A?x$	Read data from port A to register x
Output	$A!x$	Write data from register x to port A
Probe	$\overline{A}$	Is communication pending on port A?
Packet	$p[j].i$	The ith bit of p's j-th word
Assignment	$y := x$	Copy data from x to y

chip address (initially zero by default). This chip then becomes the one the spike is delivered to in targeted mode or filtered from in excluded mode, as specified by a mode-bit. In this work, we dispense with the mode-bit. Instead, we write to the memory (i.e., program) when an underflow occurs; otherwise we perform a read (i.e., table look-up).

III. PROGRAMMABLE CONNECTIVITY

To relay packets between chips, we use a communication controller named $\top$ [14]. We describe the $\top$ process using CHP (Communicating Hardware Processes) notation [15] (see Table I):

$$\begin{aligned} *[t := \mathsf{DECR}(R?); L!t \parallel [t[1].\mathsf{borr} \rightarrow \mathsf{WRITE}(t) \\ [\!]\neg t[1].\mathsf{borr} \rightarrow k := \mathsf{READ}(t); \\ [k.0 \rightarrow D!\mathsf{SEND}(t,k) [\!] \neg k.0 \rightarrow \mathsf{skip}]]] \end{aligned}$$

where $\neg$ denotes complement, k is a local byte, and t is a packet. Packets are received from R, decremented (DECR), sent off-chip on L, and delivered (SEND) to the receiver on D—but only if the bit ($k.0$) read (READ) from the SRAM is set. By appending the other bits to the packet, we can address different targets on different chips. The SRAM is not read when the underflow bit ($t[1]$.borr) is set. Instead, the packet is written (WRITE) to the SRAM.

The function calls perform the following:

$\mathsf{DECR}(\{w(1), \ldots, w(N)\})$ overwrites $w(1)$ with $w(1)-1$
$\mathsf{READ}(\{w(1), \ldots, w(N)\})$ reads from location $w(1)$
$\mathsf{WRITE}(\{w(1), \ldots, w(N)\})$ writes $w(3)$ to location $w(2)$
$\mathsf{SEND}(\{w(1), \ldots, w(N)\}, b)$ delivers packet with b appended

Notice that only the headword, which corresponds to the chip address, is decremented. It serves as the address during a read but is discarded during a write (it is always -1 in this case). The next two words serve as address and data, respectively. In the next section, we will decompose the programmable $\top$'s CHP specification.

IV. PROCESS DECOMPOSITION

We decompose the high-level specification into smaller, concurrent processes following Martin's synthesis procedure [15] (Fig. 2). We begin by decomposing DECR into two processes with the following CHP (bit-level):

$$\begin{aligned} &\mathsf{DEC} \equiv *[A?a, B?b; D!\mathsf{dec}(a,b), C!\mathsf{bor}(a,b)] \\ &\mathsf{DCTL} \equiv *[F?f; [f = \phi \rightarrow h := 1 [\!] f \neq \phi \rightarrow [h \rightarrow B!1; \\ &\qquad h := 0 [\!] \neg h \rightarrow B!0]]] \end{aligned}$$

DEC receives a data-bit on A and a borrow-bit on B (from the previous DEC) and outputs a data-bit on D and a borrow-bit on C. DCTL ensures only head-words are decremented; it feeds a true borrow-bit into the least-significant DEC on B when it detects an end-of-packet symbol (ϕ) on F.

We decompose the sequence following DECR into six concurrent processes (word-level):

$$\begin{aligned} &\mathsf{SPLIT} \equiv *[P?w; L!w \| M!w] \\ &\mathsf{SRAM} \equiv *[M_a?a; [\overline{W} \rightarrow P?m[a]; W [\!] \overline{R} \rightarrow R!m[a]]] \\ &\mathsf{SWITCH} \equiv *[\overline{A} \rightarrow A!(Q?) [\!] \overline{B} \rightarrow B!(Q?) [\!] \overline{C} \rightarrow C!(Q?)] \\ &\mathsf{MCTL} \equiv *[U?u; [u \rightarrow A?; M_a!(A?) \| F \| W; E \\ &\qquad [\!] \neg u \rightarrow M_a!(A?); R?k; [k.0 \rightarrow J!k; J [\!] \neg k.0 \rightarrow E]]] \\ &\mathsf{FILTER} \equiv *[\overline{F} \rightarrow P!(B?); F \\ &\qquad [\!] \overline{E} \rightarrow B?w; [w = \phi \rightarrow E [\!] w \neq \phi \rightarrow \mathsf{skip}]] \\ &\mathsf{SEND} \equiv *[C?w; D!\mathsf{app}(w,k) \\ &\qquad \| [w = \phi \rightarrow J; J?k [\!] w \neq \phi \rightarrow \mathsf{skip}]] \end{aligned}$$

where m is the memory array and u is the borrow-bit output by the most-significant DEC. MCTL monitors u to determine the approriate memory operation.

For a write (u true), MCTL supplies the packet's second word as the address (it dumps the head-word, which is always -1 in this case). It signals FILTER on F to supply the third word as data to SRAM, then executes the write on W. A communication on E signals FILTER to delete the remainder of the packet. For a read (u false), MCTL supplies the head-word as the address and executes the read on R. Bit zero determines whether the packet is delivered or filtered (i.e., MCTL signals SEND on J or FILTER on H).

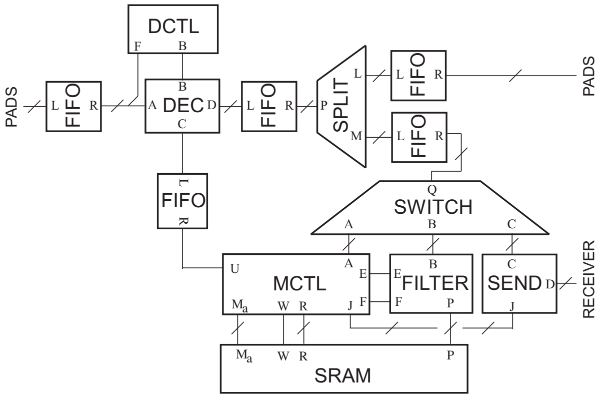

Fig. 2. Block diagram of the programmable $\top$. Packets come in from the left (PADS) and go out on the right (PADS). Each packet's head-word is decremented (DEC); the borrow-bit is set (DCTL) when an end-of-packet symbol is detected. Packets are also selectively delivered (SEND), or deleted (FILTER), via a fork in the datapath (SPLIT) and a mux (SWITCH), which also provides addresses (MCTL) for memory (SRAM) writes and reads, and data for writes (via FILTER). FIFO buffers are dispersed throughout the datapath to maintain throughput. Lines without slashes represent dataless or one-bit communications.

TABLE II

1-OF-4 CODE

binary	00	01	10	11
1-of-4	0001	0010	0100	1000

We have decomposed the programmable ⊤ into eight concurrent processes. In the next section, we synthesize these into circuits.

V. CIRCUIT SYNTHESIS

The next step in Martin's synthesis procedure is handshaking expansion (HSE), the procedure in which each CHP communication is expanded into a full four-phase request-acknowledge sequence. An active port (A) is paired with a passive port (B): A asserts the request (ao+) and waits for the acknowledge ([ai]), then deasserts the request (ao-) and waits for the acknowledge to clear ([∼ai]) before initiating the next handshake. The passive port B waits for the request ([bi]) before asserting the acknowledge (bo+), then waits for the request to clear ([∼bi]) before deasserting the acknowledge (bo-).

In this step, we also choose the data-encoding scheme. We use 1-of-4 (Table II) instead of a bundled-data (binary) because we want our circuits to be delay-insensitive. In 1-of-4, the data is **valid** if any one of a set of four wires is driven; otherwise it is **neutral** [15]. This validity functions as the request; bundled-data requires a separate signal for this purpose. A 4-input OR checks for validity or neutrality in a single 1-of-4 set, whereas a datapath with M sets requires M 4-input ORs that feed into a tree composed of M-1 C-elements (VN in Fig. 3a). We name this validity signal va for port A. Since 1-of-4 encodes two bits, we use 1-of-2 code (dual-rail) when we want to encode a single bit. For bit b, we label these two signals b.0 ($b = 0$) and b.1 ($b = 1$).

We begin with DCTL, which feeds DEC a true borrow-bit on B after it detects ϕ on F. DCTL's F acknowledge to FIFO is relayed through DEC, which acknowledges FIFO after the B communication. DCTL's HSE:

```
# DCTL #
  *[[~bi&fe.1]; h+; b.0+; [bi&~fe.1]; b.0-]
|| *[[~bi&h&fe.0]; b.1+; [bi&~fe.0]; h-; b.1-]
|| *[[~bi&~h&fe.0]; b.0+; [bi&~fe.0]; b.0-]
```

The first sequence sets the state variable h upon receiving ϕ (fe.1). The second sequence sends a true borrow-bit (b.1) during the next word (head-word); otherwise the third sequence sends a false borrow-bit (b.0).

We expand MCTL's CHP program into two HSE sequences: MADR and MRD. MADR implements the program up until the R communication. At this point, we introduce a channel, M, to communicate with MRD, which implements the rest of MCTL's program.

MADR controls FILTER, MRD, and SRAM, thus its E, F, M, and W ports are made active, whereas both A and U are passive. We introduce a latch (AALAT in Fig. 3a) between MADR and SWITCH, whose A, B, and C ports are passive as they are probed (implemented as [ai], etc.). AALAT keeps SRAM's address valid while SWITCH provides the data to FILTER. MADR's HSE:

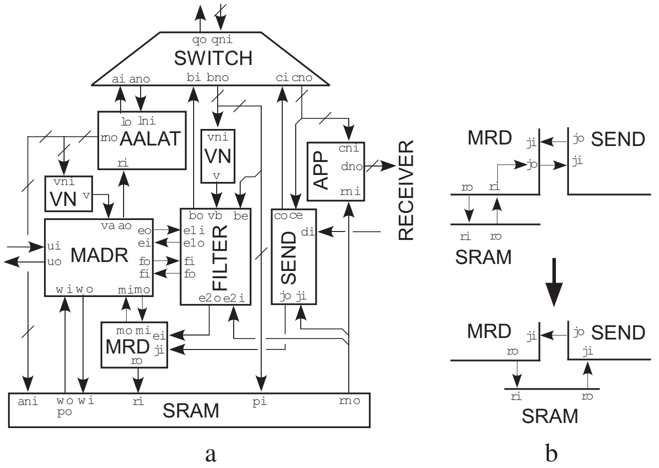

Fig. 3. Detailed block diagram of the programmable ⊤'s memory access circuits. **a** When a true underflow bit arrives on ui, MADR signals AALAT to discard the head-word of the packet and latch the second word for SRAM's address (ani). MADR executes the write and instructs FILTER to signal SWITCH to supply the third word as SRAM's data. FILTER subsequently deletes the remainder of the packet. If the underflow bit is false, MADR signals AALAT to latch the head-word for SRAM's address, and signals MRD to execute the read. If bit four is set (ji), SEND signals SWITCH to supply the second word to APP, which appends bits two and three to the words delivered to RECEIVER; otherwise (e2i), FILTER deletes the remainder of the packet. VN checks for validity or neutrality of 1-of-4 data. **b** MRD's jo and ri become shorted when we reshuffle the HSE sequence, resulting in a triangular communication between MRD, SEND, and SRAM.

```
# MADR #
   *[[u.1 & va]; s+; uo+; ao+; [~u.1 & ~va];
       k+; uo-; ao-; [va]; fo+; [fi]; wo+;
      [wi]; s-; fo-; [~fi]; eo+; [ei]; ao+;
     wo-; [~va & ~wi]; k-; eo-; [~ei]; ao-]
|| *[[ u.0 &  va]; mo+;  [mi]; uo+; ao+;
    [~u.0 & ~va]; mo-; [~mi]; uo-; ao-]
```

When data is valid (va), ao goes high to latch it and acknowledge SWITCH. In the memory write sequence (top), we introduce state variables s and k to distinguish the two communications on A. We move the F request (fo+) before the second A acknowledge (ao+)—but after the second va—to pipeline the communication. At this point, the third word of the packet is available from SWITCH, thus by asserting fo, FILTER can direct data to SRAM at the same time MADR signals the write (wo+). Once SRAM acknowledges, MADR signals FILTER (eo+) to delete the remainder of the packet. In the memory read sequence (bottom), MADR signals MRD (mo+) to execute the read.

In MRD's sequence, we make the R, E, and J ports active. We reshuffled its communications with SEND and SRAM into a triangular communication (see Fig. 3b): SRAM's R acknowledge becomes MRD's J request to SEND. Thus, when MRD signals a read, SRAM outputs data to SEND, which acknowledges MRD. A similar triangle occurs among MRD (R and E), SRAM (R), and FILTER (E). With these two triangles, MRD's HSE becomes:

```
# MRD #
   *[[mi]; ro+; [ei | ji]; mo+, ro-;
     [~mi & ~ei & ~ji]; mo-]
```

When MRD initiates the read (ro+), either FILTER (ei) or SEND (ji) acknowledges—depending on the value of the dual-rail bit from SRAM.

We implement FILTER's CHP program as three sequences, one for E and F and a third to handle multiple communications on B, which reads words until ϕ:

```
# FILTER #
   *[[ fi]; bo+; [ vb]; fo+;
     [~fi]; bo-; [~vb]; fo-]
|| *[[ ei]; bo+; [be.1 & vb]; eo+;
     [~ei]; bo-; [~vb]; eo-]
|| *[[be.0 & vb]; bo-; [~vb]; bo+]
```

In the first sequence, when MADR asserts fi, FILTER signals SWITCH (bo+) to supply data to SRAM. MADR releases fi when the write is complete. In the second sequence, ei signals FILTER to find ϕ (be.1); the third process receives words otherwise.

We implement SEND's C and D communications with SWITCH and RECEIVER as a triangular communication, with both ports active (see Fig. 3a). APP is implemented as combinational logic. SEND relays RECEIVER's acknowledge to SRAM through MRD. SEND's HSE:

```
# SEND #
   *[[ ji]; co+; [ce.1 & di]; jo+;
     [~ji]; co-; [~di]; jo-]
|| *[[ce.0 & di]; co-; [~di]; co+]
```

The first sequence waits for ji from SRAM before it signals SWITCH (co+) to supply the RECEIVER with the first word (row-address); it acknowledges MRD when ϕ appears (ce.1). The second sequence signals SWITCH to supply words to RECEIVER (column-addresses) otherwise.

Once we have HSE sequences, the final step is to compile them into production-rule sets (PRS), which are straightforward to implement with CMOS transistors [15]. Due to space constraints, we show only the synthesized CMOS circuits (see Fig. 4). In the next section we present test results that validate functionality and measure performance.

VI. Test Results

We present test results from a full-custom chip fabricated in 0.18-μm CMOS technology. This chip has a die area of 29.59mm^2 (6.19mm×4.78mm) with the programmable $\top$ occupying 0.697% of that area (Fig. 5). The array is comprised of 4×320×240 pixels: a group of 4 pixels corresponds to RGBY color video output (half VGA resolution). Two bits from the SRAM (256×16-bit) determine which of these four pixels is targeted. Each pixel is a pulse-extender configured to function as either a sample-and-hold or a leaky integrator.

In testing the chip, we are concerned with performance and functionality. One measure of performance is **burst rate**, the rate at which a chip receives words within a packet. We measured the maximum burst rate to be 62.9Mhz (Fig. 6), a 38.2% improvement upon the previous best of 45.5MHz (0.25-μm CMOS) [14].

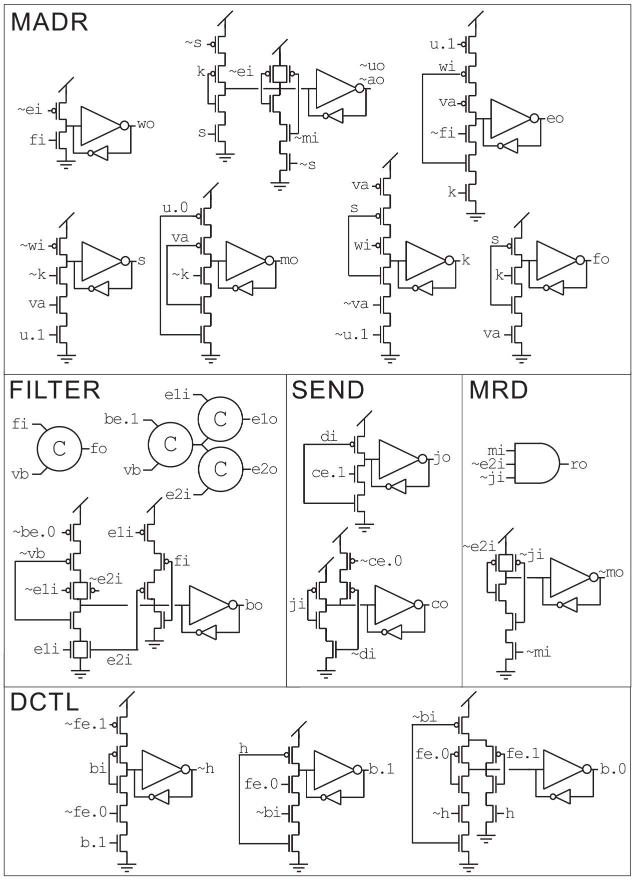

Fig. 4. Synthesized circuits. The circles with a C are C-elements, whose output goes high when both inputs are high, and low when both inputs are low. The small inverters (weak-feedback) are used to hold state. '∼' denotes signals that are complemented with an inverter (not shown). MCTL's two E communications use different ports in FILTER ($E1$ and $E2$).

We tested functionality by simulating activity from virtual chips with a computer through a USB (Universal Serial Bus) link using the packet format described in Fig. 7. When the array receives an address-event, the pixel with the corresponding address outputs a current. By measuring the response of a single pixel, we successfully demonstrated programming, filtering, and delivery (Fig. 8).

VII. Future Potential

We presented a programmable $\top$ that can relay packets between multiple chips in a grid together with test results that prove correct function on first silicon.

The programmable $\top$ is superior to an architecture that uses an external memory. With an external memory, each incoming packet would require N outgoing packets: one for each chip in a system with N chips. Thus, for links of the same bandwidth, only $1/N$th as many connections can be implemented. By integrating the SRAM on-chip and pipelining the datapath, we achieve an expandable solution.

The programmable $\top$ is ideal for building layered neural networks with stereotyped columnar connectivity similar to the neocortex. Thus, neuromorphic engineers can finally develop

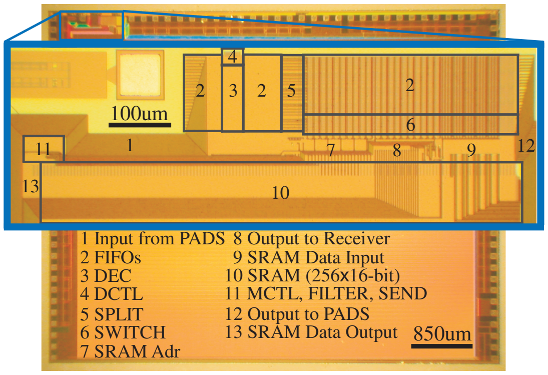

Fig. 5. Micrograph of the chip die containing the programmable ⊤. These circuits have an area of 0.206mm^2 (0.802mm×0.257mm).

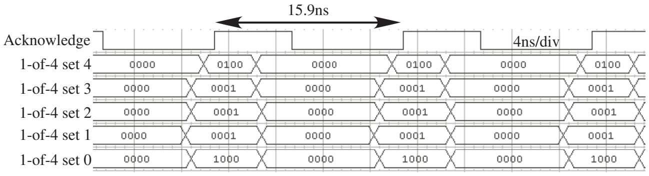

Fig. 6. Logic analyzer traces demonstrating maximum burst rate of 62.9MHz. The receiver acknowledges when the data is valid and releases the acknowledge when the data is neutral. The bits are arranged from MSB to LSB (from set 4 to set 0 and left to right within the set).

systems that expand beyond the single-chip limit.

Acknowledgment

The authors would like to thank Rajit Manohar and Virantha Ekanayake for providing the layout of the SRAM bank. We also thank Rodrigo Alvarez for designing the pulse-frequency receiver array used on this chip. This work was funded by NSF (CAREER Award ECS00-93851) and MOSIS (MEP).

References

[1] J. Elias, H.-H. Chu, and S. Meshreki, "A neuromorphic impulsive circuit for processing dynamic signals," in *Circuits and Systems, IEEE International Symposium on*, vol. 5, 1992, pp. 2208–2211 vol.5.

[2] M. Simoni and G. Cymbalyuk et. al, "A multiconductance silicon neuron with biologically matched dynamics," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 2, pp. 342–354, 2004.

[3] J. Arthur and K. Boahen, "Recurrently connected silicon neurons with active dendrites for one-shot learning," in *Neural Networks, IEEE International Joint Conference on*, vol. 3, 2004, pp. 1699–1704 vol.3.

[4] R. J. Douglas and K. A. Martin, "Neuronal circuits of the neocortex," *Annual Review of Neuroscience*, vol. 27, no. 1, pp. 419–451, 2004.

[5] P. Merolla and K. Boahen, "A recurrent model of orientation maps with simple and complex cells," in *Advances in Neural Information Processing Systems 16*, S. Thrun and L. Saul, Eds. MIT Press, 2004, pp. 995–1002.

[6] H. Kurino and M. Nakagawa et. al, "Smart vision chip fabricated using three dimensional integration technology," in *Advances in Neural Information Processing Systems 13*, T. Leen, T. Dietterich, and V. Tresp, Eds. MIT Press, 2000.

[7] E. Culurciello and A. Andreou, "Capacitive coupling of data and power for 3d silicon-on-insulator vlsi," in *Circuits and Systems, IEEE International Symposium on*, 2005, pp. 4142–4145 Vol. 4.

[8] M. Johnson and A. Al-Shamma et. al, "512-mb prom with a three-dimensional array of diode/antifuse memory cells," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 1920–1928, 2003.

[9] M. Mahowald, *An analog VLSI system for stereoscopic vision.* Boston: Kluwer, 1994.

Programming Packet

	9 8 7 6 5	4	3 2	1	0
Head	target chip-address			x	0
SRAM address	source chip-address			x	0
SRAM data	x x x x x	K	AP	x	0
Tail	x x x x x x x x x				1

Address-event Packet

	9 8 7 6 5 4 3 2	1	0
Head	source chip-address	x	0
Row	address		0
Col	address		0
Tail	x x x x x x x x x		1

Fig. 7. Programming and address-event packet formats. Both packets are comprised of four words; the LSB is the tailbit. **Programming:** Head is set so that its value is zero at the target chip. Words one and two are address and data. Packets are delivered to the on-chip receiver if K is set, with AP appended. **Address-event:** Head contains the relative address of the packet's source chip. Words one and two are the row and column-addresses; there can be multiple column-addresses.

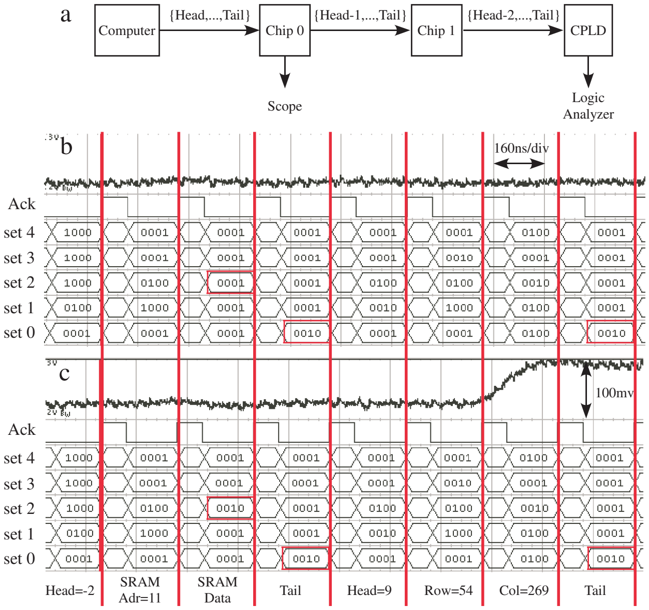

Fig. 8. Test setup and scope and logic analyzer (LA) traces. **a** In each experiment, the computer sends two packets, a programming instruction and a single address-event. The scope and LA traces are captured from the first and second chip, respectively (with the acknowledge delayed by a CPLD for data clarity). Thus, both packets' LA traces show head-words that have been decremented twice. **b** The scope trace remains flat because the ⊤ was programmed to filter packets. **c** The scope trace rises because the ⊤ was programmed to deliver packets.

[10] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 5, pp. 416–434, 2000.

[11] B. Taba and K. Boahen, "Topographic map formation by silicon growth cones," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2003.

[12] K. Boahen, "A burst-mode word-serial address-event link," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 7, pp. 1269–1300, 2004.

[13] T. Choi and P. Merolla et. al, "Neuromorphic implementation of orientation hypercolumns," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 6, pp. 1049–1060, 2005.

[14] P. Merolla, J. Arthur, B. Shi, and K. Boahen, "Expandable neuromorphic systems," *IEEE Trans. Circuits Syst. I*, submitted for publication.

[15] A. J. Martin, "Synthesis of asynchronous vlsi circuits," in *Formal Methods For VLSI Design*, J. Staunstrup, Ed. Amsterdam, The Netherlands: North-Holland, 1990, pp. 237–283.