FLEXIBLE USB 2.0-BASED ADDRESS-EVENT REPRESENTATION ROUTER

Matthew Carlberg
Department of Electrical Engineering
Columbia University
New York, NY 10027
mac2115@columbia.edu

Marc Goldman
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD 20742
jrtykohn@umd.edu

Research Advisor: Timothy K. Horiuchi

1. Abstract

A flexible address-event representation (AER) router system was designed, constructed, and tested for use in various research projects in the Computational Sensorimotor Systems Laboratory (University of Maryland). AER is an asynchronous communication protocol for transferring neural spikes in networked neuromorphic VLSI chip systems. By representing spikes as addresses, AER can functionally mimic the high parallel interconnectivity of neuronal systems. A CIP-51 microcontroller with two AER input ports and two AER output ports has been used to implement mapping, projecting, splitting and merging of AER spike streams. Controlled with MATLAB™ through USB 2.0, the system also allows for on-the-fly changes in functionality and real-time monitoring of input and output spikes.

2. Introduction

Nearly all neurons in the brains of animals form communication networks through the propagation of voltage spikes (or action potentials). It has long been believed that the average firing rate of voltage spikes, known as a rate code, is the means of encoding information within a neural network; however, further study has shown that the timing of a single spike can also contain significant quantities of information [1]. For this reason, many researchers have focused on spiking neural networks and the computational power they could provide. Since biological neural networks are so efficient at data processing, the study of neuromorphic VLSI, an attempt to build circuits that can mimic the “organizing principles used by the nervous system” [2], has become an important topic of bioengineering.

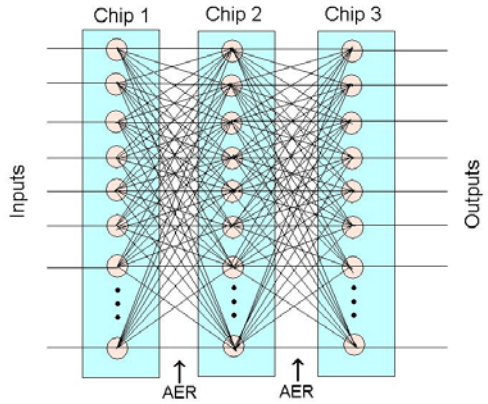

Figure 1. A Neural Network Spanning Multiple Chips: Address Event Representation (AER) protocol is used to replace this direct point to point wiring with "virtual wires".

With current silicon technology, tens of thousands of neuron equivalent circuits can be placed onto a chip; however, gate capacitance limits fan-in and fan-out to only about ten in CMOS technology. Since a single neuron in the brain can be connected to hundreds of other neurons, it is impractical to directly implement the brain's high connectivity on a silicon chip [10]. A "virtual wiring" scheme called address-event representation (AER) has been introduced for transmitting spikes between neurons [3], [4]. Instead of transmitting the actual spike, the digital identity (address) of a spiking neuron is transmitted over a single bus through the use of time-division multiplexing, mimicking the connectivity of neurons in near real-time.

AER uses an active-low, four-phase asynchronous protocol for transmitting events. Figure 2 shows how this protocol works. A sender initiates a transfer by first asserting the data bits to be sent (in this case, an address) and then pulling the request line (REQ\) low. This indicates to the receiver that new data is valid and initiates the data capture sequence.

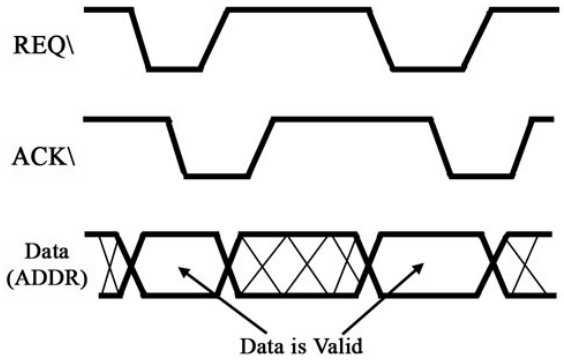

Figure 2. Asynchronous Protocol: The falling edge of the REQ\ line creates a falling edge on the ACK\ line. Data is valid while the REQ\ line is low.

When the receiver no longer needs the data lines asserted, it pulls the acknowledge line (ACK\) low. At this point, the data lines no longer have to be asserted. The sender raises REQ\, followed by the receiver raising ACK\. This sequence of events for transferring information is known as an asynchronous handshake.

AER is used extensively in projects at the Computational Sensorimotor Systems Laboratory (CSSL) at the University of Maryland. A device that can monitor events over the convenient USB interface and route events flexibly between neuron chips is anticipated to be a powerful tool in laboratory projects. It will be used for ongoing modeling work on the neural circuits in the bat echolocation system (such as the Lateral Superior Olive or the Dorsal Nucleus of the Lateral Lemniscus [5], [6]) for processing acoustic echo information for localization. In particular, the online modification feature will be used for studying the process of learning algorithms and developmental processes.

3. Project Description

An AER router that can monitor ongoing spike activity at the output of neuron chips, pass events to other chips with or without remapping, and implement complex connectivity patterns between chips is described. The router is implemented with a C8051F340 Development Kit from Silicon Laboratories™. A picture of the board is shown in Figure 3. The development kit includes a PC board containing a core CIP-51 microprocessor, a USB 2.0 compliant transceiver, a USB connector, and five 8-bit I/O ports. The CIP-51 microprocessor runs at 48 million instructions per second (MIPS) from a 48MHz clock signal (12MHZ on board oscillator with a 4X clock multiplier). Two I/O ports are used as 8-bit input addresses, two ports are used as 8-bit output addresses, and the fifth port is used for the request/acknowledge pairs required to perform asynchronous address transfers. The board also includes 256 bytes of local RAM, 4k of external RAM (XRAM), and 64k of flash memory.

The firmware of the router is programmed in C and is compiled with the PK51 Development Kit from Keil™. The router is programmed to implement the following functionalities that attempt to mimic the complex connectivity of an actual neural network: mapping, projecting, merging, and splitting.

Mapping takes one address from the output of a single neuron chip and translates it (based on a look-up table in the XRAM of the router) to an address that is passed to the input of another neuron chip. Projection takes one input address and translates it to as many as three addresses, which are transmitted

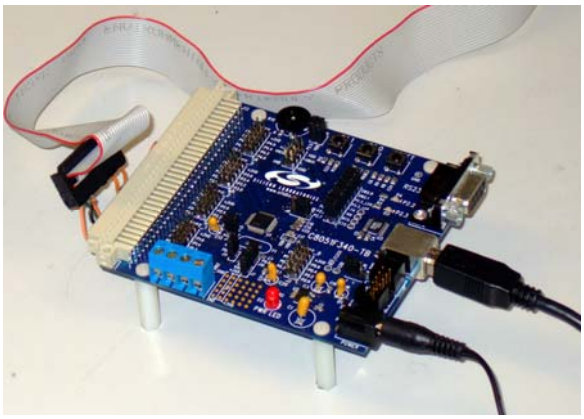

Figure 3. Overhead view of router: The input ports are at the bottom left of the device, the two output ports are in the top left. The handshake port is scene above the Red LED, the USB Port is seen at the middle of the right edge of the device. All I/O pins may be accessed from the target board connector at the left.

sequentially to the input of a single neuron chip. Merging takes addresses from two different AER ports and merges them into a single AER output port. Splitting takes one input address and translates each input event into two different addresses each sent to a separate neuron chip. These functionalities are demonstrated in Figure 4.

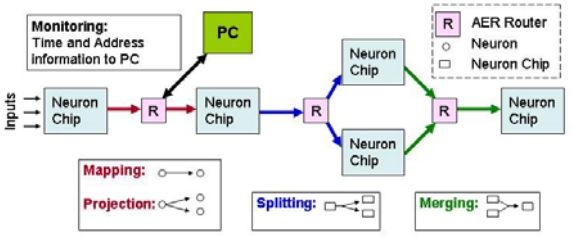

Figure 4. Demonstration of Router Functionality: Monitoring, Mapping, Projection, Splitting, and Merging are demonstrated in a neural system.

Within each of these four different biologically-inspired modes, the router can perform input monitoring. In the monitoring mode, an input address along with a 32-bit timestamp accurate

to 21 nanoseconds is stored in a buffer in XRAM. This XRAM buffer is variable in size, able to hold a maximum of 660 events. When the buffer is full, all addresses and timestamps are transferred over USB to the computer, where MATLAB™ can process the received data.

USB communication is implemented using the USBXpress™ development kit from Silicon Laboratories™. This development kit provides a set of library functions in C for the microcontroller, a Windows-based driver, and a set of library functions in Visual C++ for Windows applications that are attempting to get information from the router. The library functions assist in enumeration (the process of Windows recognizing the router) and data transfer on both the device (router) and host (computer) side. USBXpress™ fulfills the USB 2.0 specification, employing bulk mode transfers such that data is exchanged reliably at 12Mb/s (full-speed mode) through the use of error detection and numerous resending attempts [7].

MATLAB™ is used to initiate data transfers to and from the device. For USBXpress library functions to be called from MATLAB™, the use of a MEX-file is required. A MEX-file contains external code (in this case in Visual C++) that can be compiled and executed through MATLAB™. Three MEX-files were created for use by the graphical user interface (GUI) front-end to configure the device and obtain data from it. Configuration and data acquisition may also be executed from the MATLAB™ command prompt. The first MEX-file opens the device, sets its functionality (i.e. mapping, projecting, etc.), writes a 256, 412, or 768 byte look-up table to XRAM according to an input MATLAB™ matrix, and immediately closes the device. The second MEX-file opens the device, repeatedly obtains any available events for a discrete amount of time (specified by the user in a 1x1 matrix), and closes the device when the specified time is over. It outputs all timestamps and addresses obtained during the run in two matrices. Based on an input argument, the third MEX-file can: perform a single acquisition of available events from the device, change a single entry in the look-up table of the device, or close the device. It was designed for use with a Run/Stop button in the GUI and as a means of simulations of the rewiring that occurs in neural circuits during brain development. Each of these three MEX-files is used by the GUI front-end in different ways to implement the required functionality.

The GUI allows the user to interact with the router, for monitoring and loading look up tables. It is programmed using the development environment GUIDE in MATLAB™. GUIDE allows the programmer to build

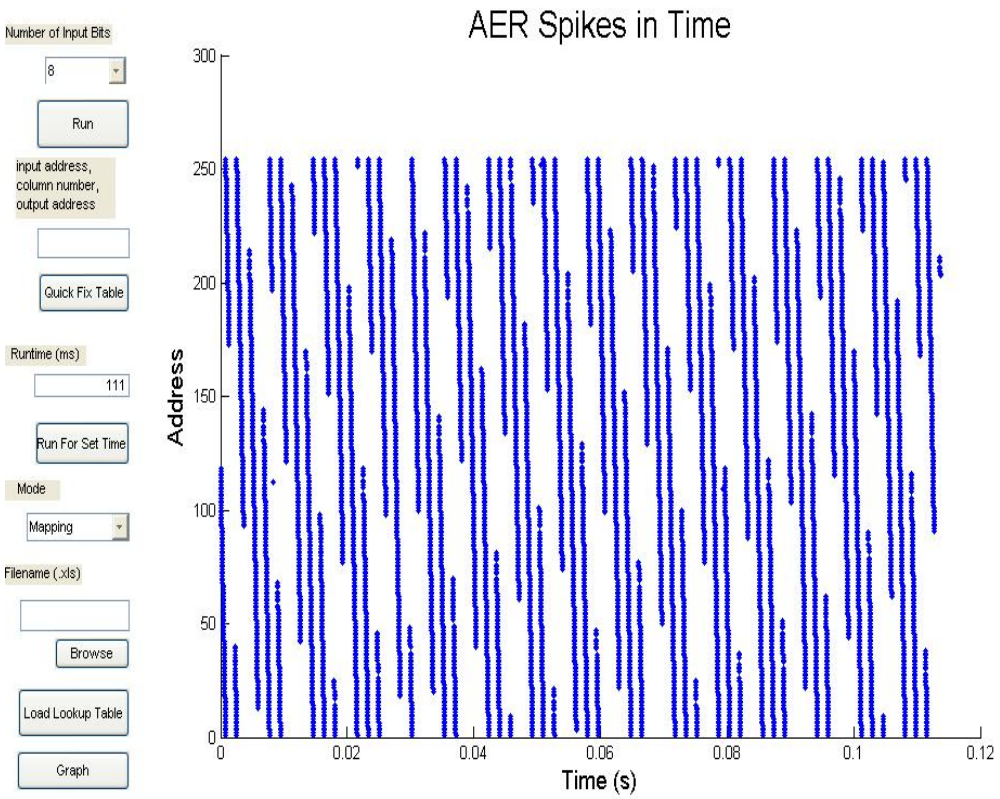

Figure 5. GUI Screenshot: The GUI allows for monitoring, graphing, and loading tables.

skeleton code based on the placement of objects (buttons, text boxes, pull down menus, etc.) in a figure. A screenshot of the GUI is shown in Figure 5.

The user has two options for monitoring—a Run-For-Set-Time and a Run/Stop function. The Run/Stop function allows the user to monitor input data for an indefinite amount of time. The Quick Fix Table function can only be accessed while in this indefinite time mode. A single value in the look-up table of the device can be modified by entering an input address, column number, and new output address. This function makes the look-up table changeable on-the-fly. The Run-For-Set-Time function allows the user to monitor AER input spikes for a duration specified in milliseconds.

Once monitoring is completed, a small amount of data processing can be performed on the data arrays. Timestamps are converted to time in seconds with an offset such that the first event occurs at 0 s. The selected value from the Number of Input Bits pull-down menu chooses the number of address bits to be plotted. Data is then graphed as a raster plot. The plot displays time in seconds on the x-axis and the corresponding address on the y-axis.

A look-up table may be loaded using the GUI from either a Microsoft™ Excel spreadsheet or a MATLAB™ matrix. When uploading a look-up table, the user chooses an option from the “Mode” pull-down menu. The selected mode and the multi-dimensional array are formatted into a one-dimensional array that is sent to the router. The router will shift from its current mode, to

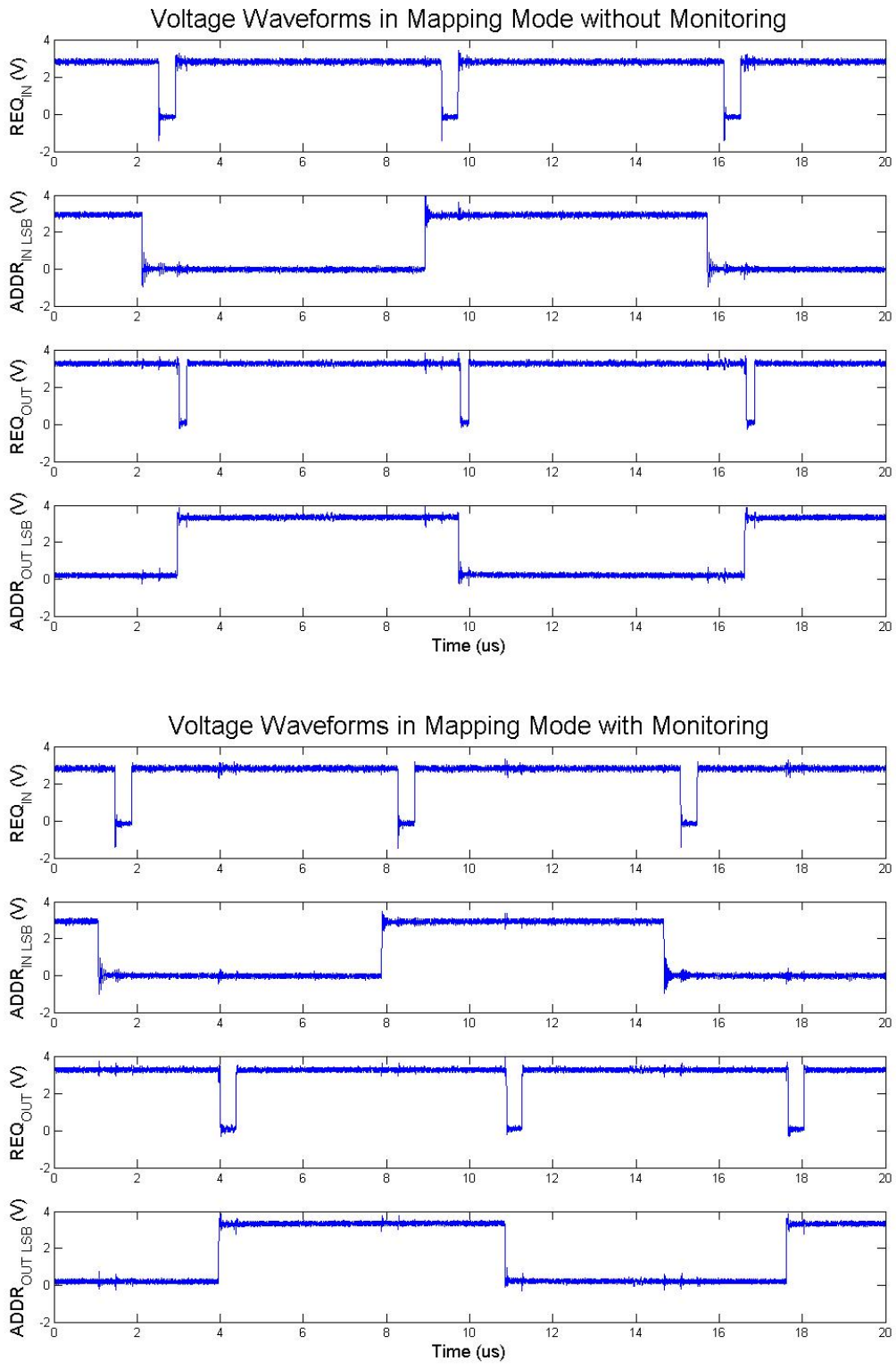

Figure 6. Mapping Mode Waveforms: Top (A) shows (i) the input request line (ii) LSB of input address (iii) the output request line and (iv) LSB of output address for three handshakes without monitoring. Bottom (B) shows the same signals (i-iv) with monitoring.

the mode specified by the new look-up table. A full look-up table cannot be uploaded when the router is actively monitoring.

4. Results

A Microchip™ PIC 18F2520 microcontroller that generates a regular pattern of AER events was used to test the router. The address of each new event is repeatedly decremented starting with address 255 and ending at 0. With this regularity, the accuracy of the router can be tested at numerous speeds. The PIC does not complete a full handshake, generating a fixed duration REQ\ signal without monitoring the ACK\ line. This introduces the possibility of missed events rather than stalled events. All figures presented were created when the PIC was generating new events every 6.8µs.

4.1 Mapping

Figure 6A shows a sequence of three input events with the router in mapping mode without monitoring. The input REQ\ line goes low at approximately 2.5µs, indicating that a valid address is available at the input port. The input address is read and an output address is determined from a look-up table on the device. Once the output address has been set, the output REQ\ is dropped at approximately 3.0µs, indicating the successful servicing of an input event within 500ns. This example illustrates an input address with 0 in the LSB mapped to an output address with 1 in the LSB. Figure 6B shows another mapping sequence when the router is also performing monitoring. The main difference in the waveforms is the time required to service each event, due to the extra time required to store new addresses and timestamps in the XRAM of the router. In this case, the input address becomes valid at 1.5µs and the output address becomes valid at 4.0µs, indicating the successful servicing of an input event within 2.5µs.

4.2 Projection

Figure 7A shows a sequence of input events with the monitor in projection mode without monitoring. In this case, the look-up table has been programmed to output zero, one, two, or three projections based on the two lowest significant bits of the input address. In projection mode, the router waits for the input handshake to be fully completed, before any projections are output. The case of three projections is the worst case timing scenario and is therefore analyzed for processing time. As depicted, an input event becomes valid at 18.1µs. The first projection is output at 19.6µs, the second at 20.1µs, and the third at 20.7µs, indicating a total service time of 2.6µs. In monitoring mode (shown in Figure 7B), an input event becomes valid at 15.9µs. The first projection is output at 19.6µs, the second at 20.2µs, and the third at 20.8µs, indicating a total service time of 4.9µs.

4.3 Splitting and Merging

Data is not yet available for splitting and merging modes. At the projects completion, these modes will be further tested and more information will be made available.

4.4 Monitoring

Two raster plots of address events versus time are shown in Figure 8A and 8B. Figure 8A has a buffer size

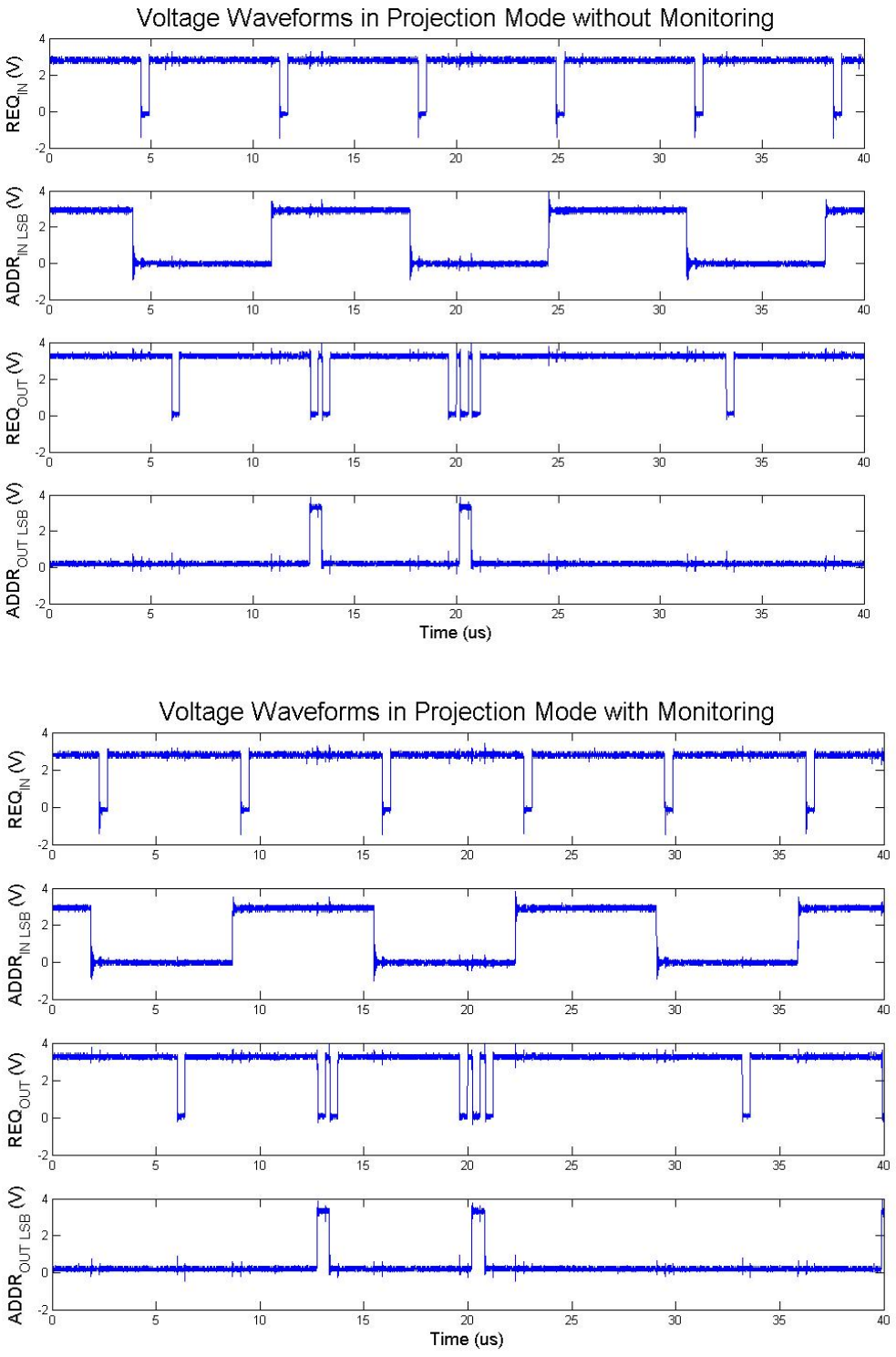

Figure 7. Projection Mode Waveforms: Top (A) shows (i) the input request line (ii) LSB of input address (iii) the output request line and (iv) LSB of output address for six handshakes without monitoring. Bottom (B) shows the same signals (i-iv) with monitoring.

set by firmware of 600 events, while Figure 8B has a buffer size of 200 events. The expected regularity of the input address events is seen in both plots. Time gaps are seen in the monitored data in both cases, due to USB transfers. This is the time interval during which a full buffer empties its contents to the computer. This transfer time depends on the buffer size used in the device as shown in Figure 8C. The minimum point on the plot corresponds to the largest USB endpoint on the device. Further explanation of the tradeoffs in buffersize is included in the "Discussion" section.

The histogram in Figure 9 shows the distribution of the time intervals between events obtained by the router. Assuming that the PIC is generating a new event exactly every 6.8μs, this plot provides information about the accuracy of the router as it monitors input events. While most events were captured at intervals of 6.8μs, a small number of events were captured late, producing a long interval and a corresponding short interval.

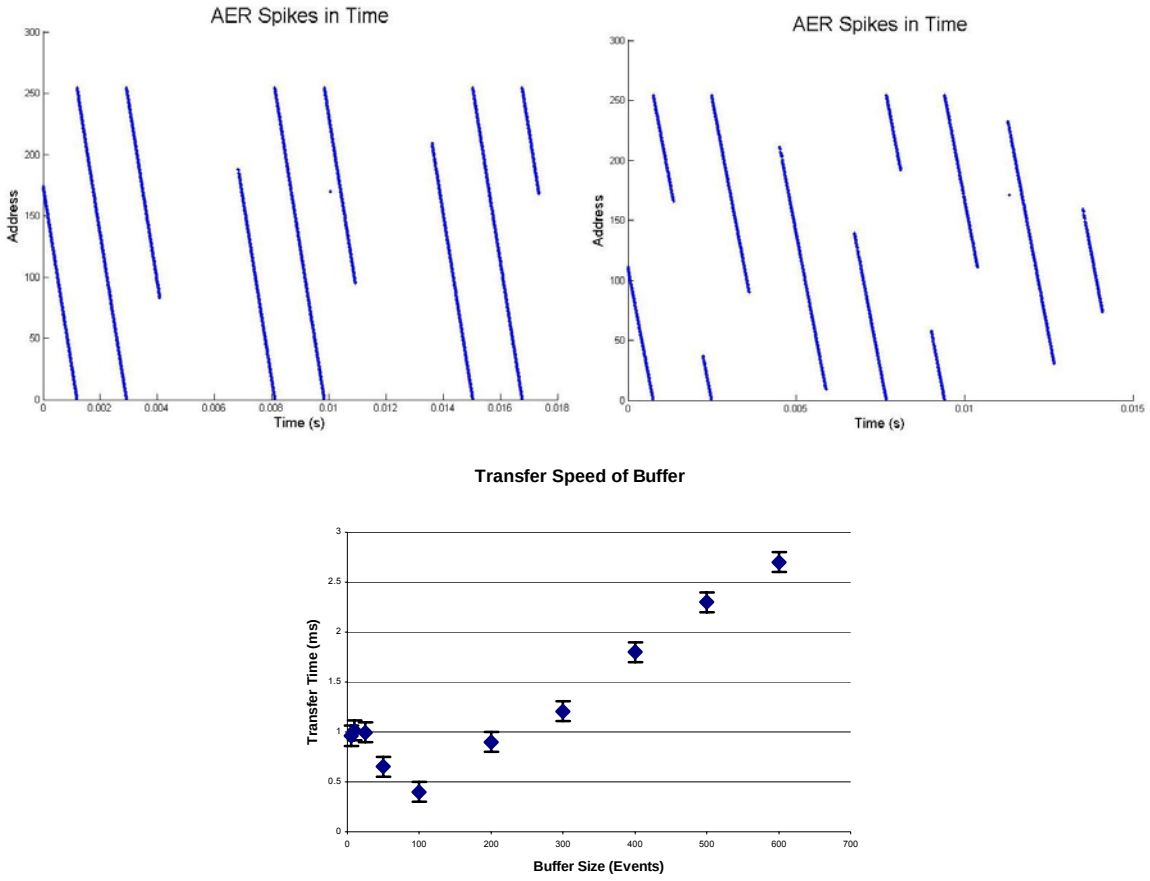

Figure 8. Effects of Changing Buffer Size: Top Left (A) Raster plot with buffer size of 600 events. Top Right (B) Raster plot with buffer size of 200 events. Bottom (C) Plot of transfer time of buffer versus buffer size.

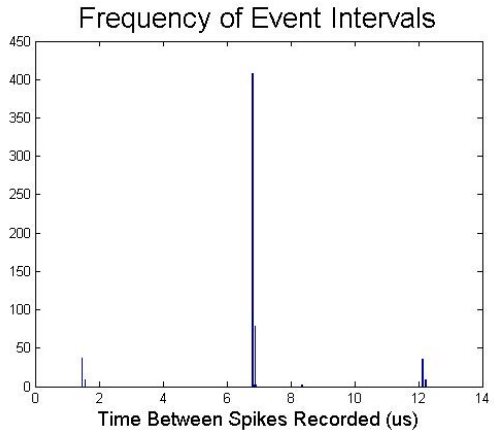

Figure 9. Frequency of Event Intervals: Shows the distribution of time between when events are detected.

5. Discussion

The router successfully implements the required monitoring and connectivity functions; however, there are a number of issues that a user must know about when attempting to use the router. Some of these issues are inherent to any microcontroller programmed for a similar purpose and some are because the router is still in the process of being developed. Examples of both are discussed next.

During USB transfers, the router is unable to detect new events. Depending on how quickly events are sent into the router, gaps can appear over time in the monitored data. As the buffer size is increased the size of the gaps are also increased, however, they occur less frequently. Based on an estimate of the frequency of incoming spikes, the best buffer size can be optimized for a particular situation. For example, if a user is interested in monitoring a burst of high frequency of events for a short period of time, a large buffer size is desirable.

In MATLAB™, there are certain tradeoffs between the Run-For-Set-Time and the Run/Stop functions. The Run-For-Set-Time function is a better choice for monitoring input events that occur regularly in the microseconds range. If this type of input is used with the Run/Stop function, the user would not be able to stop monitoring before millions of data events have been obtained causing the computer to slow down. The Run-For-Set-Time function also seems more reliable. Since the Run/Stop function repeatedly transfers computational control back and forth between compiled Visual C++ code and MATLAB™ code, the probability for missed data points appears to be greater. The Run/Stop function, however, does provide the user with an extra degree of flexibility by allowing for single location modifications in the look-up table and the ability to halt operation at any time.

Certain timing issues are also still being resolved in the device. Single spikes are missed or stalled after sending a buffer full of spikes over USB. Since the PIC test device does not actually perform a full handshake, events are missed rather than stalled. It is conceivable that USBXpress™ commands in the MEX-file that communicate with the router following a send could be responsible for causing an interrupt that could be the source of these holes in the data.

Translating clock cycles into time creates several opportunities for errors. The timer only measures from 0 to 90 s, however it does not start at zero on any particular run, so the device can wrap around from 90s to 0s anytime. The full four bytes of the timestamp cannot be stored in memory simultaneously, so it is possible that while the previous byte of a time stamp is being stored, the next byte is incrementing. These problems

are solved by taking into account the time it takes for each byte to be stored. By looking at the hexadecimal timestamp, it can be determined if such an event has occurred.

Another problem is staying aware of the voltage constraints on the router board. The router uses a 3 V supply. Using an input of 5 V can cause the board to overheat and malfunction. Properly setting the input ports and output ports in firmware is extremely important for proper functionality.

Certain routing modes also have limitations. Address 255 cannot be mapped to in Projection and Splitting modes. It is reserved for when no address should be routed to an output port. In merging mode with monitoring, the most significant bit of the timestamp is used to indicate the port on which the spike was generated. This operation is accounted for in the MATLAB™ code and does not greatly affect the operation of the circuit.

Flexibility and future improvements are an important goal of the device. Expanded look-up tables are one desirable improvement. There is not enough memory space, however, for both the event buffer and larger look-up tables in XRAM. By moving the look-up table to flash memory, more space for the look-up table will be made available and the amount of data in a block write to the computer can increase, so the gaps of data loss would be less frequent. Other uses for a larger look-up table space may include a scheme where the two input ports could be combined and the two output ports could be combined to create a 16-bit AER router mainly for single-input to single-output mapping. Additionally, multiple look-up tables could be loaded and switched between with the use of a potentiometer or jumper pin. Even without moving the look-up table to flash, one could load three mapping tables at once and access a particular one with the potentiometer. Such functionality, however, would come at the cost of reducing the number of available I/O pins.

6. Conclusion

The AER router, built in this project, will be used in circuits emulating neural networks in the Computational Sensorimotor Systems Laboratory at the University of Maryland. Researchers will be able to use the router to implement the mapping, projection, split, and merge functions. They will be able to change the look-up table using the GUI. Additionally they will be able to see what addresses are being received at what times by the monitoring function.

Acknowledgements

A special thanks to Professor Timothy K. Horiuchi our research advisor for support, guidance, and knowledge. Thanks to Rock Z. Shi, Hisham Abdalla, and Tarek Massoud of the Computational Sensorimotor Systems Laboratory. Thanks to the 2006 MERIT program and the National Science Foundation for creating such undergraduate research opportunities.

References

[1] W. Maass, “Fast Sigmoidal Networks via Spiking Neurons,” *Neural Computation*, vol. 9, pp. 279-304, 1997.

[2] C.A. Mead, “Neuromorphic Electronic Systems,” *Proc. IEEE*, vol. 78, pp. 1629-1636, 1990.
[3] M. Mahowald, “VLSI analogs of neuronal visual processing: A synthesis of form and function,” Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 1992.
[4] M. Sivilotti, “Wiring considerations in analog VLSI systems, with application to field-programmable networks,” Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 1991.
[5] R. Z. Shi and T.K. Horiuchi, “A VLSI Model of the Bat Dorsal Nucleus of the Lateral Lemniscus for Azimuthal Echolocation,” *2005 IEEE International Symposium on Circuits and Systems (ISCAS05)* , May 23-26, 2005.
[6] R. Z. Shi and T. K. Horiuchi, “A VLSI model of the bat lateral superior olive for azimuthal echolocation,” *Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS'04)*, 900–903, May 23-26, 2004.
[7] *Universal Serial Bus Specification*, Revision 2.0, 2000. <http://www.usb.org>
[8] T. Delbruck, “Simple Monitor USBXpress User Guide,” 2005. <https://svn.ini.unizh.ch/repos/avlsi/CAVIAR/wp5/USBAER/SimpleMonitorUSBXPress>
[9] MATLAB™ [computer program]. Version 7.1, Mathworks, Nantick, MA, 2005.
[10] K. A. Boahen, “Point-to-point connectivity between neuromorphic chips using address-events,” IEEE Trans. Circuits Syst. II, vol. 47, no. 5, pp.416–434, 2000.