AppNote 8 - Interfacing AER devices to SpiNNaker using an FPGA

SpiNNaker Group, School of Computer Science, University of Manchester

Luis A. Plana - 08 Mar 2017 - Version 1.2

Introduction

This note describes the implementation of an interface between Address Event Representation (AER) devices and the SpiNNaker system, using an FPGA.

AER interface

The Address Event Representation (AER) interface is an asynchronous interface that implements a 4-phase handshake protocol with active-low request ($\overline{REQ}$) and acknowledge ($\overline{ACK}$) signals. Events are transmitted in parallel, 16-bit words. The logic levels 0 and 1 are represented by voltages of 0V and 5V respectively, and most devices will operate correctly if driven by 3.3V signals. Additional information can be found in [1].

SpiNNaker link interface

The SpiNNaker interface is also an asynchronous interface but implements a 2-phase handshake protocol. This protocol, also known as Non-Return-to-Zero (NRZ), uses signal transitions, as opposed to levels, to represent data. Events are transmitted as 40-bit packets which are sent serially in 4-bit flits starting with the least significant bits. The logic levels 0 and 1 are represented by voltages of 0V and 1.8V respectively and will tolerate being driven by 2.5V and 3.3V signals. Most 2.5V devices will operate correctly when driven by SpiNNaker signals but 3.3V devices will most likely need level shifters. Additional information can be found in [2].

Mapping AER Events to SpiNNaker Multicast Packets

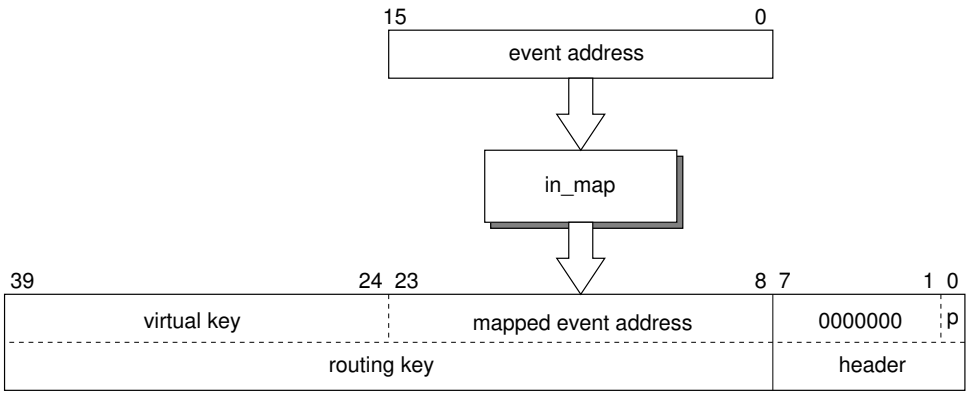

Figure 1: Mapping events to SpiNNaker packets.

As mentioned above, events in SpiNNaker are transmitted as 40-bit packets. Packets consist of a short 8-bit header, which includes a parity bit (bit[0]), and a 32-bit routing key. The routing

key is used by the SpiNNaker routers to deliver the packets to the correct destinations. As is the case for the AER devices, the routing key in most cases represents the origin of the event and not its destination or destinations.

In order to map an AER event to a SpiNNaker packet routing key, the AER device is treated as if it was a SpiNNaker chip generating events and is assigned a 16-bit *virtual key*. This virtual key must not be shared with existing chips in the SpiNNaker system. Once the virtual key is selected, a packet can be constructed as shown in Figure 1. The 16-bit event is mapped according to the characteristics of the AER device and the way it is treated in the software.

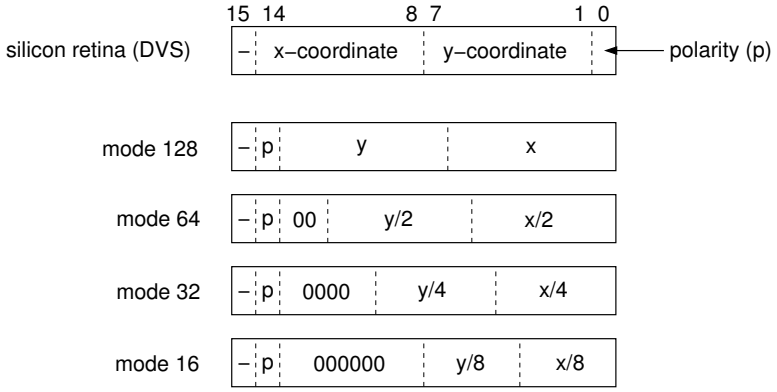

Figure 2: Silicon retina mappings.

Figure 2 shows several mappings used in the example presented below. The figure shows, at the top, the event sent by an AER silicon retina with 128x128 pixels. Four different mappings are shown below it, corresponding to 4 different sampling options: no under-sampling (128x128 pixel resolution), four neighbouring pixels grouped together (64x64 pixel resolution), groups of 16 pixels (32x32 pixel resolution) and, finally, groups of 64 pixels (16x16 pixel resolution).

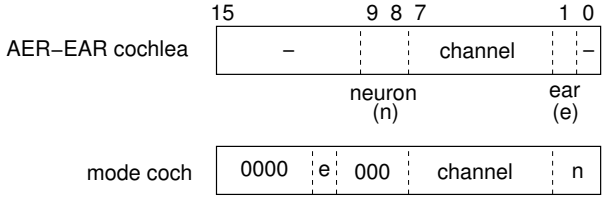

Figure 3: AER-EAR cochlea mapping.

Figure 3 shows an AER-EAR cochlea event and how it is mapped into a SpiNNaker package. The cochlea has 2 ears with 64 channels associated with each ear. Every channel contains 4 neurons. The retina and cochlea mappings have been chosen to simplify the construction of neuron populations and their placement on SpiNNaker cores.

Mapping SpiNNaker Multicast Packets to AER Events

As indicated earlier, SpiNNaker events are transported as 40-bit SpiNNaker multicast packets. The SpiNNaker packet routing key, which is used to deliver the packet to its destination, represents the origin of the event and not its destination. This key usually consists of two parts: a *virtual key* associated with the SpiNNaker chip that generated the event and an event identifier,

as shown in Figure 4. The 16-bit event identifier should be mapped according to the characteristics of the AER device and the way it is treated in the software. The most common mapping is a direct map.

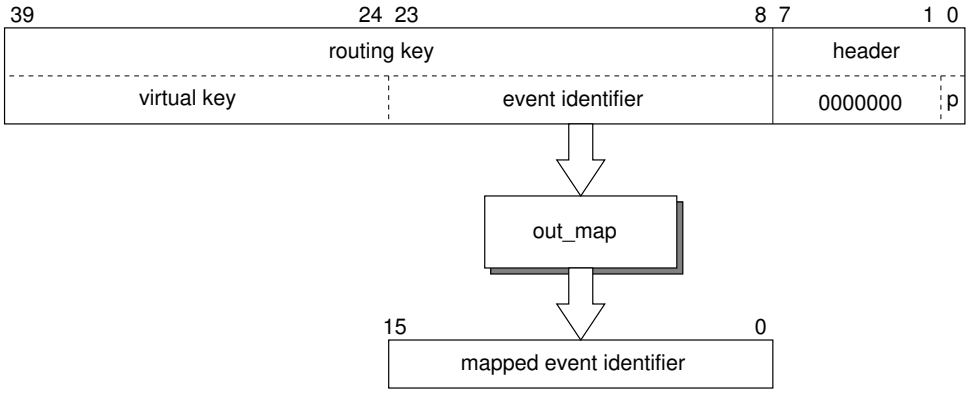

Figure 4: Mapping SpiNNaker packets to events.

Example Implementation using a Xilinx FPGA

The mappings shown above are used in a SpiNNaker–AER interface implemented on a Xilinx Spartan6 FPGA, using a RaggedStone2 board [3]. The board has two banks of I/O pins, one set at 3.3V and the other at 2.5V. The 3.3V bank is used to connect to the AER devices and the 2.5V bank is used to connect to the SpiNNaker board.

Figure 5 shows a block diagram of the implementation. The basic block in the AER-to-SpiNNaker path is the *in_mapper*, which implements the mappings shown in Figures 2 and 3 above. The constructed 40-bit SpiNNaker packet is delivered to the *dump* module which can dump the packet or pass it to the *SpiNN driver* module to be serialised and sent through a SpiNNaker link.

If at any point the SpiNNaker interface *deadlocks*, i.e., stops accepting packets, the board goes into *dump mode*. In this mode, indicated by *led4*, the board continues to accept events from the AER device but dumps them. The SpiNNaker system may need to be reset and re-booted.

Packets can also be dumped if the SpiNNaker system has not issued a *start* command to the interface to start sending packets or has issued a *stop* command to stop packet flow.

The SpiNNaker-to-AER path consists of a *SpiNN receiver* module that deserialises incoming SpiNNaker packets and passes them to the *router*, which in turn sends control packets, *i.e.*, commands, to the *control* module and AER packets to the *outmapper* which delivers the mapped event identifier to the AER device.

command	32-bit packet key	function
start	0x$----ffff$	start packet flow into SpiNNaker
stop	0x$----fffe$	stop packet flow into SpiNNaker

Table 1: Supported interface commands.

Interface commands are sent as multicast SpiNNaker packets. The router uses the packet key to detect commands and pass them to the control module. All other multicast packets are

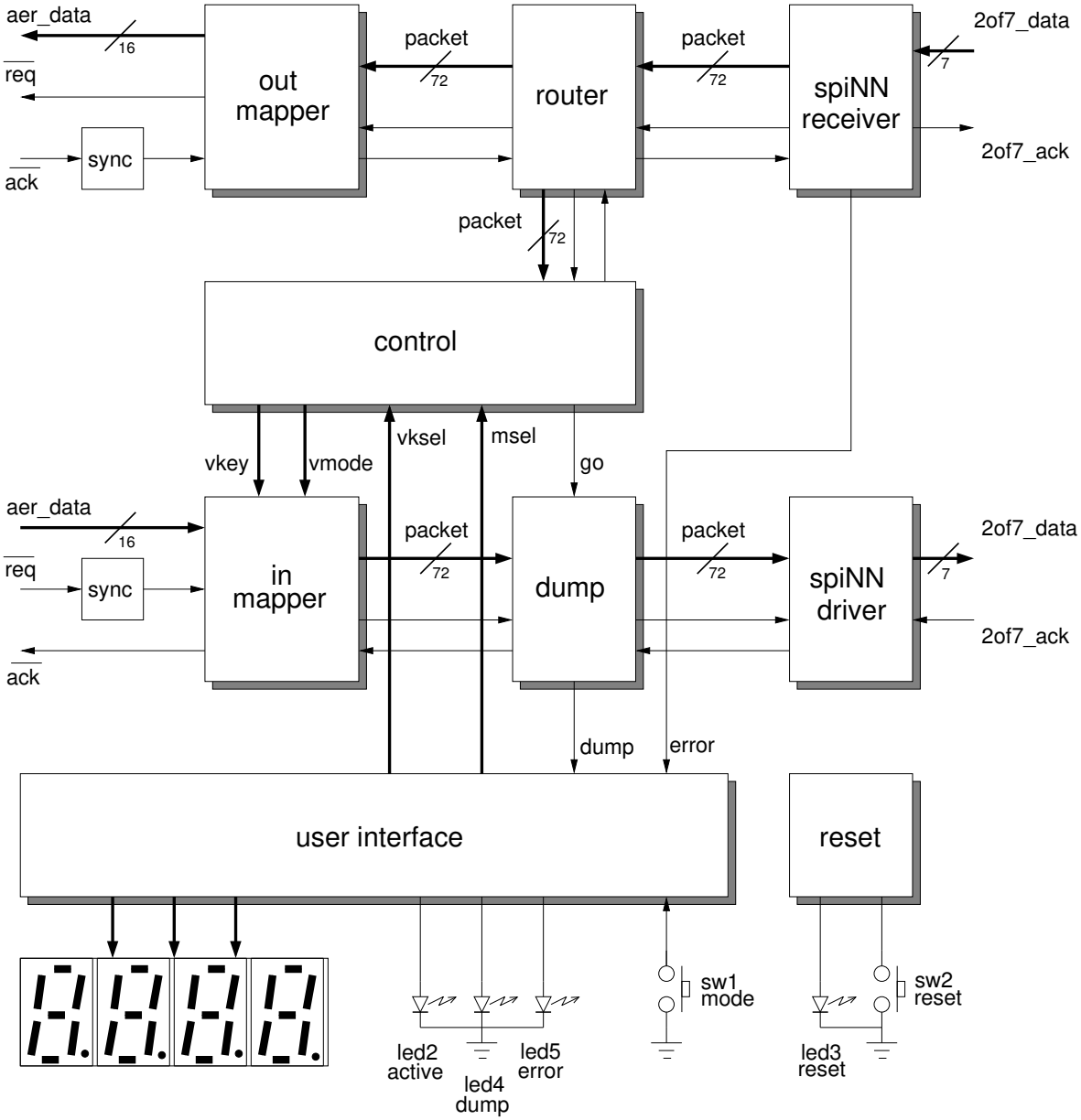

Figure 5: Interface Block Diagram.

passed to the *out_mapper* module. The router dumps all non-multicast packets. The supported commands are listed in table 1.

It is important to note the need to use of synchronisers for the incoming handshake signals in both interfaces. These asynchronous signals must be synchronised to the FPGA clock to avoid metastability. In this case, simple 2-flop synchronisers are used.

device	function
sw1	mode and virtual key selection button
sw2	reset button
led2	activity indicator (flashes continuously)
led3	reset indicator
led4	dump mode indicator
led5	error indicator
7-seg display	mode and virtual key indicator * digits show chosen mode * decimal points show chosen virtual key

Table 2: RaggedStone2 board user interface.

A *user interface* module allows the selection of the operating configuration by the user, who receives information through a 7-segment display and several leds. The functions of the buttons, leds and displays are shown in Table 2.

selection	virtual key
default	0x0200
alternate	0xfefe

Table 3: Virtual key.

As mentioned above, the AER device is assigned a virtual key as part of the mapping process. The board is configured to provide a *default* virtual key, shown in Table 3. The user can select an *alternate* virtual key, also shown in Table 3.

The *mode* button (*sw1*) is used to choose an interface configuration mode. Each configuration mode is a combination of an input map and a virtual key. The available configuration modes are shown in Table 4. The user can step through the different modes by repeatedly pushing the *mode* button. All input maps discussed above are included and there is also a *direct* input map which passes the 16-bit AER event directly to the packet, without any re-structuring. The selected configuration mode is shown on the 7 segment displays, as shown in Table 4. Note that the decimal points are used to indicate the chosen virtual key.

Implementation Code

The interface was implemented using Verilog. The code is part of **spI/O** [4], a library of FPGA designs and re-usable modules for I/O in SpiNNaker systems. **spI/O** is available as a SpiNNakerManchester GitHub repository (https://github.com/SpiNNakerManchester/spio).

mode	display	description
def_retina_128	128	default key & retina with 128x128 pixels
def_retina_64	64	default key & retina sub-sampled to 64x64 pixels
def_retina_32	32	default key & retina sub-sampled to 32x32 pixels
def_retina_16	16	default key & retina sub-sampled to 16x16 pixels
def_cochlea	coch	default key & cochlea: 2 ears/64 channels/4 neurons
def_direct	0000	default key & event passed through without mapping
alt_retina_128	.128	alternate key & retina with 128x128 pixels
alt_retina_64	. 64	alternate key & retina sub-sampled to 64x64 pixels
alt_retina_32	. 32	alternate key & retina sub-sampled to 32x32 pixels
alt_retina_16	. 16	alternate key & retina sub-sampled to 16x16 pixels
alt_cochlea	c.och	alternate key & cochlea: 2 ears/64 channels/4 neurons
alt_direct	0.000	alternate key & event passed through without mapping

Table 4: Interface configuration modes.

References

[1] P. Häfliger, "CAVIAR Hardware Interface Standards", Version 2.0, Deliverable D WP7.1b, online: http://www2.imse-cnm.csic.es/caviar/download/ConsortiumStandards.pdf, 2003.

[2] S. Temple, "AppNote 7 - SpiNNaker Links", Version 1.0, online: http://solem.cs.man.ac.uk/documentation/spinn-app-7.pdf, 2012.

[3] Enterpoint Ltd., "RaggedStone2 User Manual", Issue 1.02, online: http://www.enterpoint.co.uk/raggedstone/RaggedStone_User_Manual_Issue_1_02.pdf, 2010.

[4] L.A. Plana, J. Heathcote, J. Pepper, S. Davidson, J. Garside, S. Temple, and S. Furber, "spI/O: A library of FPGA designs and reusable modules for I/O in SpiNNaker systems." May 2014. [Online]. Available: `https://doi.org/10.5281/zenodo.51476`

Change log:

- *1.0 - 17apr13 - lap* - initial release - comments to *luis.plana@manchester.ac.uk*
- *1.1 - 03apr14 - lap* - bidirectional interface - comments to *luis.plana@manchester.ac.uk*
- *1.2 - 08mar17 - lap* - router and control modules - comments to *luis.plana@manchester.ac.uk*