Universität Heidelberg

Electrinic Vision(s) Group
Project Internship Report

A network accessible system controller for BrainScaleS-2

Maximilian Stucke

supervised by
Yannik Stradmann, Joscha Ilmberger

June 28, 2022

Abstract

The BrainScaleS-2 (BSS-2) Cube system is a highly configurable neuromorphic computing setup, consisting of neuromorphic ASICs and up to four FPGAs, that play a crucial role in hardware stability. In the current state, FPGAs are power cicled through a USB bussed microcontroller. This solution is not stable, nor user friendly and does not scale well. Power management systems that are accessible through the network offer more flexibility and stability to the user and scale better with larger setups.
A Raspberry Pi was added to the system and the existing BrainScaleS-1 (BSS-1) power management solution sw-macu was modified to detect BSS-2 Cube systems by identifying the Raspberry Pi IPv4 address in the hardware database. The Raspberry Pi provides a JTAG interface to the FPGAs, enabling direct network control and configuration via a XVC server.
The setup was found to work reliably while greatly improving accessibility and system stability.

Contents

1 Introduction **3**

2 Implementations **5**

2.1 Legacy python control 5

2.2 SysCtrl 5

2.2.1 Adaptations 7

2.2.2 Build environment 9

2.3 Xilinx Virtual Cable 9

1 Introduction

BrainScaleS-2 (BSS-2) systems are complex neuromorphic setups, consisting of neuromorphic ASICs and multiple FPGAs. These FPGAs play a crucial role in hardware stability and are key to enable stable experimentation on this platform. This work focuses on BSS-2 *Cube* systems, that incorporate a single neuromorphic ASIC and up to four FPGAs. These FPGAs can be manually powered through a *UCD9246* [Ins22] power management chip via I^2C. Currently, this is handled through a microcontroller in the setup, interfaced by a USB bus. This approach is unstable and does not scale for larger setups. Hence, a network approach through TCP is better suited to improve stability, avert scaling issues and impove user accessibility.
Since the microcontroller lacks this ability, it is omitted and replaced by a Raspberry Pi with network access. This work focuses mainly on adapting the existing *sw-macu* repository, the current BSS-1 solution, to fit the needs discussed above.

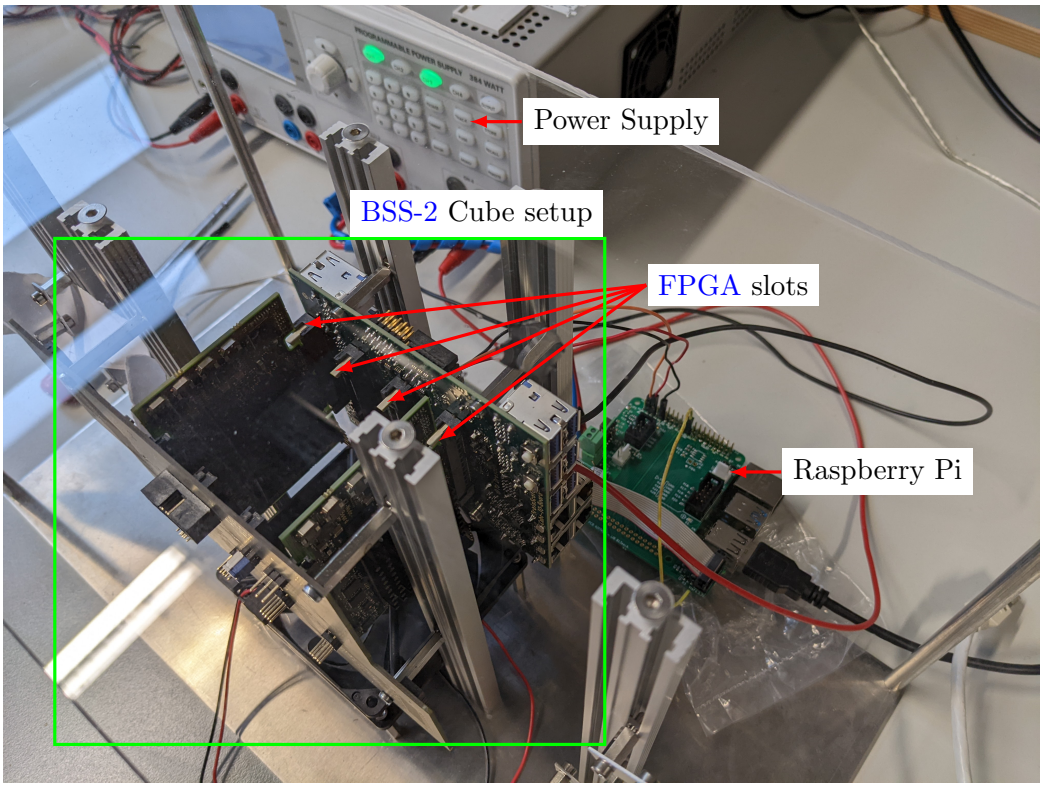

Figure 1: Typical BrainScaleS-2 Cube setup

The *sw-macu* repository includes server- and client-side software, that enables network communication through the Remote Call Framework (RCF) [Del22b]. The repository contains three different modules. The *SysCtrl* server runs on the Raspberry Pi and handles communication with *SysStatus* and *reticleCtrl*, which run on a cluster machine in the network, as well as I^2C communication with the power management IC..

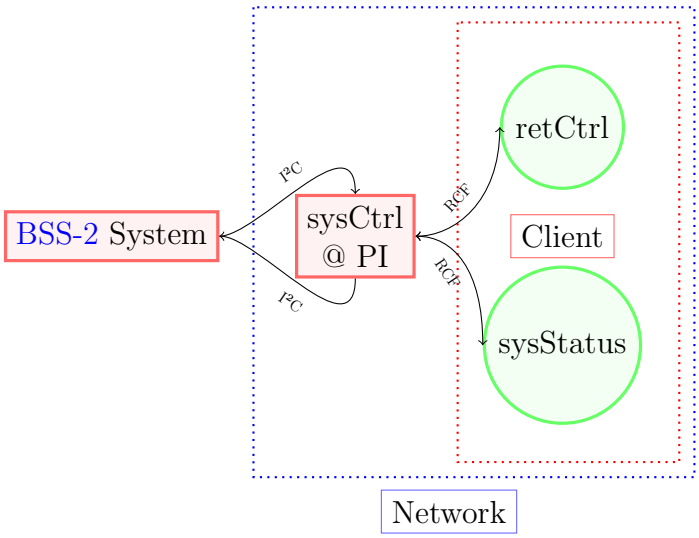

Figure 2: System interface diagram

The Raspberry Pi enables further connectivity to BrainScaleS-2 systems. It is able to provide a generic JTAG interface to the FPGAs, that can then be used by Vivado [Xil22] through a *Xilinx Virtual Cable* (XVC) [der22] server. This enables network control and configuration of the FPGAs, improving user accessibility and general hardware stability.

2 Implementations

2.1 Legacy python control

The *sw-macu* repository also contains python scripts, that can be executed directly on the Raspberry Pi to control the power management IC as well. As a first step, the functionality of this code was tested. It has to be noted, that this is legacy code and not intended for live usage anymore.

```
python fpga_ctrl.py --wafer 4 --fpga 1 --power 1
```

Listing 1: Manual powering of an FPGA through I^2C

As a consequence, FPGA no. 1 was powered up. This script works by directly writing to the power management chip via I^2C. To further simplify this, consider the snippet in listing 2.

```
import smbus
import time
# I2C bus on Raspberry Pi
DEVICE_BUS = 1
# Address of power IC
DEVICE_ADDR = 0x68
bus = smbus.SMBus(DEVICE_BUS}
# Write to power FPGA (FF for on)
bus.write_byte_data(DEVICE_ADDR,0,FF)
```

Listing 2: Minimal python script to power FPGA

2.2 SysCtrl

The SysCtrl server runs on the Raspberry Pi and can be interfaced through RCF by a client on the network, e.g. by a user on the cluster frontend. The software has implemented functionality, such as temperature and voltage monitoring, that are not present on current BrainScaleS-2 Cube systems. The code is written in a way, that uninitialized features can cause segmentation faults, as well as verbose logging errors. Corresponding object are not created, despite their pointers getting called in other functions.
Consider an example from *funcs.cpp* in listing 3.

```
...
LOG(INFO) << "Add Power:";
for (YAML::iterator it = yamlNode["power"].begin();
        it != yamlNode["power"].end(); ++it) {
        ...
        if (type.compare("powerit") == 0) {
                ...
                else if (pitVersion == 2) {
                        LOG(INFO) << "use powerit_v2 class";
                        (sys.power)->mainpower = new powerit_v2(
                        i2cPowerItBus,
                        i2cPowerItAddr,
                        sys.i2cMutexes["power"],
                        kibLogptr);
                        ...
                }
                ...
        }
        ...
}
```

Listing 3: readPower function from funcs.cpp

BrainScaleS-2 Cube systems do not have a dedicated power board and hence no additional power information that is supplied in the yaml file. If this node is not read, the *mainpower* object is never created. However, this module gets called in other functions by default during runtime, raising a segmentation fault.
To avert this issue, sysCtrl is built with the hardware-database *hwdb* as a dependency, allowing to query for system information. The local IPv4 address of the Raspberry Pi is compared to the database. If the address corresponds to a BrainScaleS-2 Cube system, erronious routines are skipped. Additionally, a register inside the FPGA is set via I^2C, containing the so-called *wafer-id* that defines the physical setup. This information is used by the FPGA to determine its IP address for network communication.

2.2.1 Adaptations

The BrainScaleS-1 solution depends on RCF 2.2. In the current iteration, RCF was migrated to version 3.2 to prevent maintaining multiple branches [Del22a]. To enable system detection, a *SystemInfo* class was added to *SysCtrl*.

```
class SystemInfo {
public:
        halco::hicann::v2::IPv4* local_ip =
                new halco::hicann::v2::IPv4;
        int* wafer_id = new int;

        //Constructor
        SystemInfo();
        //Destructor
        ~SystemInfo();

private:
        //get local ip of macu
        void getLocalIp();
        // get wafer id from hwdb
        void getWaferId();
};
```

Listing 4: SystemInfo class header

The function *getLocalIp* queries the IPv4 address of the Raspberry Pi on the current network interface, *getWaferId* then queries the hardware database for all available system information and compares the previously obtained IPv4 address. When a match has been found, the corresponding system type and wafer-id are known.

```
void SystemInfo::getWaferId() {
...
        db.load(db.get_default_path());
        wafers = db.get_wafer_coordinates();
        for(auto & wafer : wafers) {
                hwdb4cpp::WaferEntry entry;
                entry = db.get_wafer_entry(wafer);
                if(*addr == entry.macu) {
                        *wafer_id = wafer.toEnum();
                 }
                else {
                        continue;
                }
...
}
```

Listing 5: Implementation of *getWaferId*

The results are saved in public member variables, as defined in the *public* section of listing 4.

2.2.2 Build environment

One needs to differentiate between client and server targets. The former are always run on x86 hardware and can be build with the latest visionary container. SysCtrl is run on the Raspberry Pi and therefore needs to be cross compiled with a new container.
To have a reproducible way of meeting all package requirements of *hwdb* and *sysCtrl*, containerization is best. To manage different versions of packages that are required, a modifed version of the package manager *Spack* [Gam+19] was used to bootstrap the required packages into a *Singularity* [ncc22] container.
For *hwdb*, all package requirements can be met by installing the *visionary-dls* spack-package. The main challenge lies with satisfying dependencies for sysCtrl, since the previously used containers do not have any available definition files. Hence, needed packages and their versions had to be discovered manually.
From the resulting package list, a spack-package was created. The deployed containers are listed in table 1.

Container	host arch	target arch
Debian11-x86-cross.sif	x86_64	x86_64/aarch64
Debian11-aarch64.sif	aarch64	N/A

Table 1: Containers

2.3 Xilinx Virtual Cable

Xilinx Virtual Cable (XVC) is a protocoll, based on TCP/IP, that enables the communication to many SoCs, e.g. FPGAs. The Raspberry Pi provides a JTAG-interface through bit-banging its GPIO pins, that can then be accessed through *Vivado* from any cluster machine, by running a XVC server on the system.

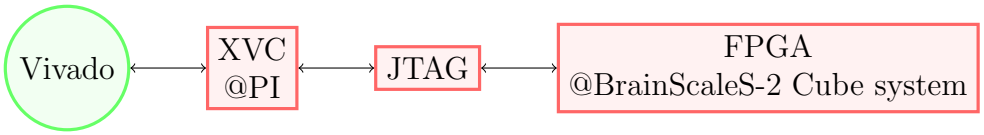

Figure 3: XVC interface

One of the issues encontered with this setup were signal inconsistencies, present within the JTAG-interface. The TDO signal, that is send from the GPIO pins of the Raspberry Pi passes through a logic level shifter present on a HAT board. The resulting signal was observed as faulty, as its amplitude is below the needed 3.3 V threshold. A bad solder connection was identified as the cause of this problem. Upon rectifying the connection, a usable TDO signal was observed.

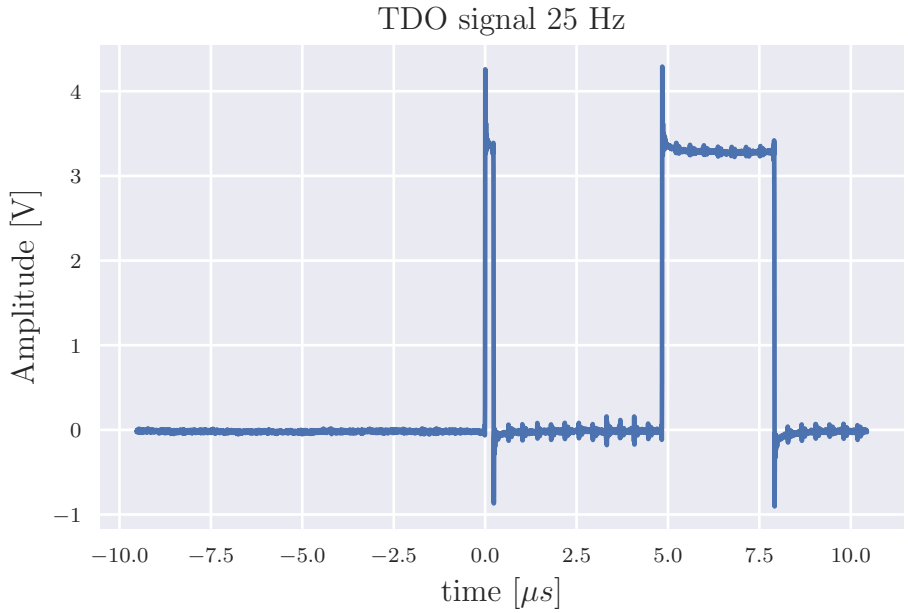

Figure 4: TDO signal comparison

As illustrated in figure 4, a valid signal was obtained. Due to inconsistent timing of the Raspberry Pi, one should adhere to minimum JTAG delay of 100 µs
With this infrastructure, direct communication between FPGAs on BrainScaleS-2 systems and any networked host on the cluster is possible, making it very easy to flash bitfiles, read FPGA information, etc.

Acronyms

ASIC Application-specific integrated circuit. 1, 3

BSS-1 BrainScaleS-1. 1, 3

BSS-2 BrainScaleS-2. 1, 3, 4

FPGA Field Programmable Gate Array. 1, 3, 4, 6

XVC Xilinx Virtual Cable. 1

Glossary

sw-macu BrainScaleS-1 system monitoring solution. 1

References

[Gam+19] Todd Gamblin et al. “Spack Community BoF”. In: *ISC High Performance 2019*. June 2019.

[Del22a] Deltavsoft. *Migrating RCF 2.2 to RCF 3.0*. Apr. 6, 2022. URL: https://www.deltavsoft.com/doc/_migration.html.

[Del22b] Deltavsoft. *Remote Call Framework*. Version 3.2. Apr. 6, 2022. URL: http://deltavsoft.com.

[der22] derekmulcahy. *xvcpi*. https://github.com/derekmulcahy/xvcpi. 2022.

[Ins22] Texas Instruments. *UCD9246*. Apr. 6, 2022. URL: https://www.ti.com/product/UCD9246?utm_source=google&utm_medium=cpc&utm_campaign=app-null-null-GPN_EN-cpc-pf-google-eu&utm_content=UCD9246&ds_k=UCD9246&DCM=yes&gclid=EAIaIQobChMIvrvonJ2O9wIVFobVCh1B5wHrEAAYASAAEgJsL_D_BwE&gclsrc=aw.ds.

[ncc22] nccgroup. *Singularity*. https://github.com/nccgroup/singularity. 2022.

[Xil22] Xilinx. *Vivado*. Version ML. Apr. 6, 2022. URL: https://www.xilinx.com/products/design-tools/vivado.html.