

Extending Compiler Support for the BrainScaleS Plasticity Processor

Bachelor's Thesis Presentation

Arthur Heimbrecht

March 14, 2017

Contents

PPU Architecture

GCC Structure

Extending GCC for the PPU

Results

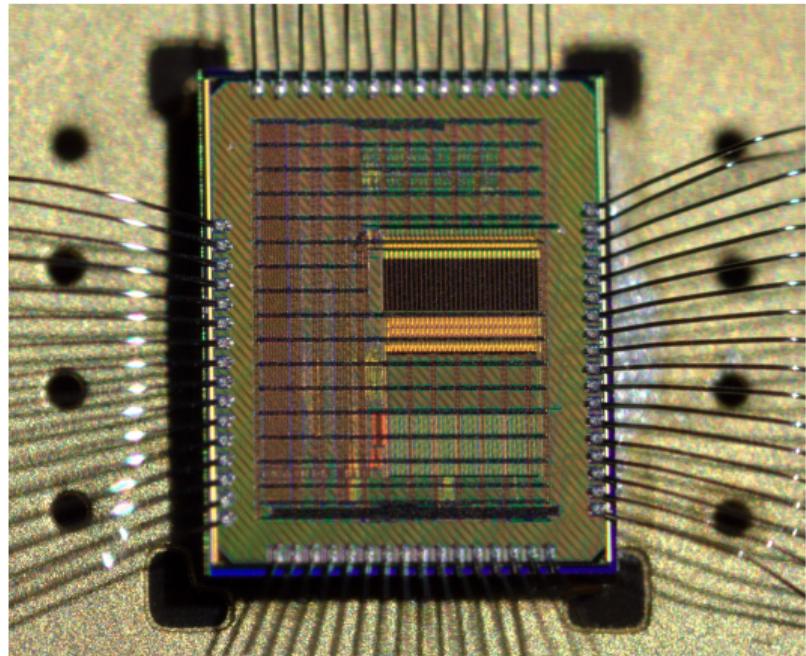


Figure 1: Photograph of HICANN-DLS chip, Friedmann et al.

HICANN-DLS

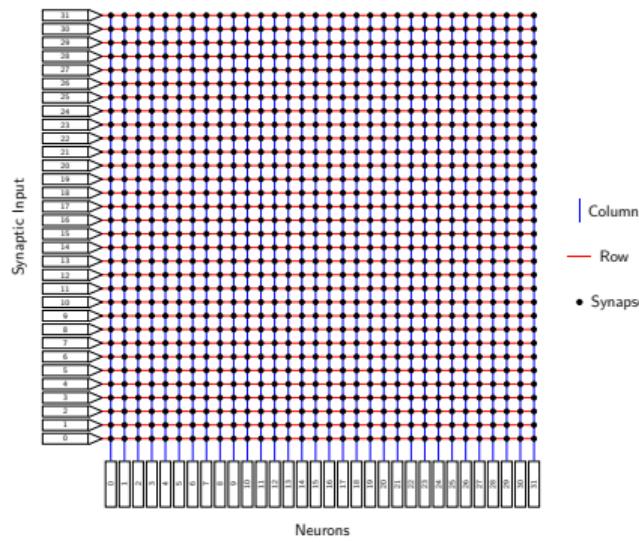


Figure 2: Schematic Representation of the Synapse Array on HICANN-DLS

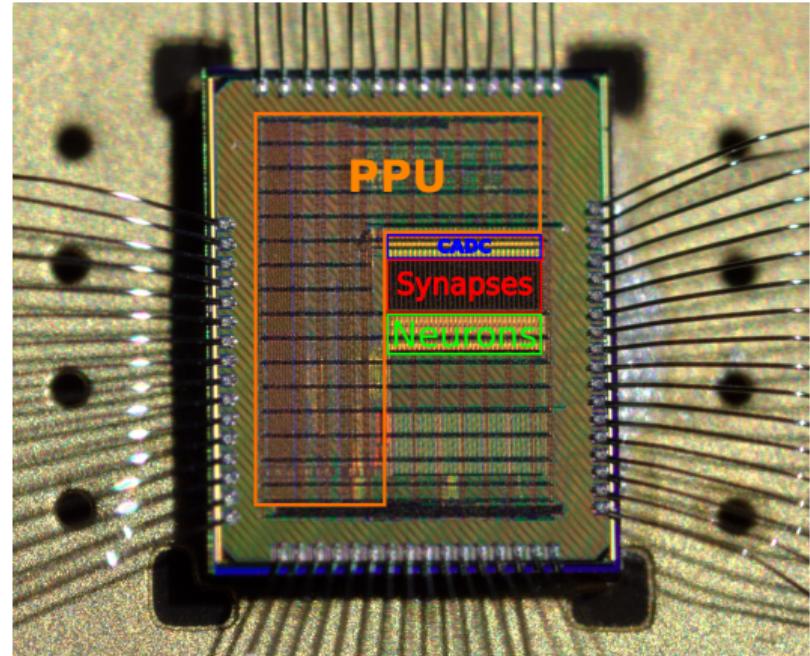


Figure 3: Photograph of the HICANN-DLS Chip, modified from Friedmann et al.

Plasticity Processing Unit

“Common” von-Neumann architecture

Machine Instructions

Arithmetic Logic Unit (ALU)

$\text{latency}(\text{register}) \ll \text{latency}(\text{memory})$

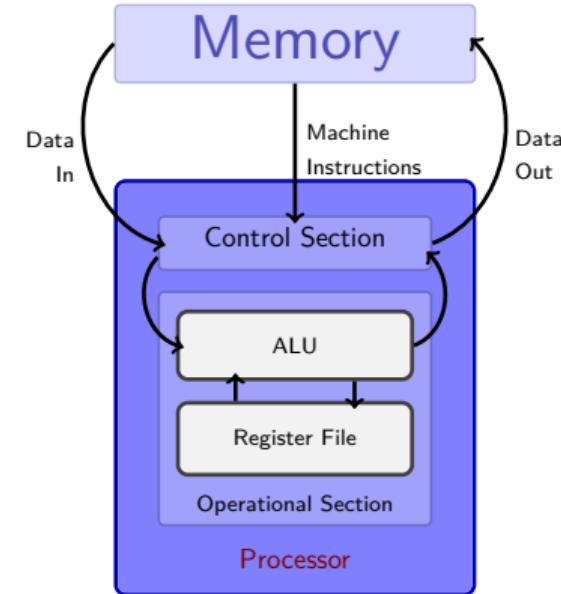


Figure 4: Schematic of General Purpose Processor with von-Neumann Architecture

PPU Architecture

Based on POWER architecture

Vector Extension

- Parallelization
- Performance

Weight Updates

Access to Synaptic Array

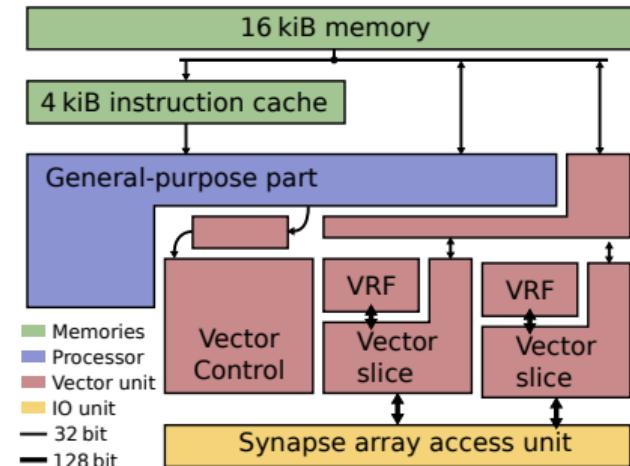


Figure 5: Structure of nux Architecture

Compiler Structure

Program $\xrightarrow{\text{compile}}$ Executable

Compiling Stages

Compiler \rightarrow Assembly

Back-End is target-dependent

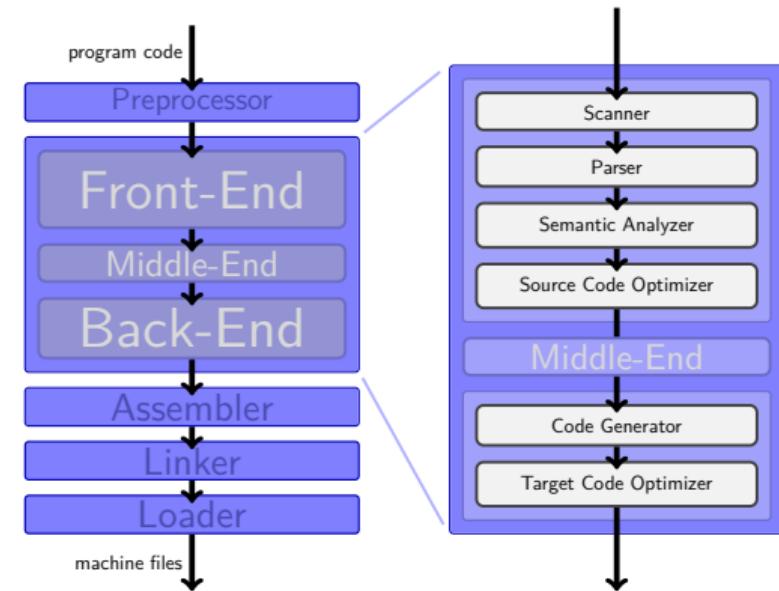


Figure 6: Structure of Compiling Process and Compiler

GCC for the PPU

No “official” support of s2pp

Currently using macros

listing of PPU code

Existing AltiVec extension

Figure 7: Structure of Compiling Process and Compiler

Intrinsics → **use this**

Main Work of this Thesis

No “official” support of s2pp

Currently using macros

listing of PPU code

Existing AltiVec extension

Figure 8: Structure of Compiling Process and Compiler

Intrinsics → **use this**

New Features

-mcpu=nux target flag

vector attribute

s2pp vector intrinsics

Inline assembly

Supports optimization

Outlook

Further testing for bugs

Existing applications with `libnux`

Extending test coverage

Debugging support?

New tool for development!

```
1 void measure_correlation( uint32_t ca_hist[],  
2                           uint32_t ac_hist[],  
3                           vector<uint8_t> ca_offsets[],  
4                           vector<uint8_t> ac_offsets[])  
5 {  
6     vector<uint8_t> select = fxv_splatb(reset_ca | reset_ac);  
7     vector<uint8_t> ca_meas;  
8     vector<uint8_t> ac_meas;  
9     for (uint32_t index = 0; index < dls_num_synapse_vectors; index++) {  
10         vector<uint8_t> ca_offset = ca_offsets[index];  
11         vector<uint8_t> ac_offset = ac_offsets[index];  
12         asm volatile (  
13             // Load causal and acausal  
14             "fxvinx %[ca_meas], %[ca_base], %[index]\n"  
15             "fxvinx %[ac_meas], %[ac_base], %[index]\n"  
16             "fxvshb %[ca_meas], %[ca_meas], -1\n"  
17             "fxvshb %[ac_meas], %[ac_meas], -1\n"  
18             // Subtract offsets  
19             "fxvsubbf %[ca_meas], %[ca_meas], %[ca_offset]\n"  
20             "fxvsubbf %[ac_meas], %[ac_meas], %[ac_offset]\n"  
21             // Reset correlation measurement  
22             "fxvoutx %[select], %[ca_base], %[index]\n"  
23             : [ca_meas] "+kv" (ca_meas),  
24               [ac_meas] "+kv" (ac_meas)  
25             : [ca_base] "r" (dls_causal_base),  
26               [ac_base] "r" (dls_acausal_base),  
27               [index] "r" (index),  
28               [ca_offset] "kv" (ca_offset),  
29               [ac_offset] "kv" (ac_offset),  
30               [select] "kv" (select);  
31         vector<uint8_t> weights = fxv_in(dls_weight_base, (uint8_t*)(index));  
32         ...
```

References

- S. Friedmann, J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier, "Demonstrating hybrid learning in a flexible neuromorphic hardware system", (2016).
- T. Flik, *Mikroprozessortechnik und rechnerstrukturen*, ger, 7., neu bearb. Aufl. (Springer, Berlin ; Heidelberg [u.a.], 2005), XIV, 649 S.
- K. D. Cooper and L. Torczon, *Engineering a compiler*, eng, 2. ed., Previous ed.: 2004 (Elsevier, Amsterdam ; Heidelberg [u.a.], 2012), XXIII, 800 S.
- A. V. Aho, *Compiler, Prinzipien, techniken und werkzeuge*, ger, 2., aktualisierte Aufl., it Informatik, Index S. 1227-1253 (Pearson Studium, München [u.a.], 2008), XXXVI, 1253 S.
- S. Friedmann, *Nux manual*, (2016).
- *Gnu compiler collection internals manual*,
<https://gcc.gnu.org/onlinedocs/gccint/index.html>, Free Software Foundation Inc. (2017).

Whiteboard

7	6	5	4	3	2	1	0
			2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
7	6	5	4	3	2	1	0
			2^5	2^4	2^3	2^2	2^1
7	6	5	4	3	2	1	0
-1			2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
							2^{-7}

Figure 9: Comparison of the Representation of Weights in Synapses and the Fractional Representation of Vector Components for Fixed-Point Saturation Arithmetic

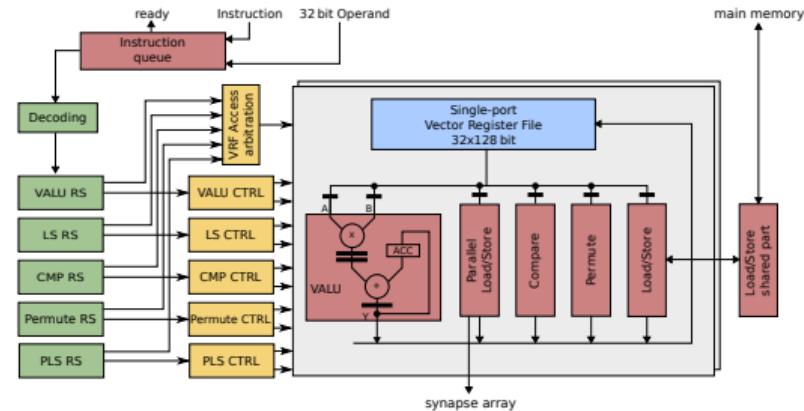
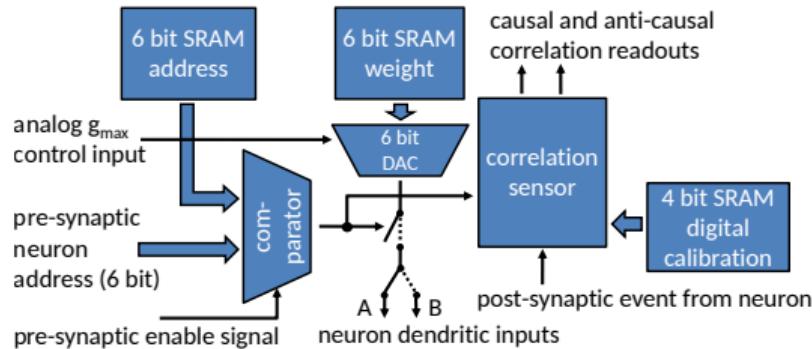
PPU is the processor which is part of HICANN-DLS and mainly responsible for plasticity.

nux refers to the architecture of the PPU, see figure.

s2pp describes the PPU's VE and is part of the nux architecture.

Appendix

Additional Figures



	31	23	15	7	0
mnemonic	operand	operand	operand		
addi	r1 register address	r2 register address	5 immediate operand		

Figure 11: Representation of Assembly Instruction addi as a Machine Instruction in Memory. The immediate value 5 is added to register r2 and the result written in r1.

Additional Figures

		0	7	15									31									63									127
QI		Quarter Integer																													
HI		Half Integer																													
SI														Single Integer																	
SF														Single Float																	

Figure 13: Vector structures are 128 bits wide and split into common word sizes.

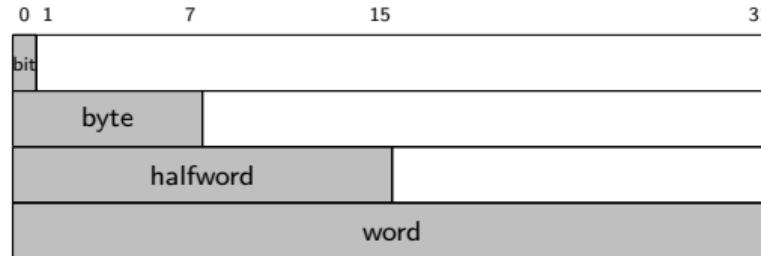


Figure 14: Illustration of Word Sizes for 32-bit Words