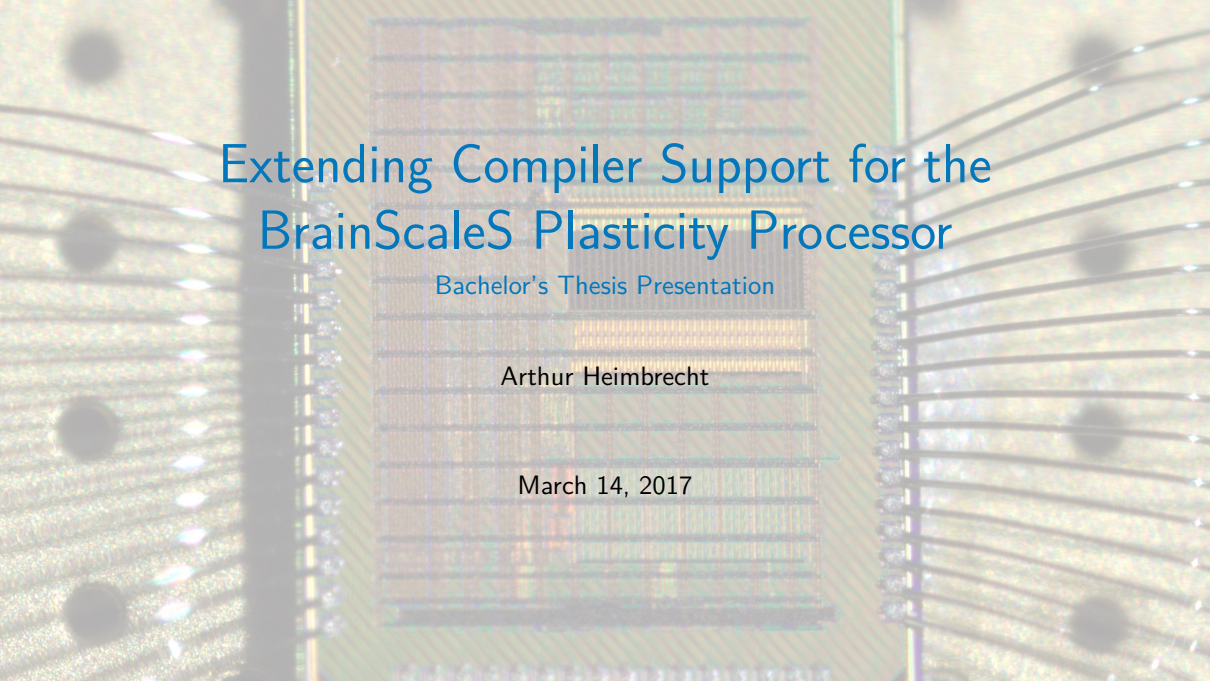

Extending Compiler Support for the BrainScaleS Plasticity Processor

Bachelor's Thesis Presentation

Arthur Heimbrecht

March 14, 2017

Contents

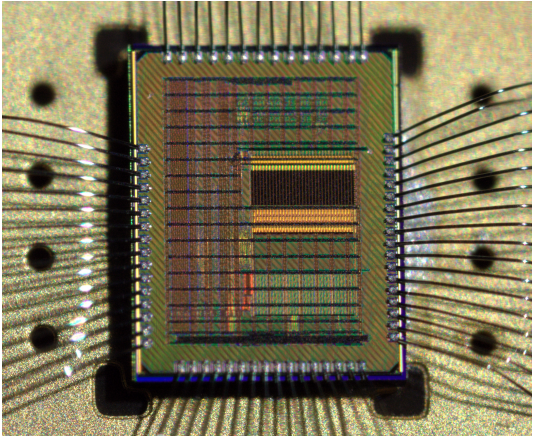

Figure 1: Photograph of HICANN-DLS chip, Friedmann et al.

HICANN-DLS

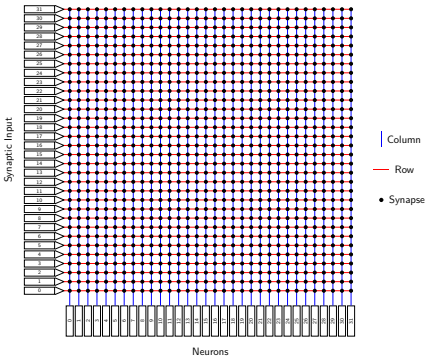

Figure 2: Schematic Representation of the Synapse Array on HICANN-DLS

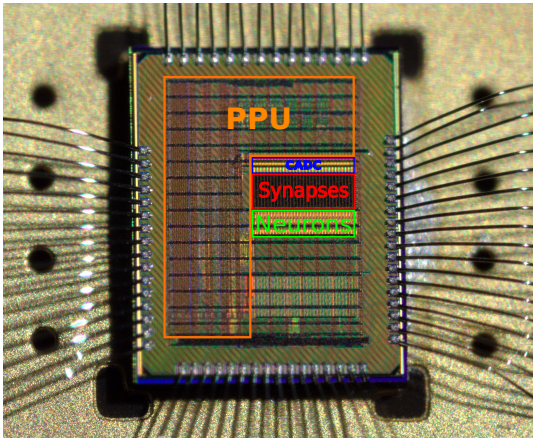

Figure 3: Photograph of the HICANN-DLS Chip, modified from Friedmann et al.

Plasticity Processing Unit

- “Common” von-Neumann architecture
- Machine Instructions
- Arithmetic Logic Unit (ALU)

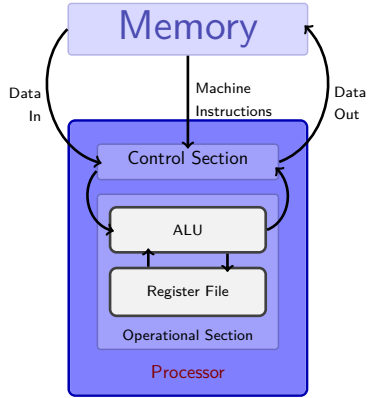

Figure 4: Schematic of General Purpose Processor with von-Neumann Architecture

Plasticity Processing Unit

- "Common" von-Neumann architecture
- Machine Instructions
- Arithmetic Logic Unit (ALU)
- $\text{latency}(\text{register}) \ll \text{latency}(\text{memory})$

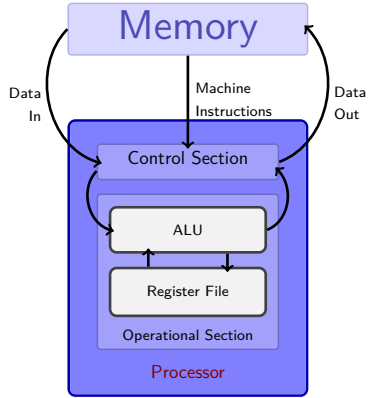

Figure 4: Schematic of General Purpose Processor with von-Neumann Architecture

PPU Architecture

- Based on POWER architecture
- Vector Extension
  → Parallelization
  → Performance
- Weight Updates
- Access to Synaptic Array

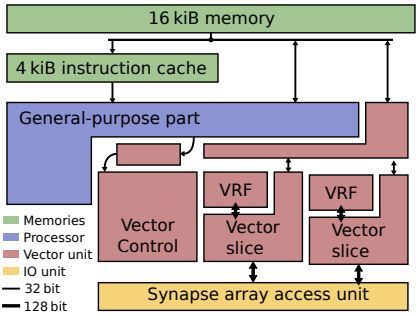

Figure 5: Structure of nux Architecture

Compiler Structure

- Program $\xrightarrow{\text{compile}}$ Executable
- Compiling Stages
- Compiler $\rightarrow$ Assembly
- Back-End is target-dependent

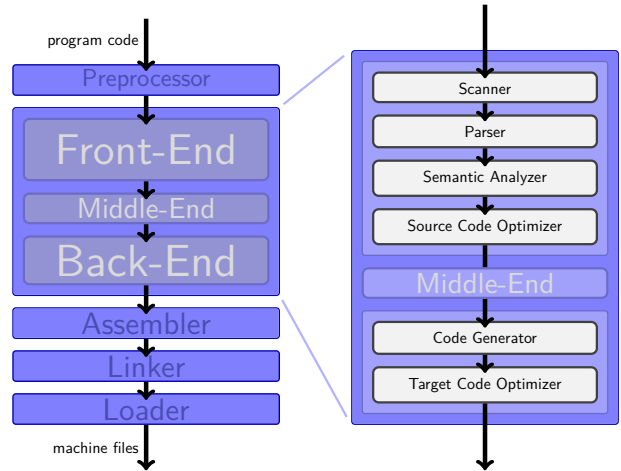

Figure 6: Structure of Compiling Process and Compiler

GCC for the PPU

- No “official” support for s2pp
- Currently using macros

Example from ppu_sweep.c

```
1  ...
2  /** Definitions for vector registers **/
3  #define VR_TMP     0
4  #define VR_CAUSAL  1
5  #define VR_ACAUSAL 2
6  #define VR_RST     3
7  #define VR_NULL_C  4
8
9  ...
```

GCC for the PPU

- No “official” support for s2pp
- Currently using macros

Example of Macro Usage

```
1  ...
2    fxv_splatb(VR_TMP, 1);
3    fxv_addbm(VR_TMP_1, VR_A, VR_TMP);
```

GCC for the PPU

- No “official” support for s2pp
- Currently using macros
- Existing AltiVec extension

Example of Macro Usage

```
1  ...
2    fxv_splatb(VR_TMP, 1);
3    fxv_addbm(VR_TMP_1, VR_A, VR_TMP);
```

Example of AltiVec

```
1  ...
2    tmp = vec_splat(1);
3    tmp = vec_add(tmp, tmp);
```

GCC for the PPU

- No “official” support for s2pp
- Currently using macros
- Existing AltiVec extension
- Intrinsics → **use this**

Example of Macro Usage

```
1  ...
2    fxv_splatb(VR_TMP, 1);
3    fxv_addbm(VR_TMP_1, VR_A, VR_TMP);
```

Example of AltiVec

```
1  ...
2    tmp = vec_splat(1);
3    tmp = vec_add(tmp, tmp);
```

Main Work of this Thesis

- Add GCC support for s2pp
- Overcome missing documentation
- AltiVec inspiration
- Components:
  - Vector Register
  - `vector` attribute
  - New intrinsics
  - ...

Example of Macro Usage

```
1  ...
2    fxv_splatb(VR_TMP, 1);
3    fxv_addbm(VR_TMP_1, VR_A, VR_TMP);
```

Example of AltiVec

```
1  ...
2    tmp = vec_splat(1);
3    tmp = vec_add(tmp, tmp);
```

New Features

- `-mcpu=nux` target flag

Compiling Directive

```
1  powerpc-linux-eabi-ppc -mcpu=nux
```

Example File

```
1  #include<s2pp.h>
2
3  start(){
4    return;
5  }
```

New Features

- `-mcpu=nux` target flag
- `vector` attribute

Compiling Directive

```
1  powerpc-linux-eabi-ppc -mcpu=nux
```

Example File

```
1  #include<s2pp.h>
2
3  start(){
4    vector int8_t vec;
5    return;
6  }
```

New Features

- `-mcpu=nux` target flag
- `vector` attribute
- s2pp vector intrinsics

Compiling Directive

```
1  powerpc-linux-eabi-ppc -mcpu=nux
```

Example File

```
1  #include<s2pp.h>
2
3  start(){
4  vector int8_t vec =fxv_splat(1);
5  vec = fxv_add(vec, vec);
6  return;
7  }
```

New Features

- `-mcpu=nux` target flag
- `vector` attribute
- s2pp vector intrinsics
- Inline assembly

Compiling Directive

```
1  powerpc-linux-eabi-ppc -mcpu=nux
```

Example File

```
1  #include<s2pp.h>
2
3  start(){
4  vector int8_t vec =fxv_splat(1);
5  vec = fxv_add(vec, vec);
6  asm ("fxvaddbm %0, %0, %0"
7       :"=kv" (vec)::);
8  return;
9  }
```

New Features

- `-mcpu=nux` target flag
- `vector` attribute
- s2pp vector intrinsics
- Inline assembly
- Supports optimization

Compiling Directive

```
1  powerpc-linux-eabi-ppc -mcpu=nux -O1
```

Example File

```
1  #include<s2pp.h>
2
3  start(){
4  vector int8_t vec =fxv_splat(1);
5  vec = fxv_add(vec, vec);
6  asm ("fxvaddbm %0, %0, %0"
7       :"=kv" (vec)::);
8  return;
9  }
```

Outlook

Comparison of unoptimized code (left) and `-O1` optimized code (right)

```
1  start:
2  stwu %r1,-64(%r1)
3  stw %r31,60(%r1)
4  mr %r31,%r1
5  li %r9,1
6  fxvsplatb %f12,%r9
7  li %r9,16
8  fxvstax %f12,%r31,%r9
9  li %r9,16
10 fxvlax %f12,%r31,%r9
11 li %r9,16
12 fxvlax %f11,%r31,%r9
13 fxvaddbfs %f12,%f12,%f11
14 li %r9,16
15 fxvstax %f12,%r31,%r9
16 #APP
17  # 277 "nux/main.c" 1
18 fxvaddbm %f12, %f12, %f12
19  # 0 "" 2
20 #NO_APP
21 li %r9,16
22 fxvstax %f12,%r31,%r9
23 nop
24 addi %r11,%r31,64
25 lwz %r31,-4(%r11)
26 mr %r1,%r11
27 blr
```

```
1  start:
2  stwu %r1,-48(%r1)
3  li %r9,1
4  fxvsplatb %f12,%r9
5  li %r9,16
6  fxvstax %f12,%r1,%r9
7  fxvlax %f12,%r1,%r9
8  fxvlax %f11,%r1,%r9
9  fxvaddbfs %f12,%f12,%f11
10 fxvstax %f12,%r1,%r9
11 #APP
12  # 277 "nux/main.c" 1
13 fxvaddbm %f12, %f12, %f12
14  # 0 "" 2
15 #NO_APP
16 fxvstax %f12,%r1,%r9
17 addi %r1,%r1,48
18 blr
```

Outlook

- Further testing for bugs
- Existing applications with `libnux`
- Extending test coverage
- Debugging support?
- **New tool for development!**

Comparison of unoptimized code (left) and `-O1` optimized code (right)

```
1   start:
2   stwu %r1,-64(%r1)
3   stw %r31,60(%r1)
4   mr %r31,%r1
5   li %r9,1
6   fxvsplatb %f12,%r9
7   li %r9,16
8   fxvstax %f12,%r31,%r9
9   li %r9,16
10  fxvlax %f12,%r31,%r9
11  li %r9,16
12  fxvlax %f11,%r31,%r9
13  fxvaddbfs %f12,%f12,%f11
14  li %r9,16
15  fxvstax %f12,%r31,%r9
16  #APP
17   # 277 "nux/main.c" 1
18  fxvaddbm %f12, %f12, %f12
19   # 0 "" 2
20  #NO_APP
21  li %r9,16
22  fxvstax %f12,%r31,%r9
23  nop
24  addi %r11,%r31,64
25  lwz %r31,-4(%r11)
26  mr %r1,%r11
27  blr
```

```
1   start:
2   stwu %r1,-48(%r1)
3   li %r9,1
4   fxvsplatb %f12,%r9
5   li %r9,16
6   fxvstax %f12,%r1,%r9
7   fxvlax %f12,%r1,%r9
8   fxvlax %f11,%r1,%r9
9   fxvaddbfs %f12,%f12,%f11
10  fxvstax %f12,%r1,%r9
11  #APP
12   # 277 "nux/main.c" 1
13  fxvaddbm %f12, %f12, %f12
14   # 0 "" 2
15  #NO_APP
16  fxvstax %f12,%r1,%r9
17  addi %r1,%r1,48
18  blr
```

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

Figure 7: Comparison of the Representation of Weights in Synapses and the Fractional Representation of Vector Components for Fixed-Point Saturational 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.

Additional Figures

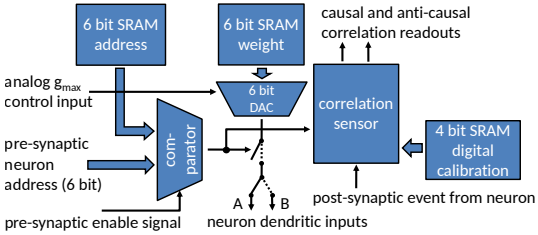

Figure 8: Block Diagram of the Synapse Circuit (modified from Friedmann et al.).

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

Figure 9: 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`.

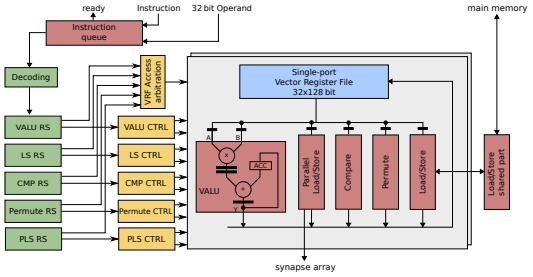

Figure 10: Detailed Structure of the s2pp Vector Extension (taken from Friedmann et al.)

Additional Figures

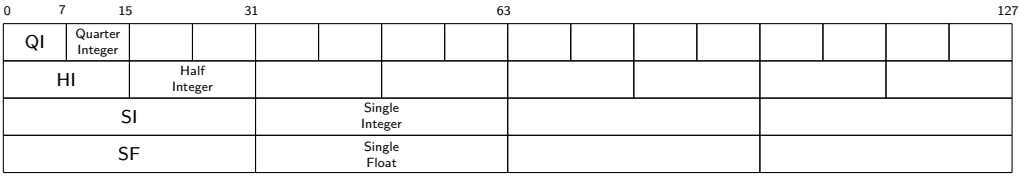

Figure 11: Vector structures are 128 bits wide and split into common word sizes.

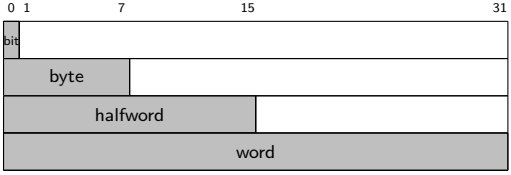

Figure 12: Illustration of Word Sizes for 32-bit Words