Currently known bugs/features of the PPU and how to workaround them

Arthur Heimbrecht

February 7, 2017

1. Conditionals
2. fxvsplatb
3. fxvpck
4. syncing

conditionals

```
li %r0, 1
li %r1, 0x2222
li %r2, 0x3333
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r0
fxvsplath 4, %r0
fxvcmph 0
fxvaddhm 3, 1, 2, 1
fxvaddhm 4, 2, 2, 2
```

expected result

```
vr0: 0001 0001 0001 0001 0001 0001 0001 0001
vr1: 2222 2222 2222 2222 2222 2222 2222 2222
vr2: 3333 3333 3333 3333 3333 3333 3333 3333
vr3: 5555 5555 5555 5555 5555 5555 5555 5555
vr4: 0001 0001 0001 0001 0001 0001 0001 0001
```

actual result

```
vr0: 0001 0001 0001 0001 0001 0001 0001 0001
vr1: 2222 2222 2222 2222 2222 2222 2222 2222
vr2: 3333 3333 3333 3333 3333 3333 3333 3333
vr3: 5555 5555 5555 5555 5555 5555 5555 5555
vr4: 5555 5555 5555 5555 5555 5555 5555 5551
```

possible workarounds with fxvaddhm

this abuses the the bug

zeros

```
li %r0, 0
li %r1, 1
li %r2, 0x2222
li %r3, 0x3333
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r3
fxvsplath 4, %r1
fxvsplath 5, %r1
fxvcmph 1
fxvaddhm 4, 0, 0, 0
fxvaddhm 4, 2, 1, 1
fxvaddhm 5, 0, 0, 0
fxvaddhm 5, 3, 3, 2
```

```
vr0: 0000 0000 0000 0000
vr1: 0001 0001 0001 0001
vr2: 2222 2222 2222 2222
vr3: 3333 3333 3333 3333
vr4: 5555 5555 5555 5555
vr5: 0000 0000 0000 0000
```

previous content

```
li %r0, 0
li %r1, 1
li %r2, 0x2222
li %r3, 0x3333
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r3
fxvsplath 4, %r1
fxvsplath 5, %r1
fxvcmph 1
fxvaddhm 4, 4, 0, 0
fxvaddhm 4, 2, 1, 1
fxvaddhm 5, 5, 0, 0
fxvaddhm 5, 3, 3, 2
```

```
vr0: 0000 0000 0000 0000
vr1: 0001 0001 0001 0001
vr2: 2222 2222 2222 2222
vr3: 3333 3333 3333 3333
vr4: 5555 5555 5555 5555
vr5: 0001 0001 0001 0001
```

first operand's value

```
li %r0, 0
li %r1, 1
li %r2, 0x2222
li %r3, 0x3333
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r3
fxvsplath 4, %r1
fxvsplath 5, %r1
fxvcmph 1
fxvaddhm 4, 2, 0, 0
fxvaddhm 4, 2, 1, 1
fxvaddhm 5, 3, 0, 0
fxvaddhm 5, 3, 3, 2
```

```
vr0: 0000 0000 0000 0000
vr1: 0001 0001 0001 0001
vr2: 2222 2222 2222 2222
vr3: 3333 3333 3333 3333
vr4: 5555 5555 5555 5555
vr5: 3333 3333 3333 3333
```

possible workarounds with fxvsel

this is pretty but only valid for airthmeitc without accumulator

zeros

```
li %r0, 0
li %r1, 1
li %r2, 0x2222
li %r3, 0x3333
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r3
fxvsplath 4, %r1
fxvsplath 5, %r1
fxvcmph 1
fxvaddhm 4, 2, 1, 0
fxvsel 4, 4, 0, 1
fxvaddhm 5, 3, 3, 0
fxvsel 5, 5, 0, 2
```

```
vr0: 0000 0000 0000 0000
vr1: 0001 0001 0001 0001
vr2: 2222 2222 2222 2222
vr3: 3333 3333 3333 3333
vr4: 5555 5555 5555 5555
vr5: 0000 0000 0000 0000
```

previous content

```
li %r0, 0
li %r1, 1
li %r2, 0x2222
li %r3, 0x3333
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r3
fxvsplath 4, %r1
fxvsplath 5, %r1
fxvcmph 1
fxvaddhm 0, 2, 1, 0
fxvsel 4, 0, 4, 1
fxvaddhm 0, 3, 3, 0
fxvsel 5, 0, 5, 2
```

```
vr0: 0000 0000 0000 0000
vr1: 0001 0001 0001 0001
vr2: 2222 2222 2222 2222
vr3: 3333 3333 3333 3333
vr4: 5555 5555 5555 5555
vr5: 0001 0001 0001 0001
```

first operand's value

```
li %r1, 1
li %r2, 0x2222
li %r3, 0x3333
fxvsplath 1, %r1
fxvsplath 2, %r2
fxvsplath 3, %r3
fxvsplath 4, %r1
fxvsplath 5, %r1
fxvcmph 1
fxvaddhm 4, 2, 1, 0
fxvsel 4, 4, 2, 1
fxvaddhm 5, 3, 3, 0
fxvsel 5, 5, 3, 2
```

```
vr0: 0000 0000 0000 0000
vr1: 0001 0001 0001 0001
vr2: 2222 2222 2222 2222
vr3: 3333 3333 3333 3333
vr4: 5555 5555 5555 5555
vr5: 3333 3333 3333 3333
```

problem

```
li %r0, 0
...
li %r0, 1
fxvsplatb 0, %r0
```

expected result and actual result

```
vr0: 0001 0001 0001 0001 0001 0001 0001 0001
```

expected result according to fxv.h

```
vr0: 0000 0000 0000 0000 0000 0000 0000 0000
```

fxv.h content

```
...
/** Bugfix for fxvsplatb in HICANN-DLS v1 !
 * This instruction has incorrect hazard detection for the general-purpose register
 * argument. E.g.:
 *   lis r9, 15
 *   fxvsplatb 1, r9
 * will use the old value of register r9.
 *
 * The fix uses the fxvsplath instruction, which does hazard detection correctly. */
//#define fxv_splatb(x, y) _fxv_insn_gpr1("fxvsplatb", x, y)
#define _fxv_splatb(rt, gpra) \
  asm volatile("fxvsplath␣" #rt ",␣%[a]" \
      :: [a] "r"( (( (gpra) & 0xff) << 8) | ( (gpra) & 0xff) ))
#define fxv_splatb(x, y) _fxv_splatb(x, y)
/* End of bugfix for fxvsplatb */
...
```

fxvpck

```
li %r0, 0x1122
li %r1, 0x3344
fxvsplath 0, %r0
fxvsplath 1, %r1
fxvpcku 2, 0, 1
```

```
vr0: 1122 1122 1122 1122 1122 1122 1122 1122
vr1: 3344 3344 3344 3344 3344 3344 3344 3344
vr2: 1111 1111 1111 1111 3333 3333 3333 3333
```

possible workarounds

syncing after load and store

```
li %r0, 0x11
fxvsplatb 0, %r0
li %r31, 0x3000
fxvstax 0, 0, %r31
sync
li %r31, 0x2000
```

```
2000: 0000 0000 0000 0000 0000 0000 0000 0000
...
3000: 1111 1111 1111 1111 1111 1111 1111 1111
```

syncing after splat?

```
li %r0, 0x11
li %r0, 0x22
fxvsplatb 0, %r0
sync
li %r0, 0x33
```

```
vr0: 2222 2222 2222 2222 2222 2222 2222 2222
```

apperently not necessary