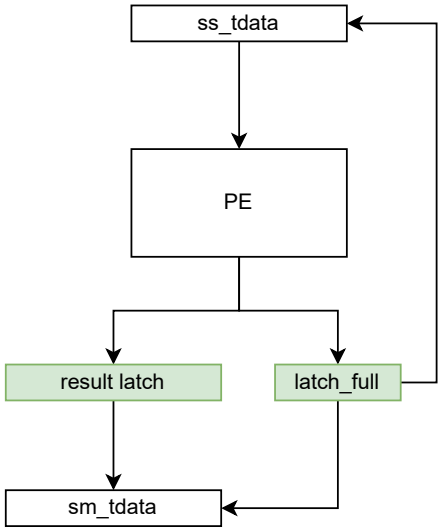
# Improve FIR



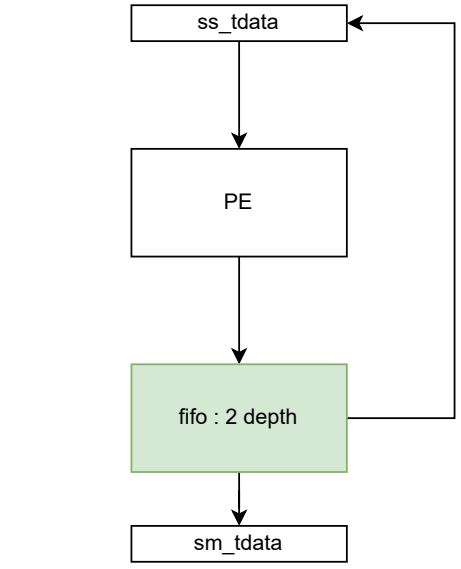

## full - ss_tready

ss_tready = !latch_full

pe result latch , latch full <= 1

sm_tvalid &sm_tready & latch_full , latch full <= 0

next ss_tdata need to wait current sm_tvalid



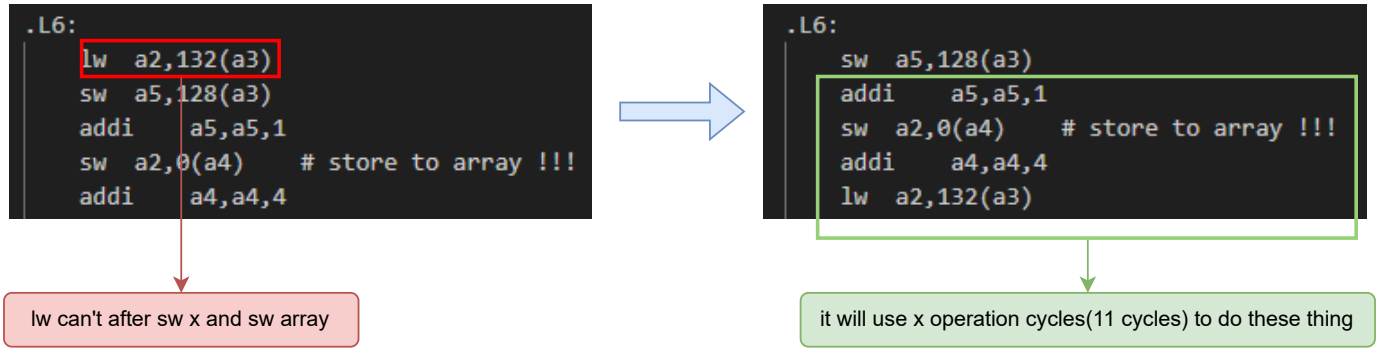

## full - ss_tready

have 2 depth fifo can pass next ss_tdata ready before current sm_tdata valid.

So, it can do that, change the next x input before the current y output.

```
25      wb_write(reg_fir_x_in, t);
26      for (uint8_t t = 1; t < N; t++) {
27          temp = wb_read(reg_fir_y_out);
28          wb_write(reg_fir_x_in, t);
29          outputsignal[t - 1] = temp;
30      }
```

```
25      wb_write(reg_fir_x_in, t);
26      for (uint8_t t = 1; t < N; t++) {
27          wb_write(reg_fir_x_in, t);
28          temp = wb_read(reg_fir_y_out);
29          outputsignal[t - 1] = temp;
30      }
```




Why it can reduce cycle ?

| | | | | | |
|---|---|---|---|---|---|
| ss_tready | 1 | ... | 0 | 0 | 1 |
| sm_tvalid | 0 | ... | 1 | 0 | 0 |

next x need wait y valid .

| | | | | | |
|---|---|---|---|---|---|
| ss_tready | 1 | ... | 1 | 0 | 0 |
| sm_tvalid | 0 | ... | | | |

next x input

current y will use these cycle to output