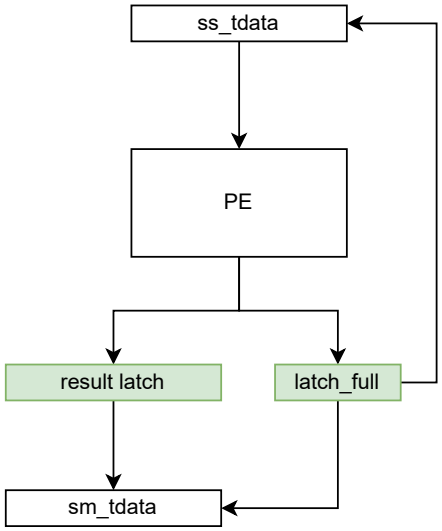
Improve FIR



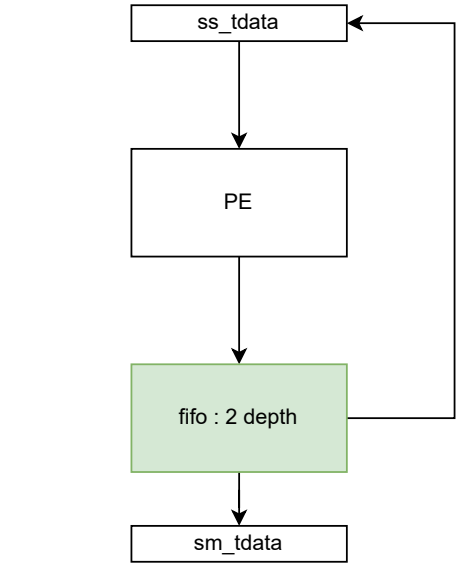
full - ss_tready

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

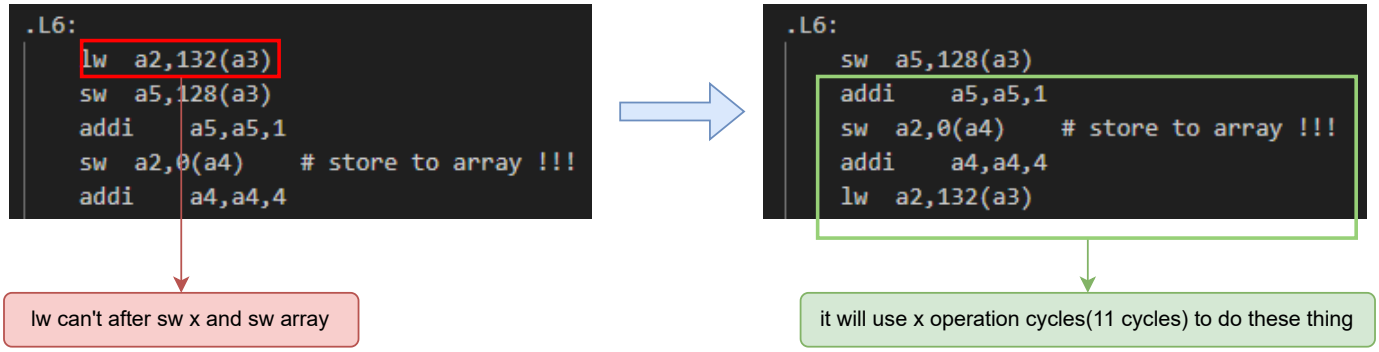
have 2 depth fifo can pass next ss_tdata ready before current sm_tdata valid.

```
424     assign sm_tdata  = sm_fifo_data;
425
426     fifo
427     #(
428         .WIDTH      (pDATA_WIDTH),
429         .DEPTH      (2)
430     )   sm_fifo
431     (
432         .clk        (axis_clk),
433         .rst_n      (axis_rst_n),
434         .pre_full   (sm_full),
435         .pre_empty  (sm_empty),
436         .w_valid    (data_wr_en),
437         .r_ready    (sm_tready & sm_tvalid),
438         .data_in    (result),
439         .data_out   (sm_fifo_data)
440     );
```

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