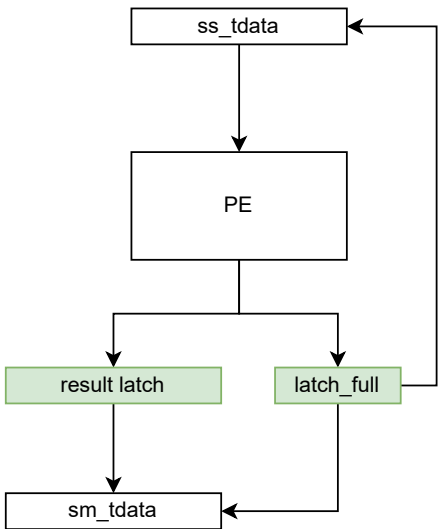


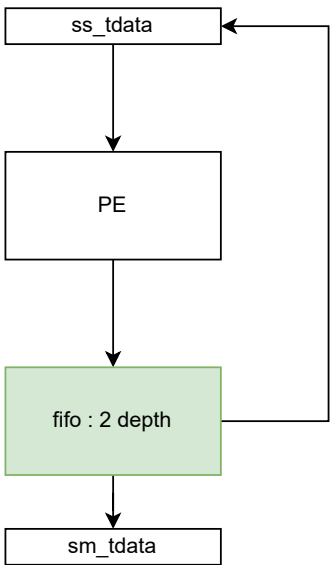
# Improve FIR



## 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  }
```



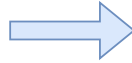
```
25  wb_write(reg_fir_x_in, t);
26  for (uint8_t t = 1; t < N; t++) {
27      wb_write(reg_fir_x_in, t);
28      temp = wb_read(reg_fir_y_out);
29      outputsignal[t - 1] = temp;
30  }
```

```

.L6:
lw  a2,132(a3)
sw  a5,128(a3)
addi a5,a5,1
sw  a2,0(a4) # store to array !!!
addi a4,a4,4

```

lw can't after sw x and sw array



```

.L6:
sw  a5,128(a3)
addi a5,a5,1
sw  a2,0(a4) # store to array !!!
addi a4,a4,4
lw  a2,132(a3)

```

it will use x operation cycles(11 cycles) to do these thing

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