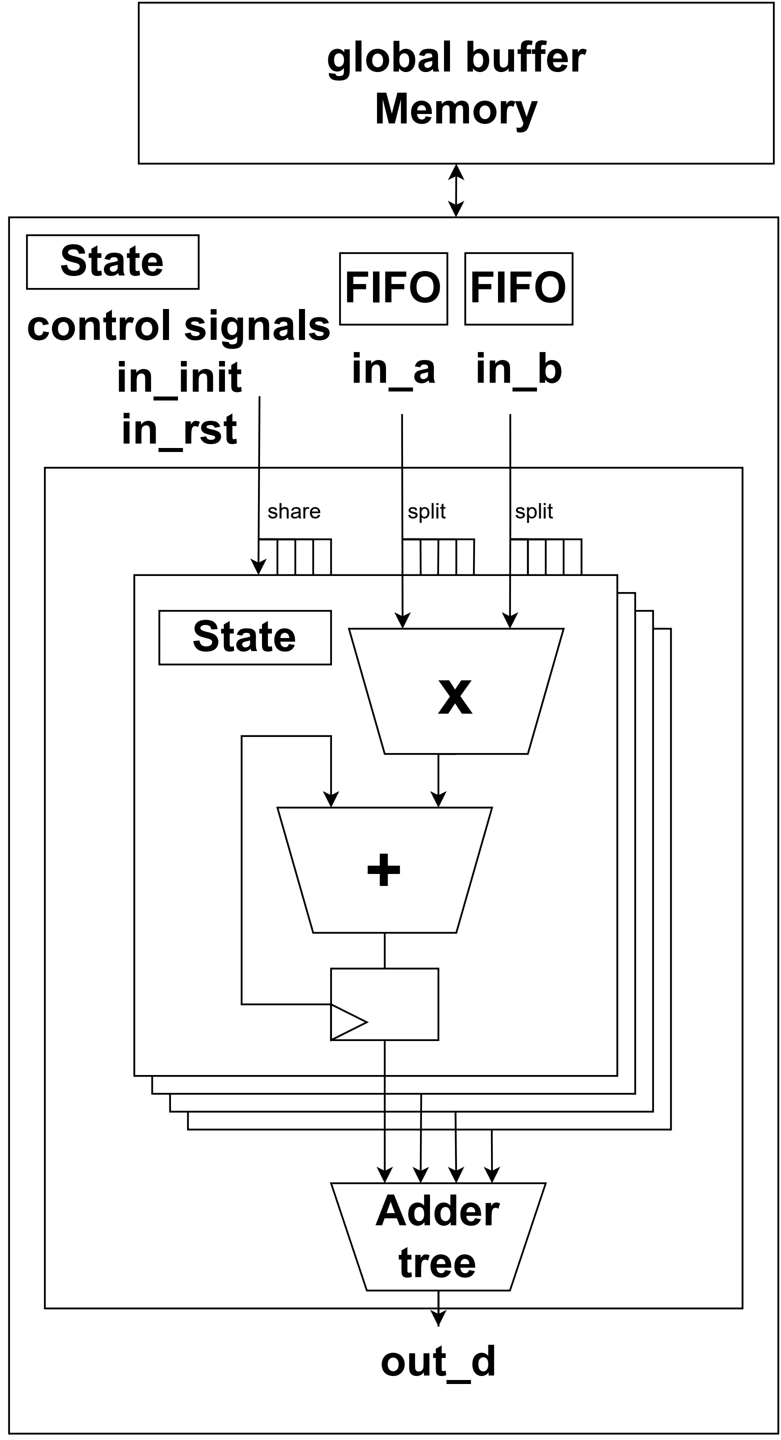
PE Controller

Hardware System Design

Spring, 2023

PE Controller

- Vector-vector inner product
- FSM over **`PEStack`**
- Global buffer (simulated for now)
  - Dual-port Memory
  - Assume 1 port connected to CPU over AXI
  - Limited bandwidth
- Local buffer
  - **`SyncFIFOBuffered`**
  - Scalable
  - (will be) high bandwidth



Amaranth examples

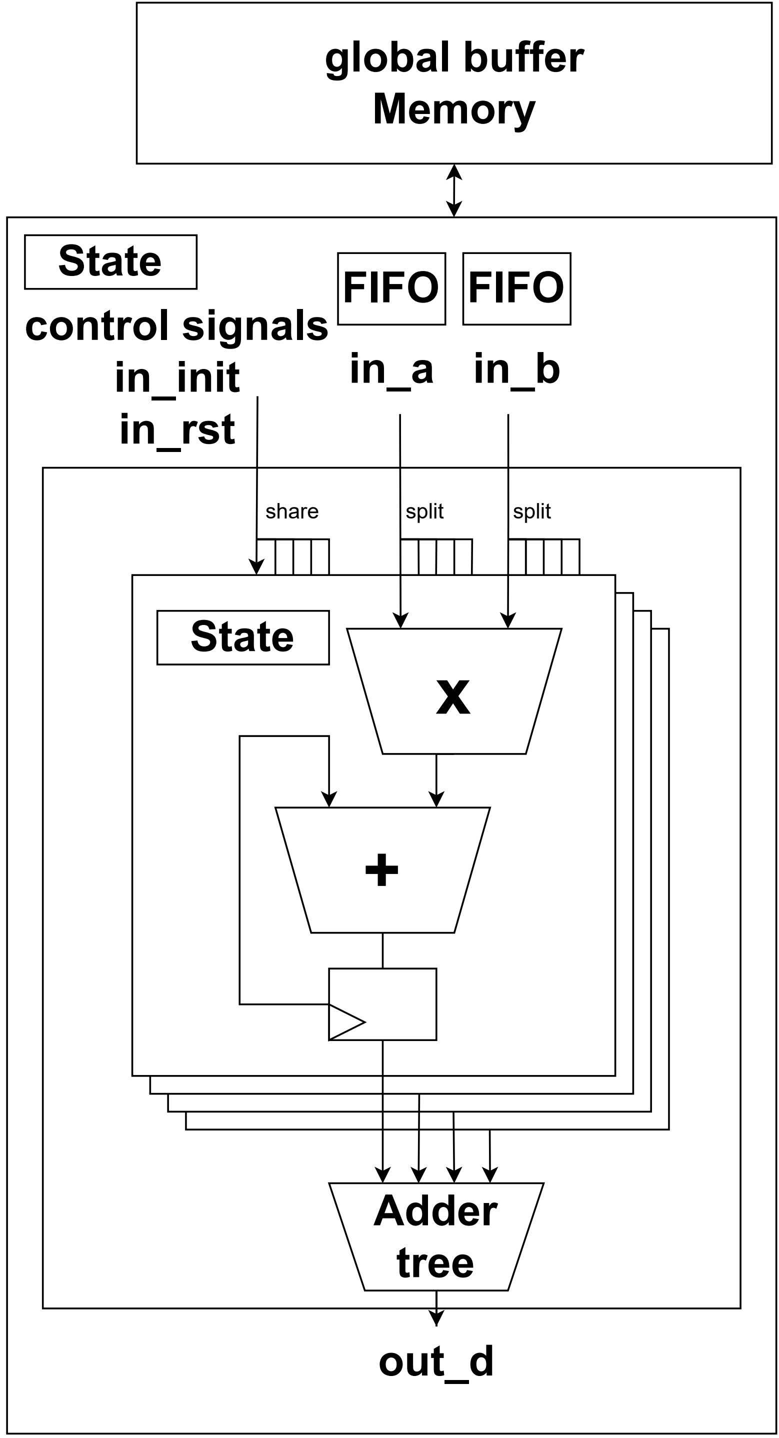
- Assignment order
- **Memory**
- **FIFO**
- FSM

- **Tutorial code**
  https://www.notion.so/tutorial-code-f5ba421763394b56968822fc6af7a7f0?pvs=4

Memory layout

- Code area
  - Address 0 → **`0xCAFE_NNNN`**
    - Where **`NNNN`** is number of stores done
    - **`0xCAFE_0000`** → code is ready
      - initialize module
      - State change from INIT to FETCH

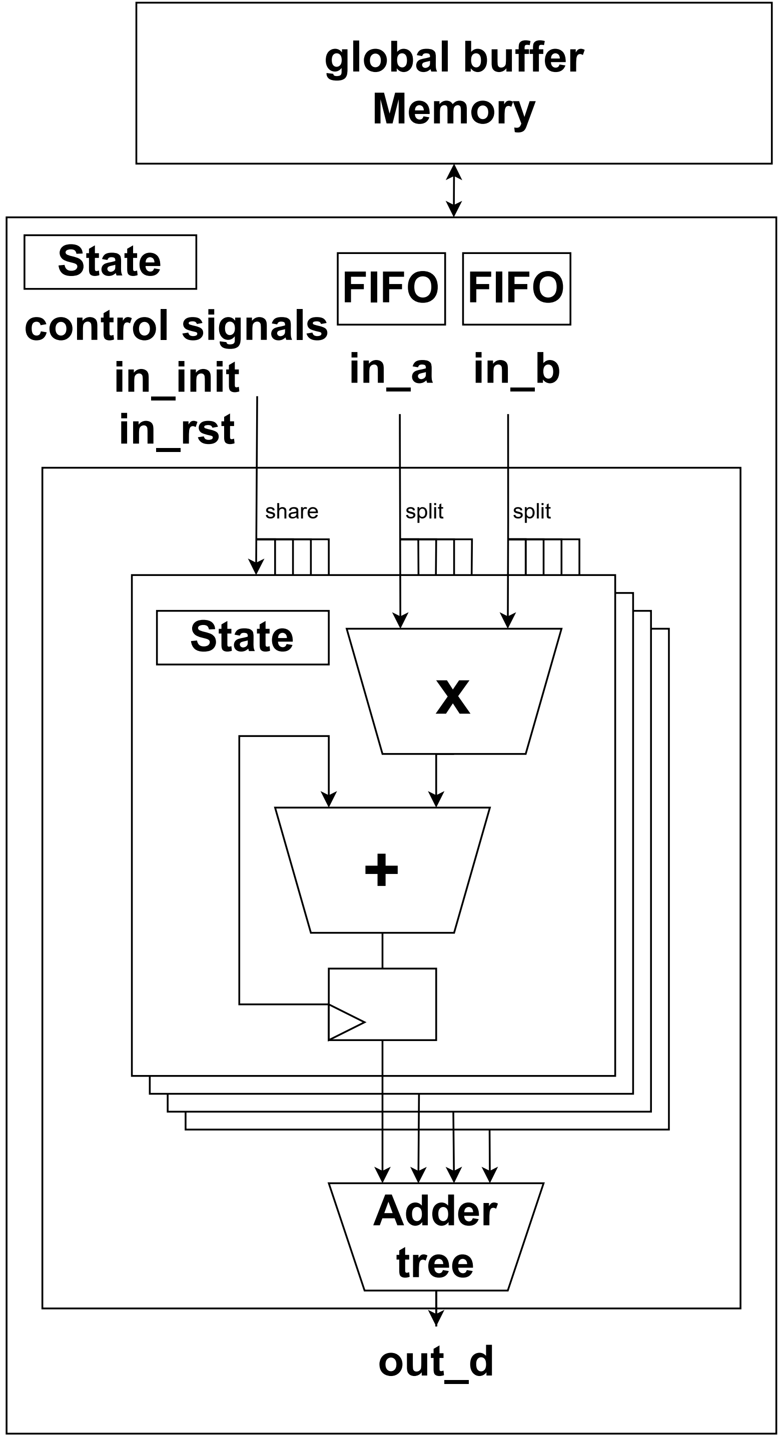
    …
  - Address N > 0 → **`0xCAFE_CAFE`**
    - Mark end of code area
    - State change from FETCH to INIT
- Data area
  - Address N+1 ~



PE Controller

- Instruction set
    - SET_M
    - LOAD
    - STORE
    - EXEC
    - FLUSH

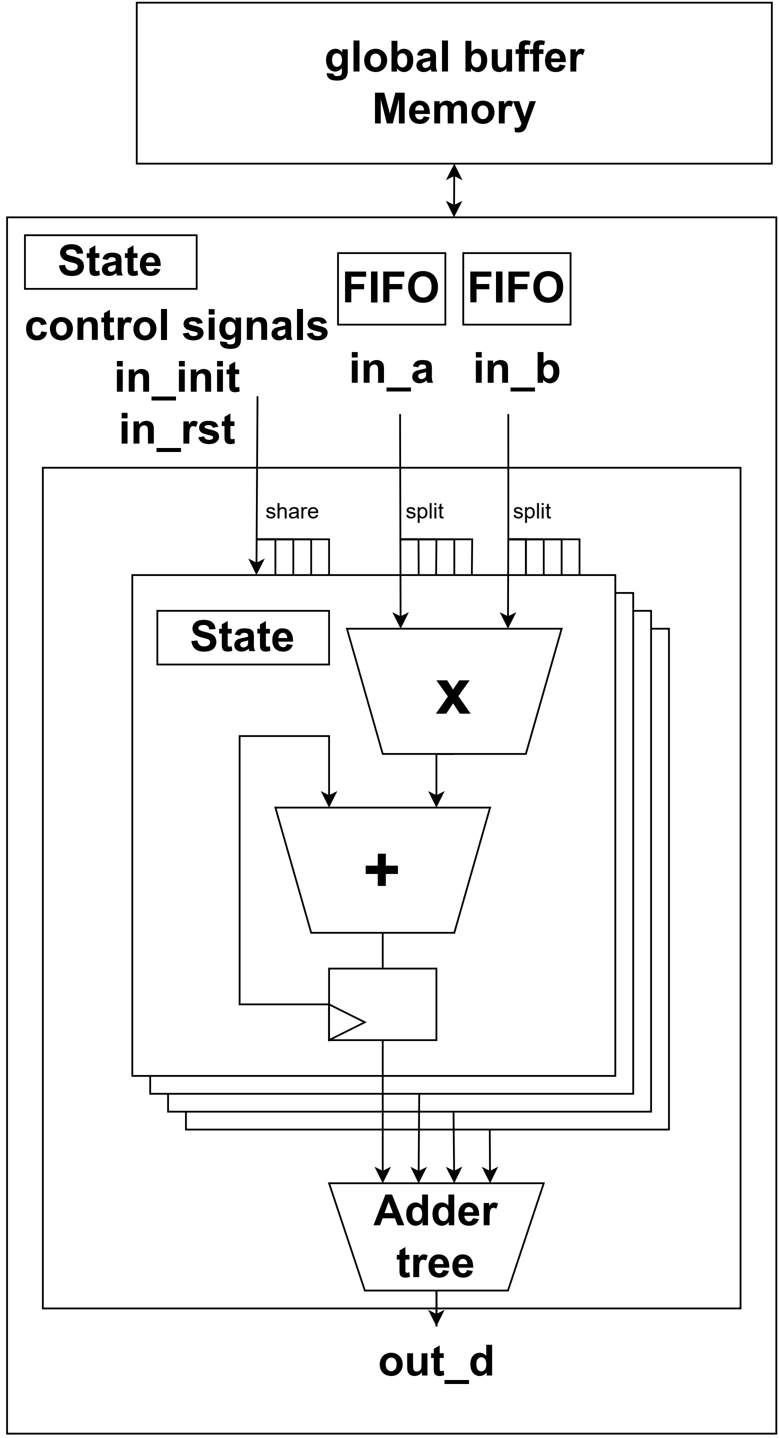
- States
    - INIT
    - FETCH
    - DECODE
    - LOAD
    - EXEC
    - STORE
    - FLUSH



Instruction set (1/3)

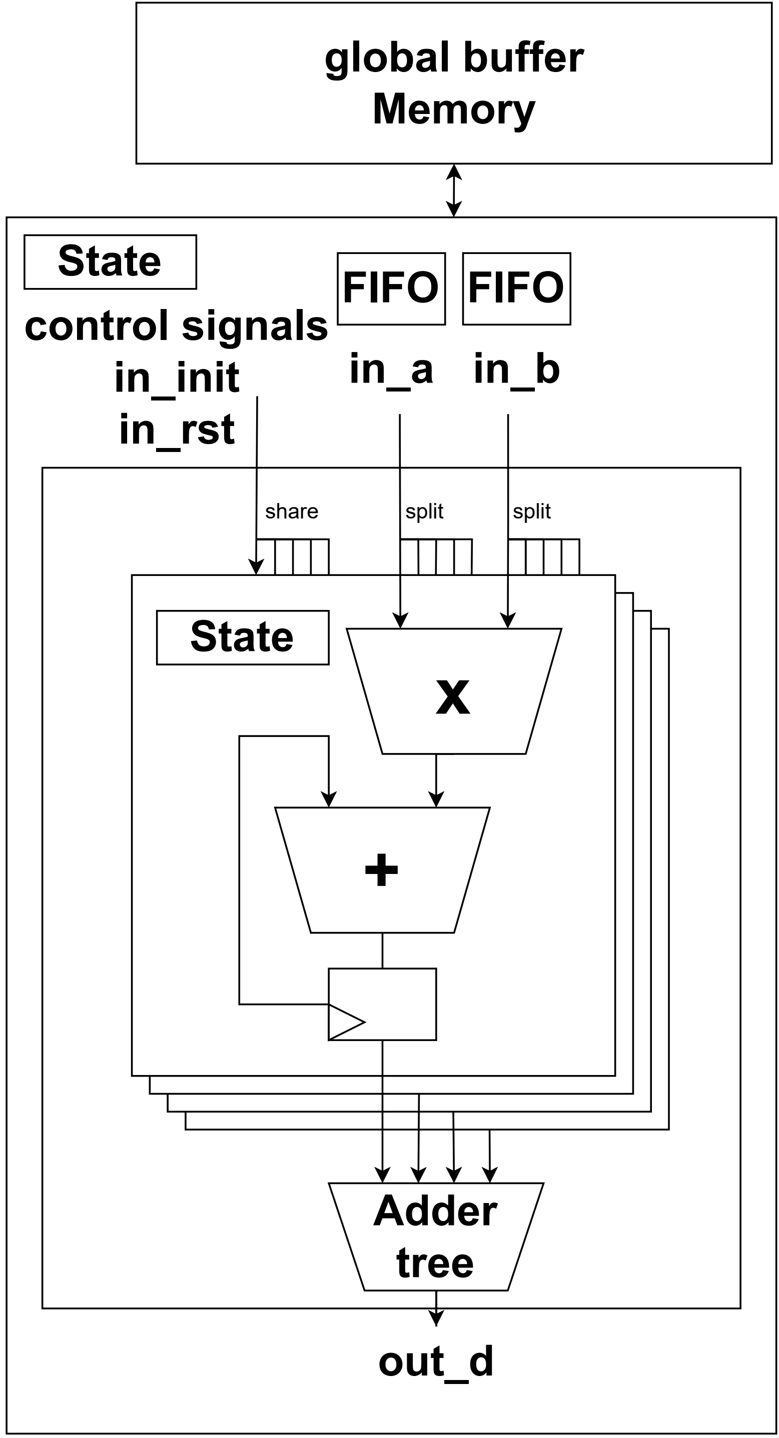
- 32-bit per instruction
  - **`4b OPCode + 4b V1 + 24b V2`**
    (MSB) (LSB)

- SET_M (set metadata)
  - **`SET_M + None + fan_in`**
  - **`fan_in`**
    - Least significant **`cnt_bits`**-bit of **`V2`**
    - number of memory lines
    - Data size = **`width * fan_in`**



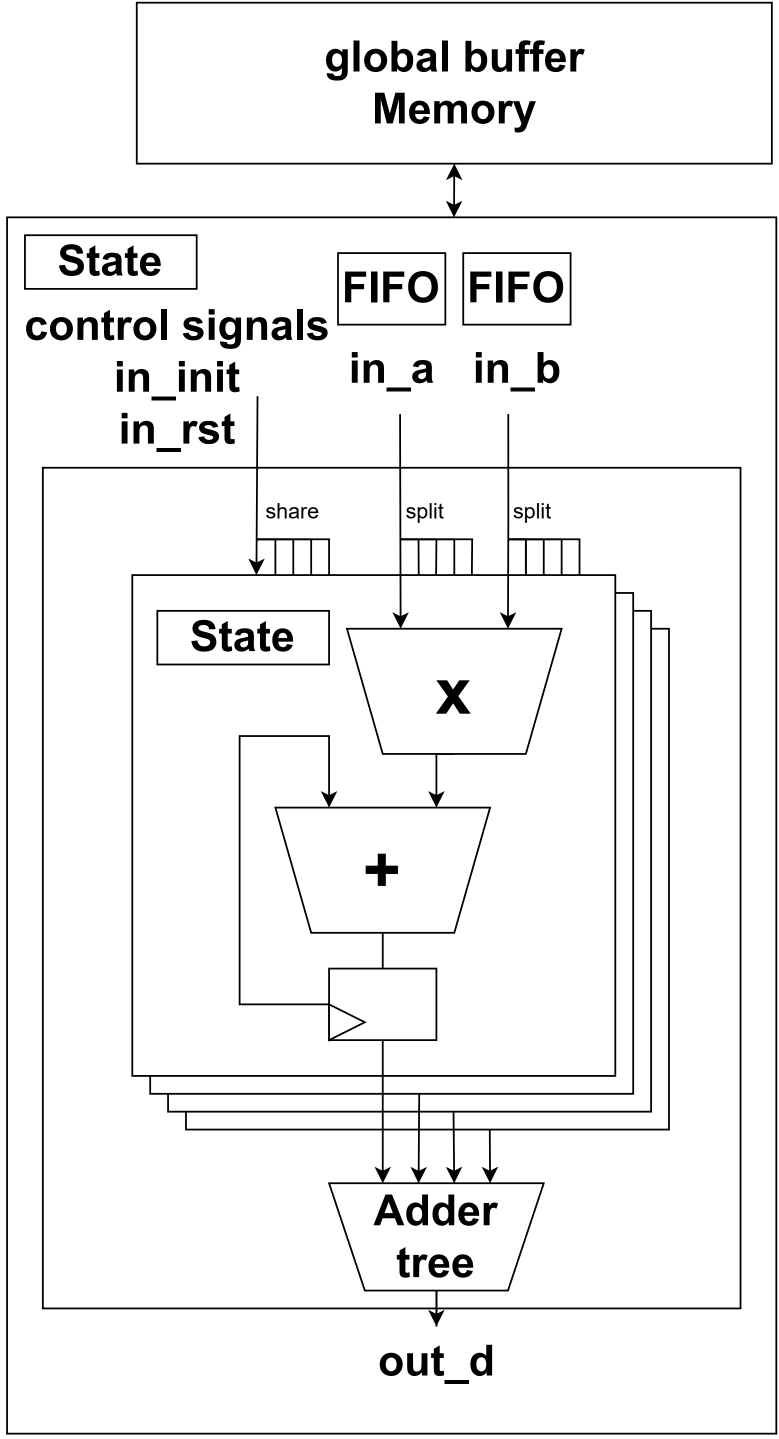
Instruction set (2/3)

- LOAD (global buffer → FIFO)
  - **`LOAD + LOAD_DEST + gb_addr`**
  - **`LOAD_DEST`**
    - select one of {FIFO A, FIFO B}
  - **`gb_addr`**
    - global buffer address to start loading from
    - Data size = **`width * fan_in`**
- STORE (PE out_d → global buffer)
  - **`STORE + None + gb_addr`**
  - Data size = **`acc_bits * 1`**



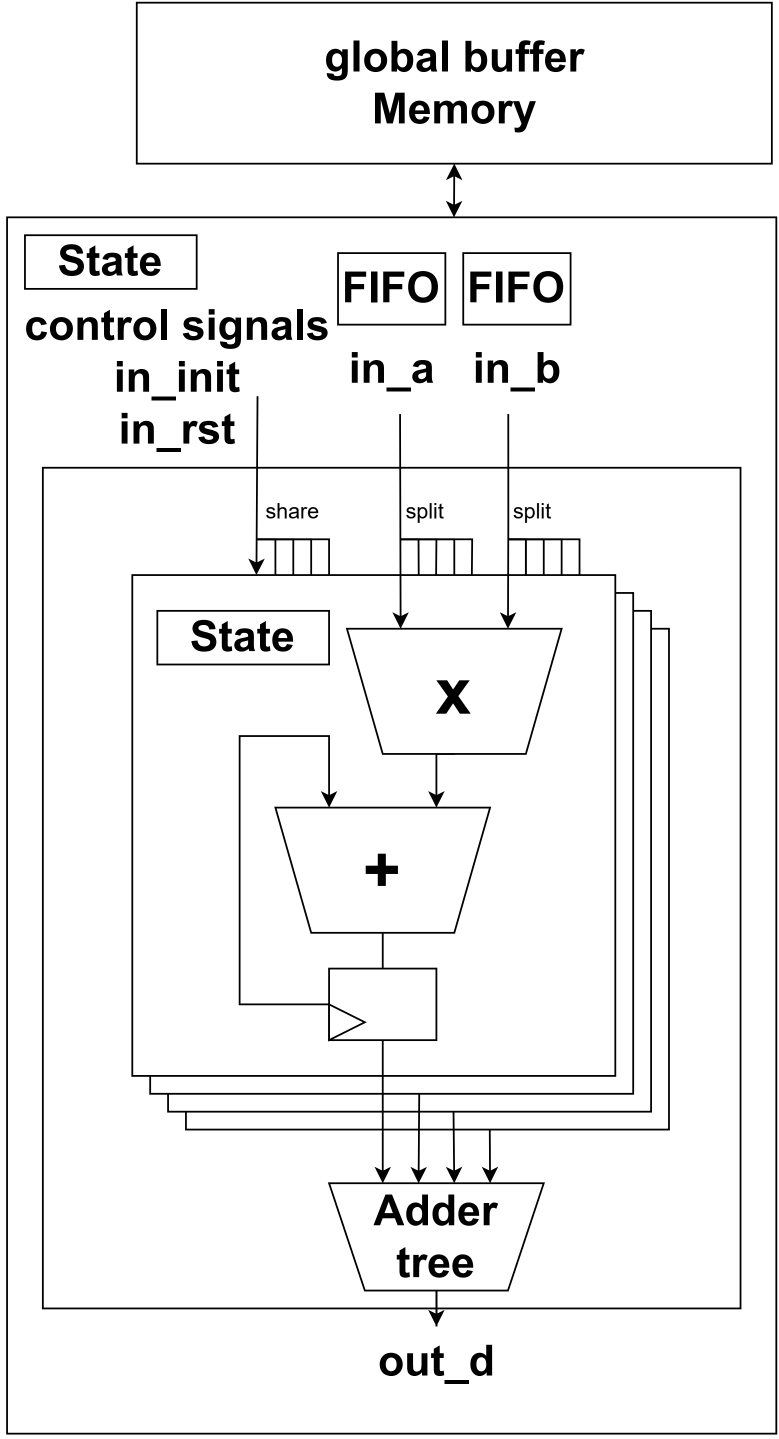
Instruction set (3/3)

- EXEC (compute inner product)
  - **`EXEC+\`**
    `[None + reuse_b + None + activation_code]+\`
    `value_for_activation`
    (Ignore **`activation_code`** and **`value_for_activation`** for now)
  - Feed PE for **`fan_in`** cycles
  - **`reuse_b`**
    - Reuse option for FIFO B (weight)
    - If set, FIFO B feeds itself while feeding PE
- FLUSH (empty FIFO B)
  - **`FLUSH + None + None`**



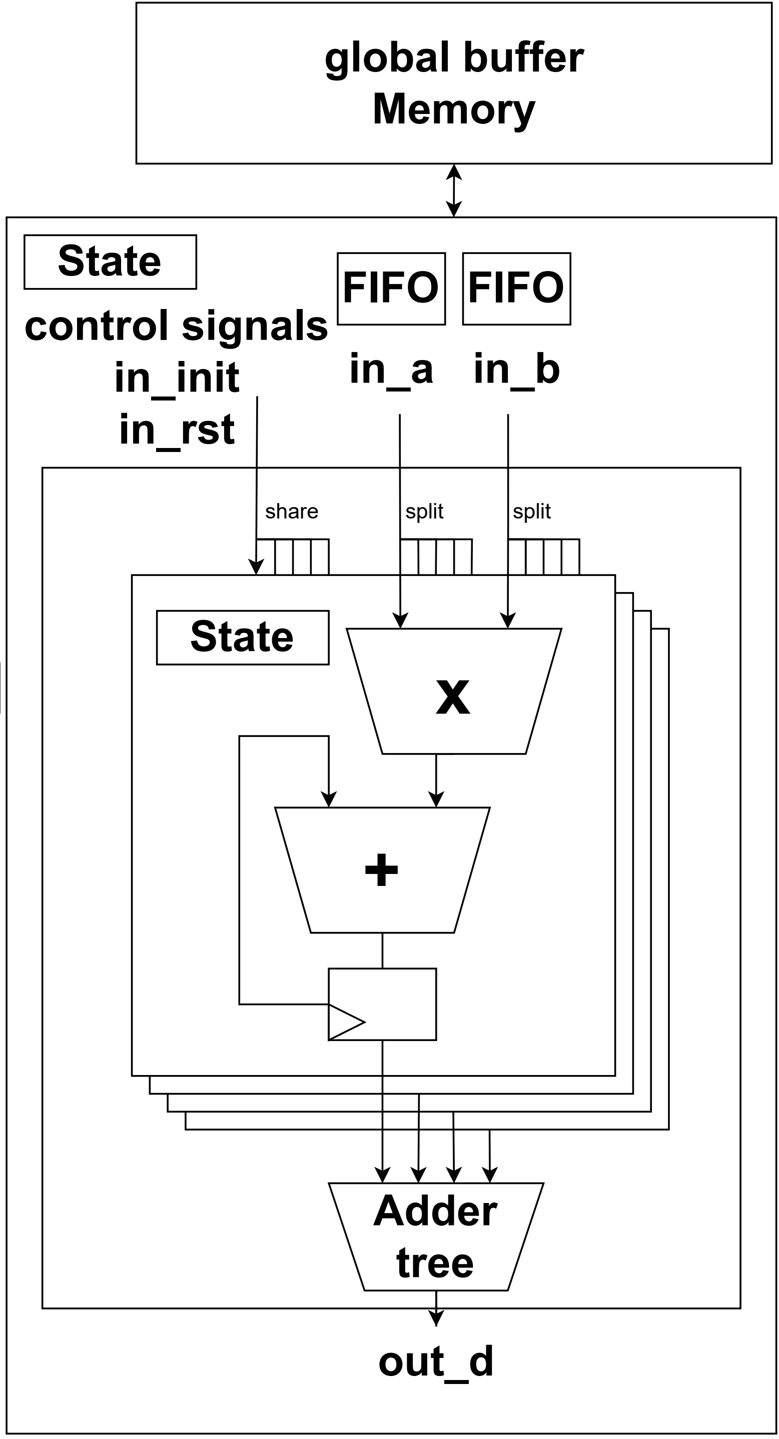
States (1/4)

- INIT
  - Wait for global buffer address 0 == **`0xCAFE0000`**
  - INIT → FETCH
- FETCH (fetch code from global buffer)
  - manage & use PC
  - Initialize internal registers in bulk
  - 2 cycles
    - set PC + memory read delay
  - FETCH → DECODE



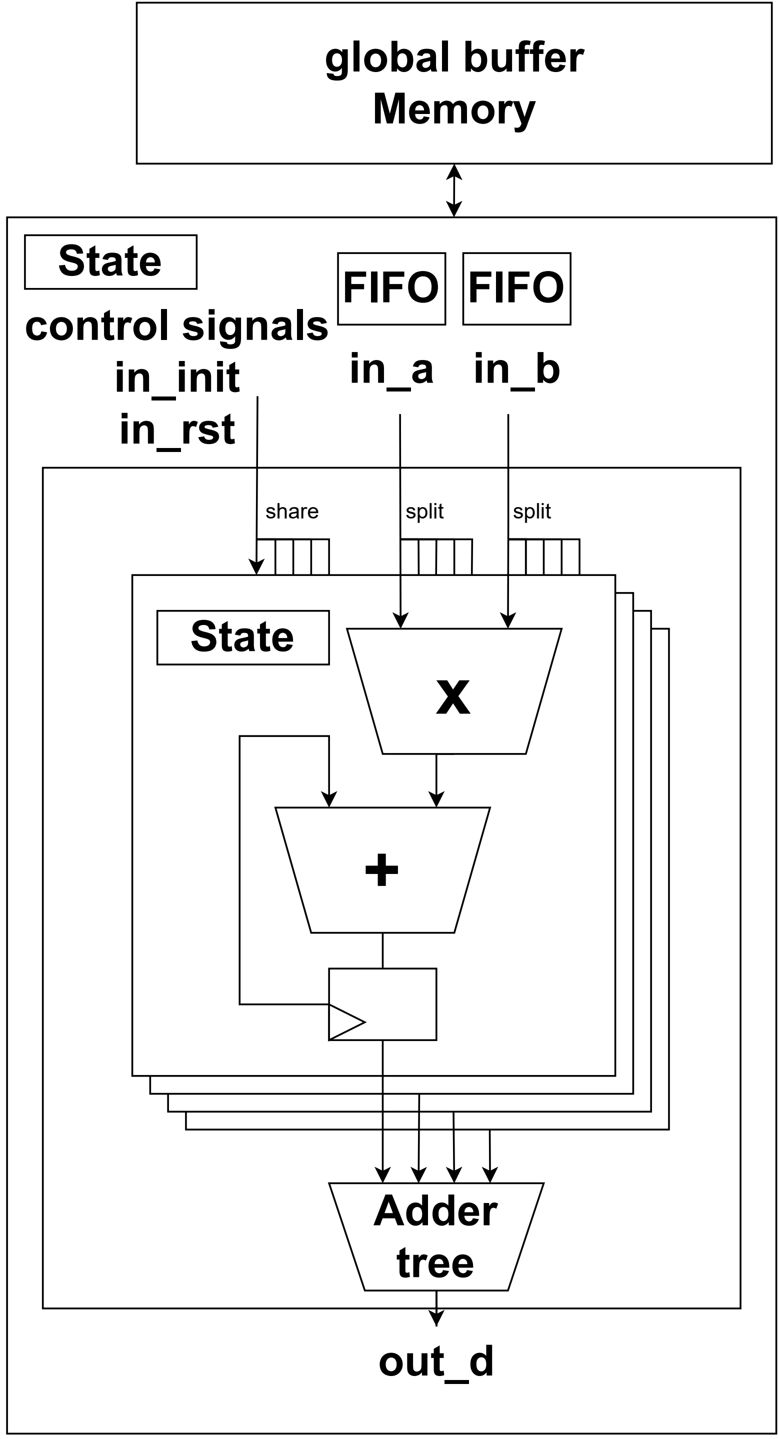
States (2/4)

- DECODE (decode instruction & decide next)
  - Handle SET_M instruction
  - 1 cycle
  - DECODE → [INIT | FETCH | LOAD | EXEC | STORE | FLUSH]
    - If undefined instruction then DECODE → INIT
- LOAD (global buffer → FIFO)
  - Destination = [FIFO A | FIFO B]
  - 1 + `fan_in * 1` cycles
  - LOAD → FETCH



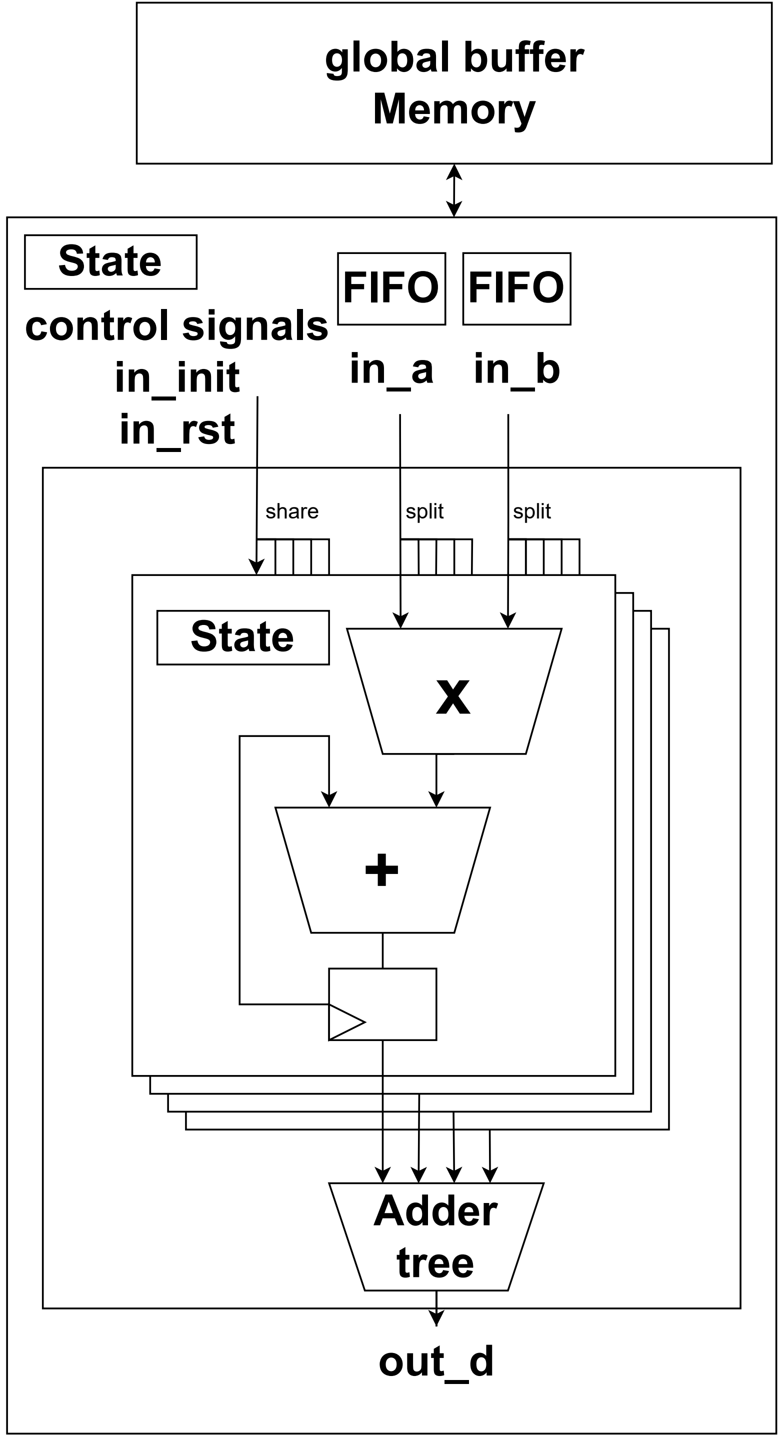
States (3/4)

- EXEC (compute inner product)
  - Feed PE until at least one FIFO is empty
  - If **`reuse_b`**, FIFO B feeds itself while feeding PE
  - **`fan_in`** cycles
  - EXEC → FETCH
- STORE (PE out_d → global buffer)
  - 1 cycle
  - Write number of stores done at address 0
  - STORE → FETCH



States (4/4)

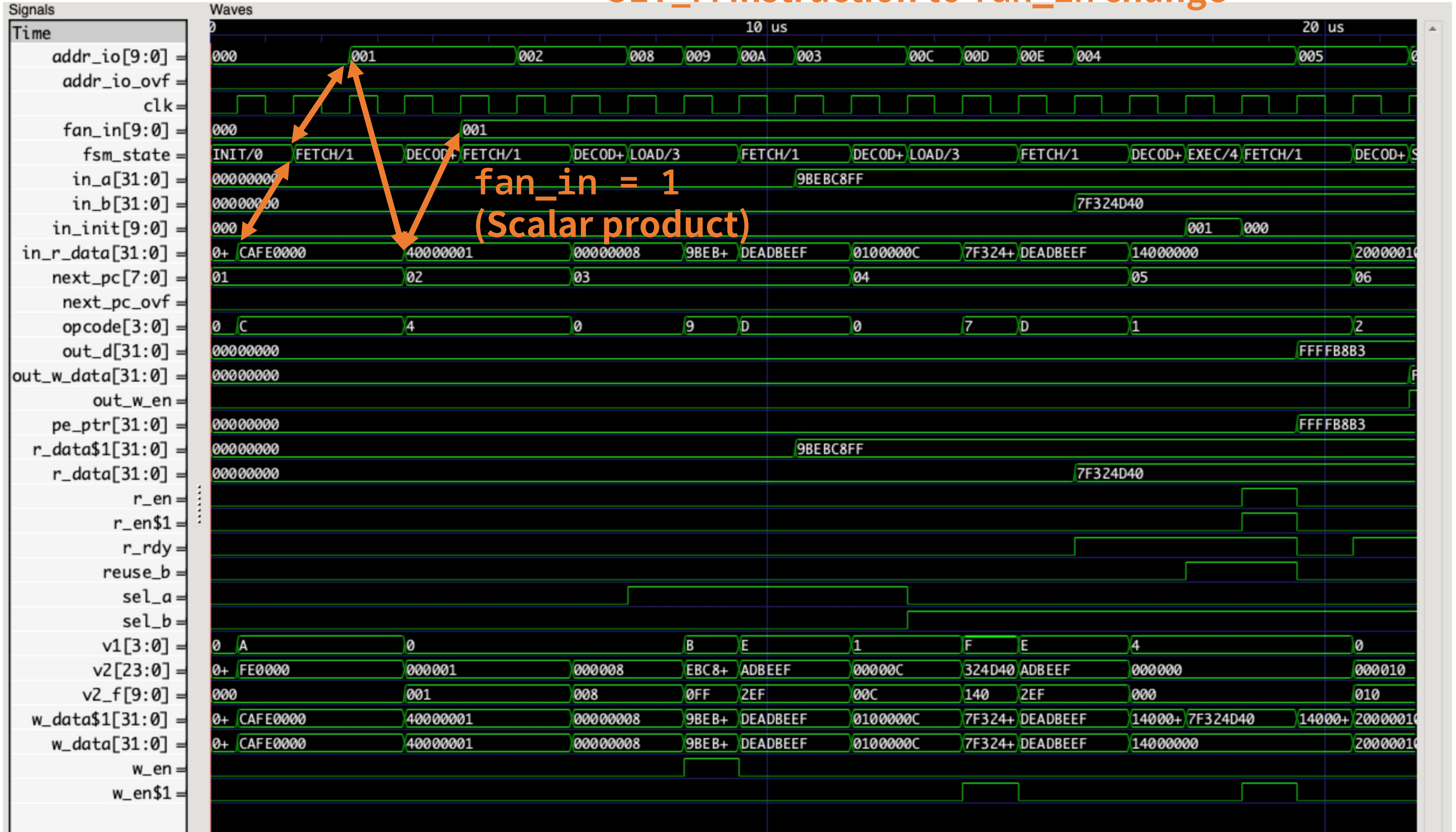
- FLUSH (flush FIFO B)
  - Minimum 1 cycle
  - FLUSH → FETCH



PE Controller

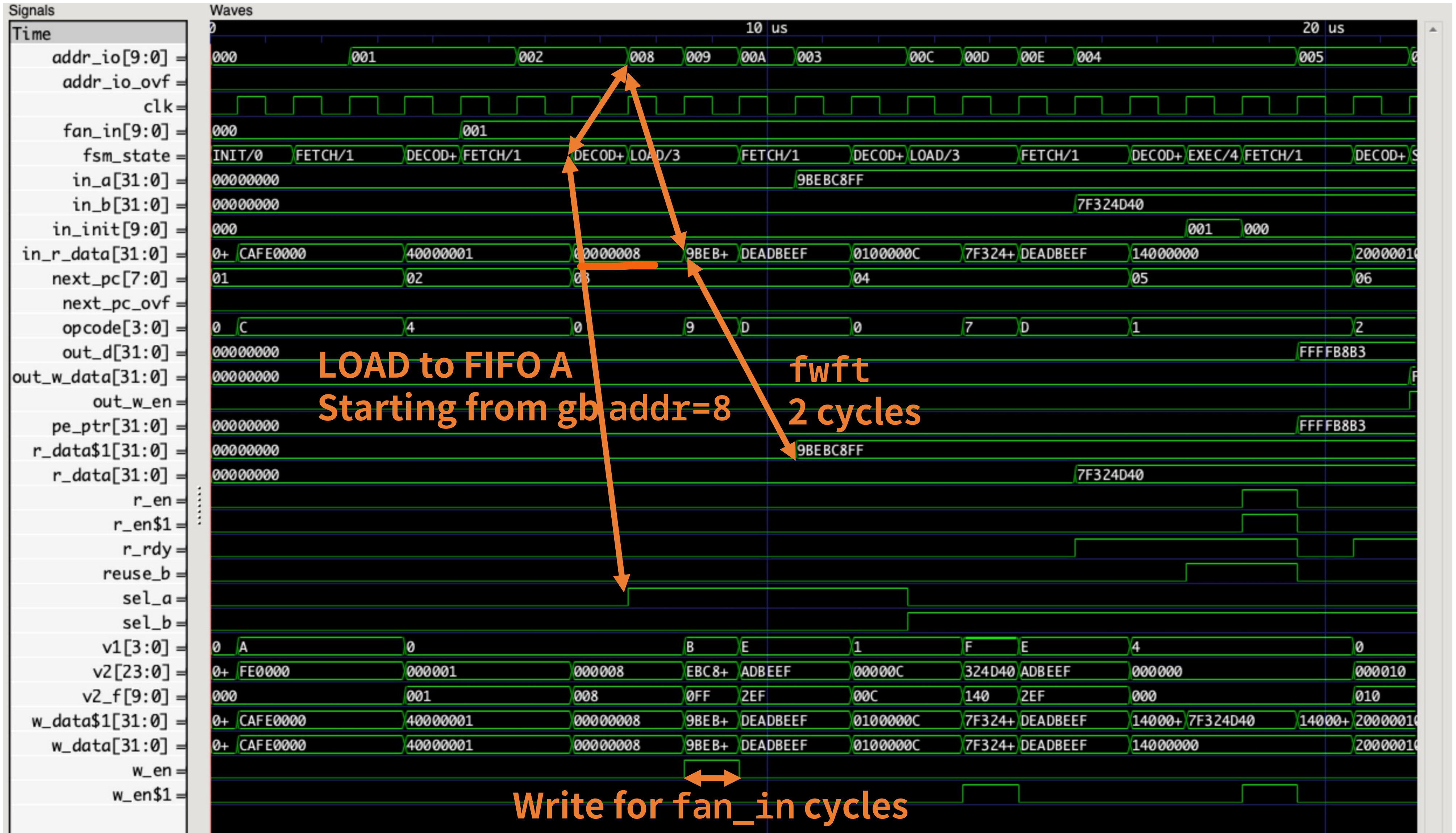
1 cycle delay

- in_r_data=0xCAFE_0000 to INIT → FETCH
- INIT → FETCH to addr_io change
- addr_io change to in_r_data update
- SET_M instruction to fan_in change

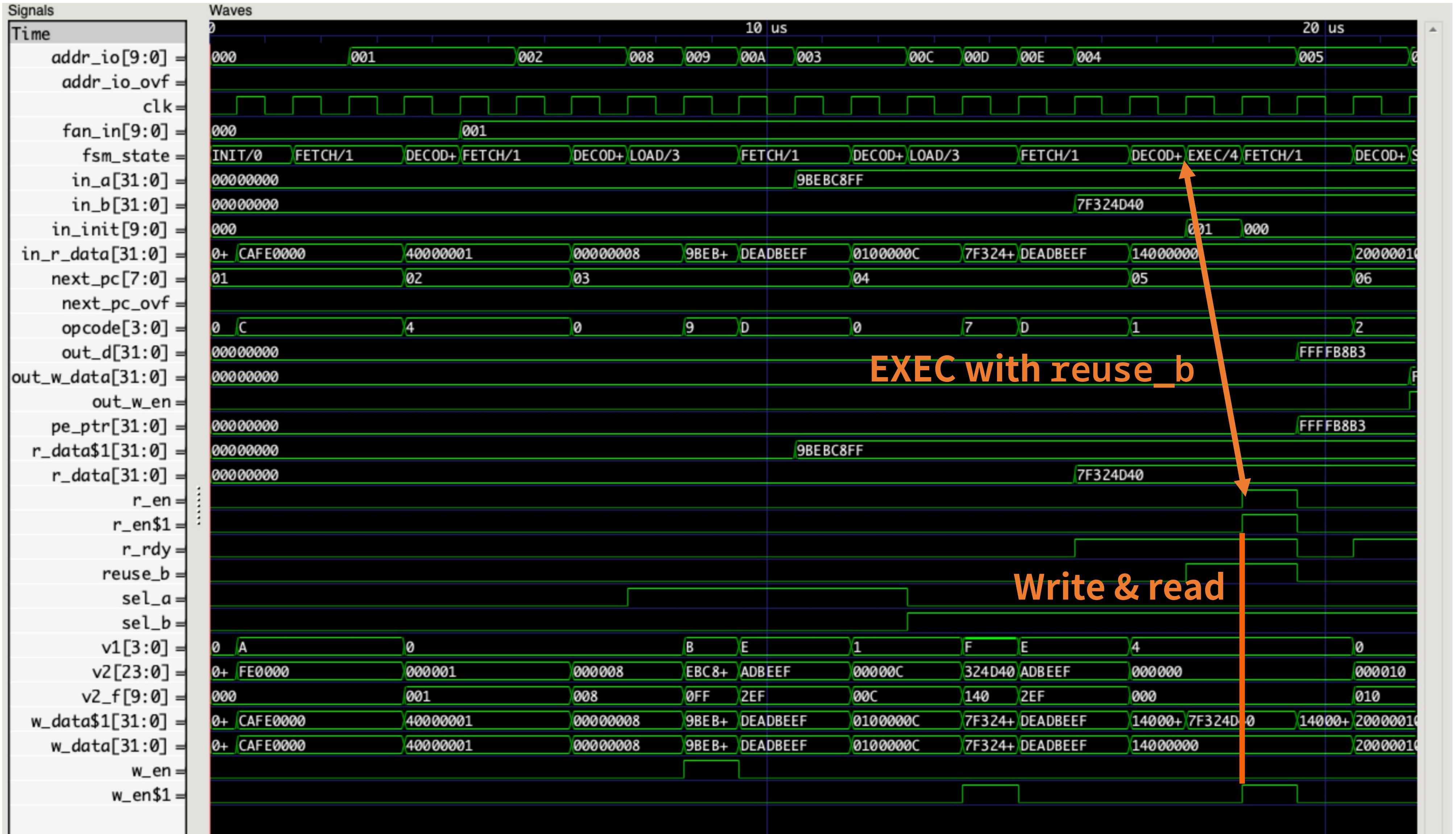


```
addr_io[9:0]      000 | 001 | 002 | 008 | 009 | 00A | 003 | 00C | 00D | 00E | 004 | 005
fan_in[9:0]       000 | 001
fsm_state         INIT/0 | FETCH/1 | DECOD+ | FETCH/1 | DECOD+ | LOAD/3 | FETCH/1 | DECOD+ | LOAD/3 | FETCH/1 | DECOD+ | EXEC/4 | FETCH/1 | DECOD+
in_a[31:0]        00000000 | 9BEBC8FF
in_b[31:0]        00000000 | 7F324D40
in_init[9:0]      000 | 001 | 000
in_r_data[31:0]   0+ | CAFE0000 | 40000001 | 00000008 | 9BEB+ | DEADBEEF | 0100000C | 7F324+ | DEADBEEF | 14000000 | 2000001
next_pc[7:0]      01 | 02 | 03 | 04 | 05 | 06
opcode[3:0]       0 | C | 4 | 0 | 9 | D | 0 | 7 | D | 1 | 2
out_d[31:0]       00000000 | FFFFB8B3
out_w_data[31:0]  00000000
pe_ptr[31:0]      00000000 | FFFFB8B3
r_data$1[31:0]    00000000 | 9BEBC8FF
r_data[31:0]      00000000 | 7F324D40
v1[3:0]           0 | A | 0 | B | E | 1 | F | E | 4 | 0
v2[23:0]          0+ | FE0000 | 000001 | 000008 | EBC8+ | ADBEEF | 00000C | 324D40 | ADBEEF | 000000 | 000010
v2_f[9:0]         000 | 001 | 008 | 0FF | 2EF | 00C | 140 | 2EF | 000 | 010
w_data$1[31:0]    0+ | CAFE0000 | 40000001 | 00000008 | 9BEB+ | DEADBEEF | 0100000C | 7F324+ | DEADBEEF | 14000+ | 7F324D40 | 14000+ | 2000001
w_data[31:0]      0+ | CAFE0000 | 40000001 | 00000008 | 9BEB+ | DEADBEEF | 0100000C | 7F324+ | DEADBEEF | 14000000 | 2000001
```

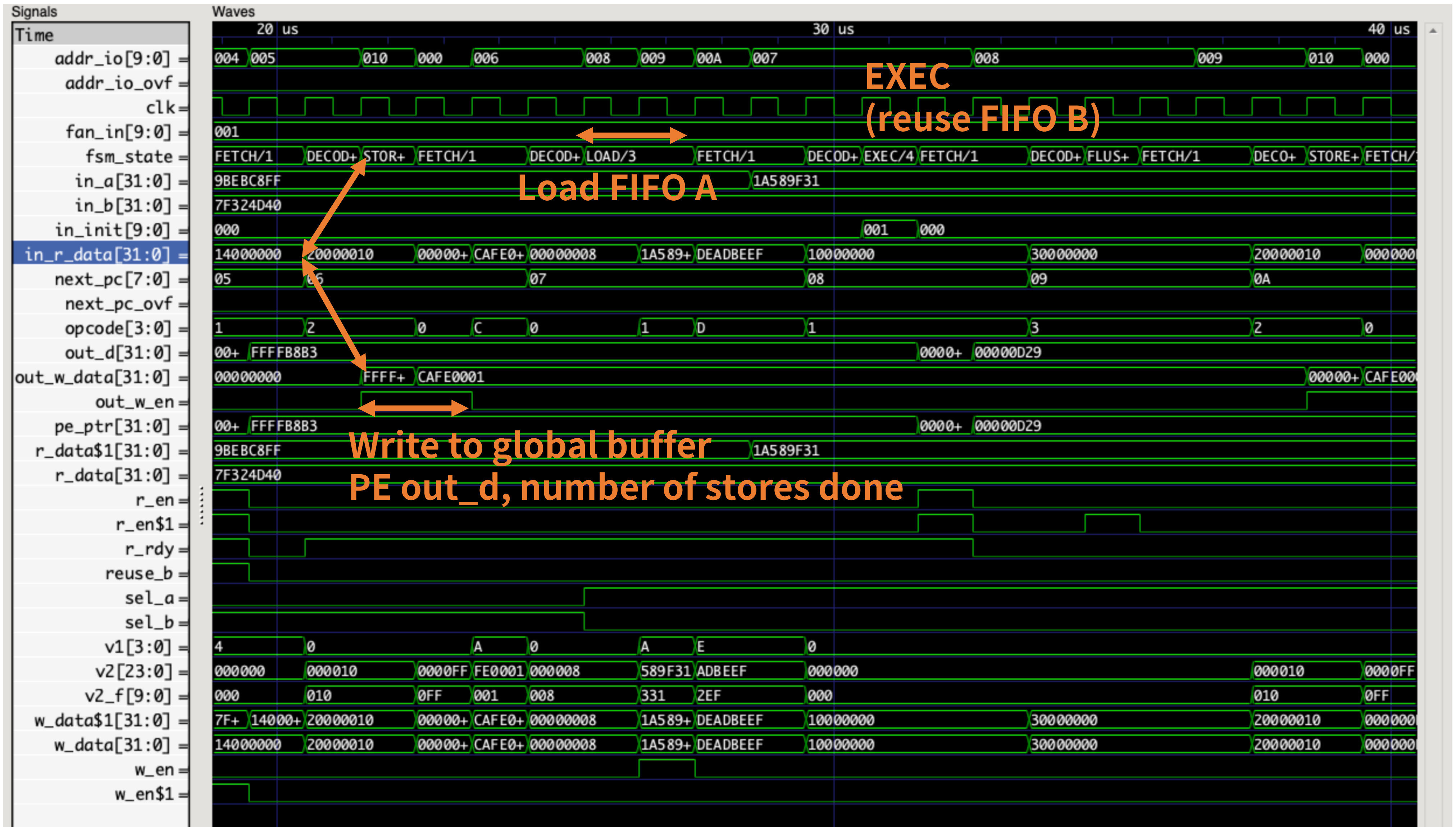
PE Controller



PE Controller



PE Controller



PE Controller

- OK if your code supports only
  - width=32
  - signed=True
- OK if your waveform is different than ours
  - Specification is important
  - If you want to change spec

    (Clarify & validate) your (intention & spec) on (code & report)

- Skeleton code

  https://www.notion.so/skeleton-code-3d33625065d2401083d6f9cf3c66dc41?pvs=4

Report

- Report should
  - be **PDF**
  - be Korean or English
  - explain core points of waveform & code
- Hand in **zip** file containing
  - Report PDF
  - Waveform (vcd file, screenshots)
  - Code (result should be reproducible)
- Due **5/2 23:59 KST** on ETL
  - 1 submission per team
  - ~3 weeks
  - Start early! **Midterm on 4/24**

- Thorough documentation(report) will help
  - team communication
  - yourself in coming weeks

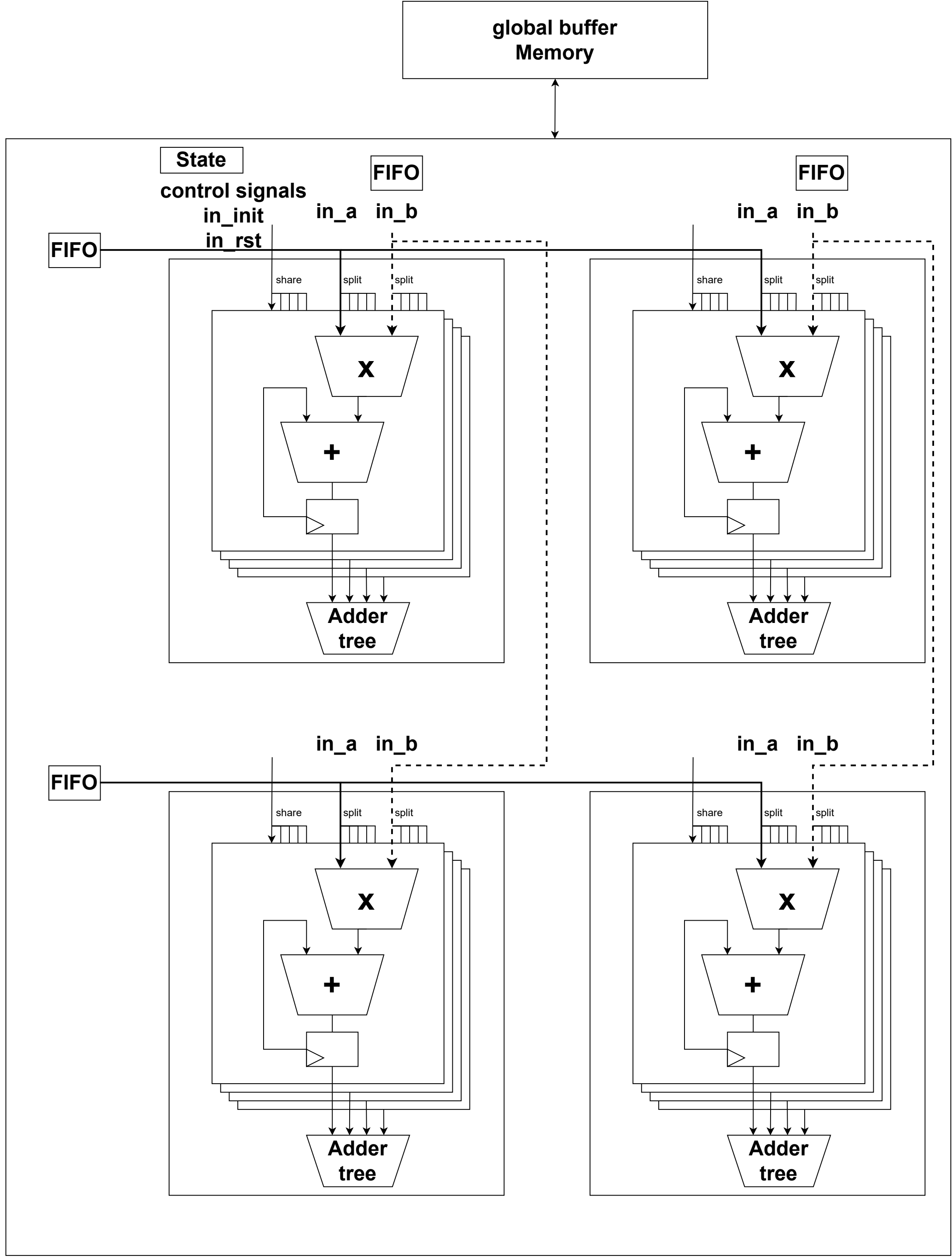
Upcoming practice sessions

- 4/19 Q&A session
- 4/26 midterm review, Q&A session, more on mk1
- Attendance checked

- Use practice sessions effectively!
  - Team communication & implementation
  - Exploit TA

Project sneak peek

- Matrix-matrix outer product
- Extend
  - SET_M
    - Set **`sys_h, sys_w`**
    - MM size you will use
    - 0-base (**`sys_h=sys_w=0`** → 1 by 1)
  - LOAD
    - Data size = `width * fan_in * [sys_h | sys_w]`
  - STORE
    - Data size = **`acc_bits * sys_h * sys_w`**


global buffer
Memory
State
control signals
in_init
in_rst
FIFO
in_a
in_b
share
split
x
+
Adder
tree