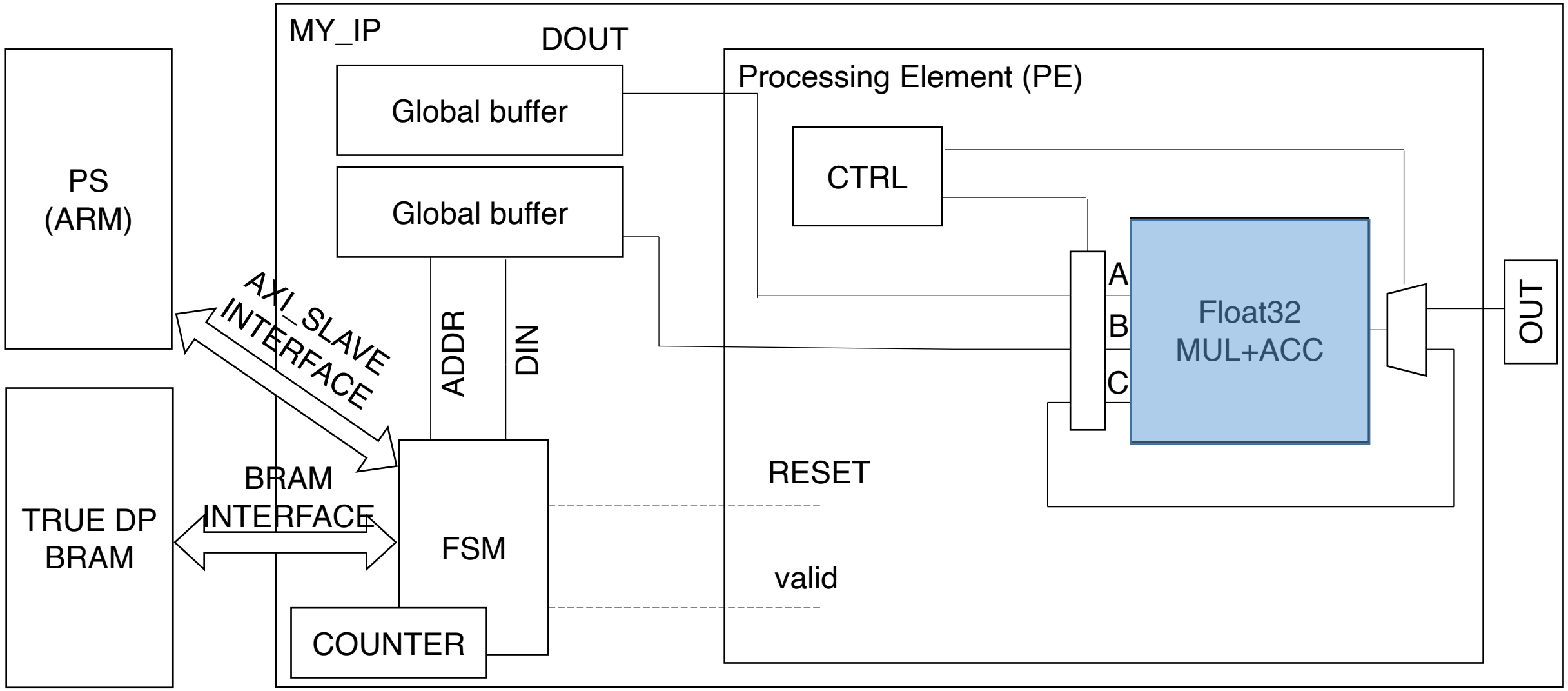
Practice 6

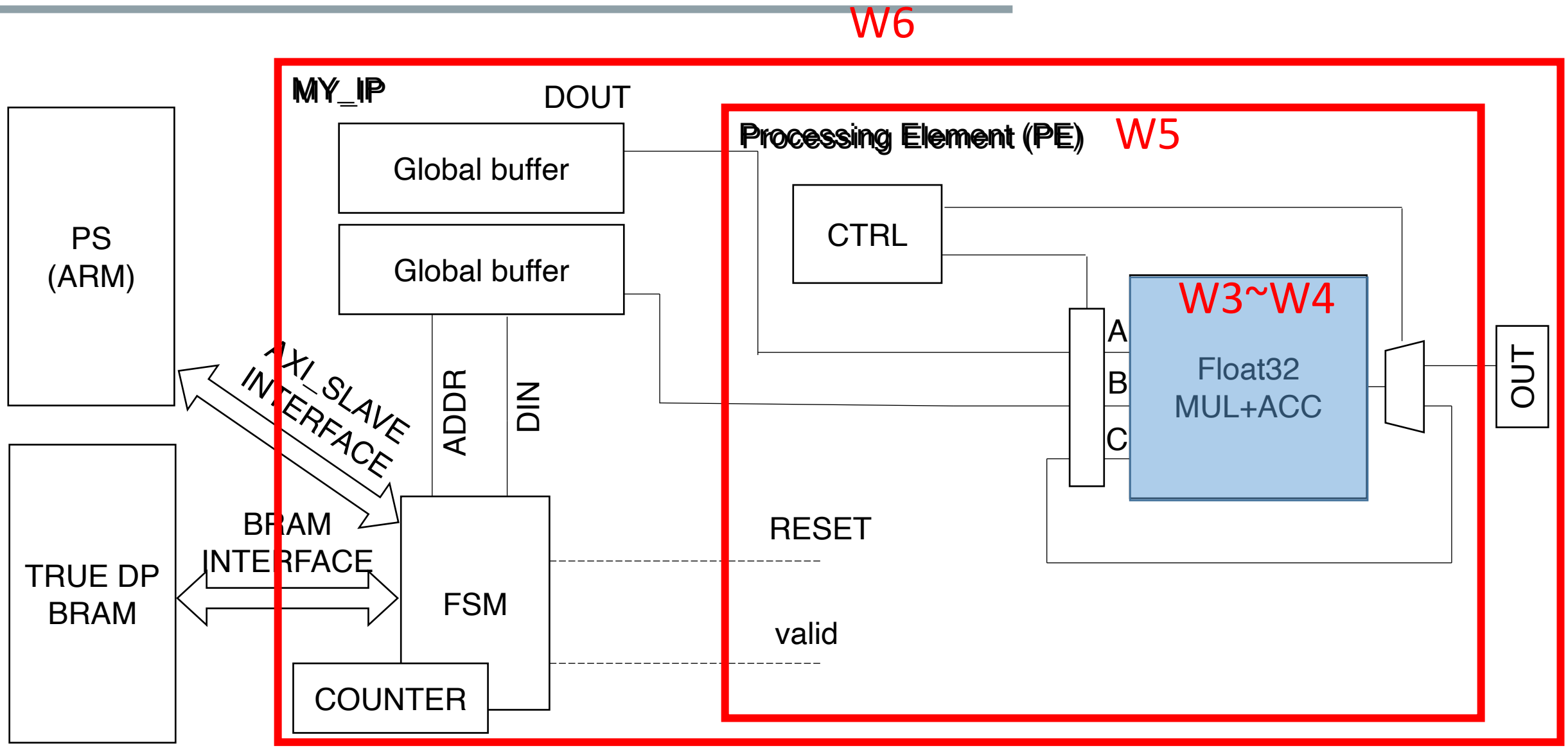
- **BRAM to PE controller**

Computing Memory Architecture
Lab.

Final Project Overview: Matrix Multiplication IP



Final Project Overview: Matrix Multiplication IP



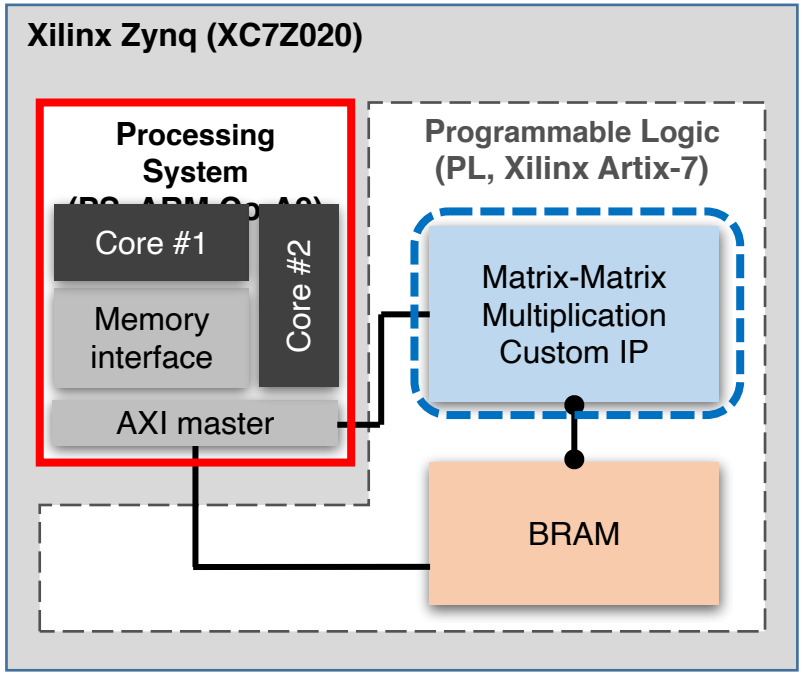
Final Project Overview: Matrix Multiplication IP

- CNN is our application
  - Convolution layer becomes a matrix-matrix multiplication after convolution lowering
    e.g., 32x64 matrix * 64x64 matrix ➔ 32x64 matrix
- ARM CPU runs the main function which calls your MM IP on PL
  - MM for 32x64 weight matrix * 64x64 input matrix multiplication
- BRAM is used for data transfer between SW and HW

MM function on Hardware (Software running on CPU)

```
for(i=0; i<32; i+=1) {
  for(j=0; j<64; j+=1) {
    for(k = 0; k < 64; k++){
      Output[i][j] += Input[i][k]*W[k][j]
    }
  }
}
```

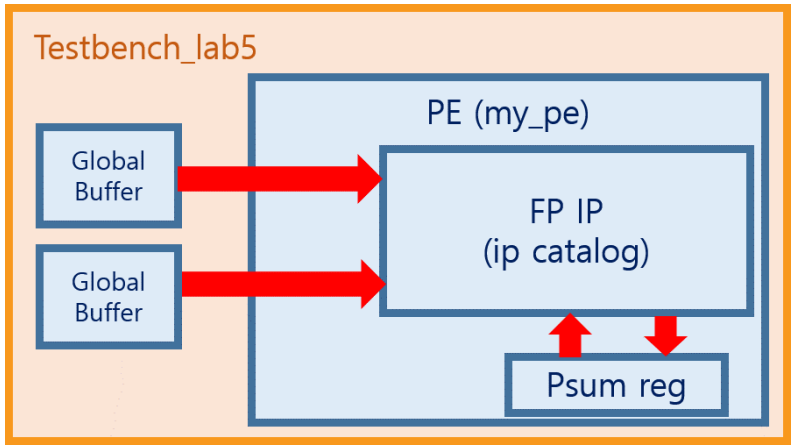
Fused multiply



Main Practice

From Lab 5 to Lab 6

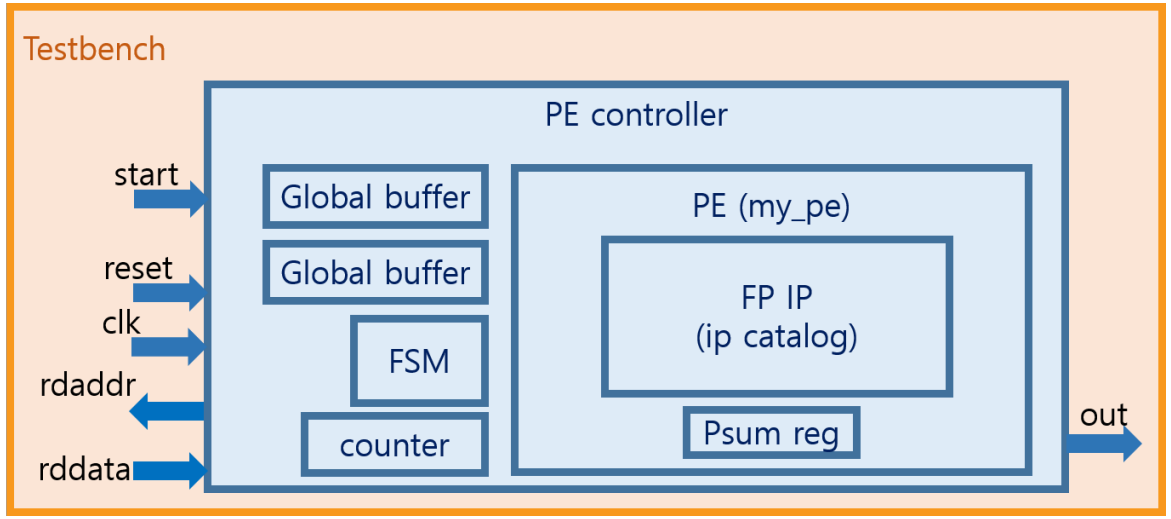
- **Lab 5:**
  - We implemented PE and tested MAC(multiply & accumulate) operation.
    - with 16 sequential inputs from each Global Buffer.



- **Lab 6:**
  - PE controller
    - Controlling PE using FSM

Practice

- **Implementing PE controller**
  - Implement PE controller based on PE you made in Lab5.
  - PE Controller
    - PE controller consists of PE and FSM.
    - FSM controls PE to calculate inner product (e.g., data1[0]*data2[0] + data1[1]*data2[1]+ … + data1[15]*data2[15]) with several states.
      - Inner product can be made with MAC operation of PE

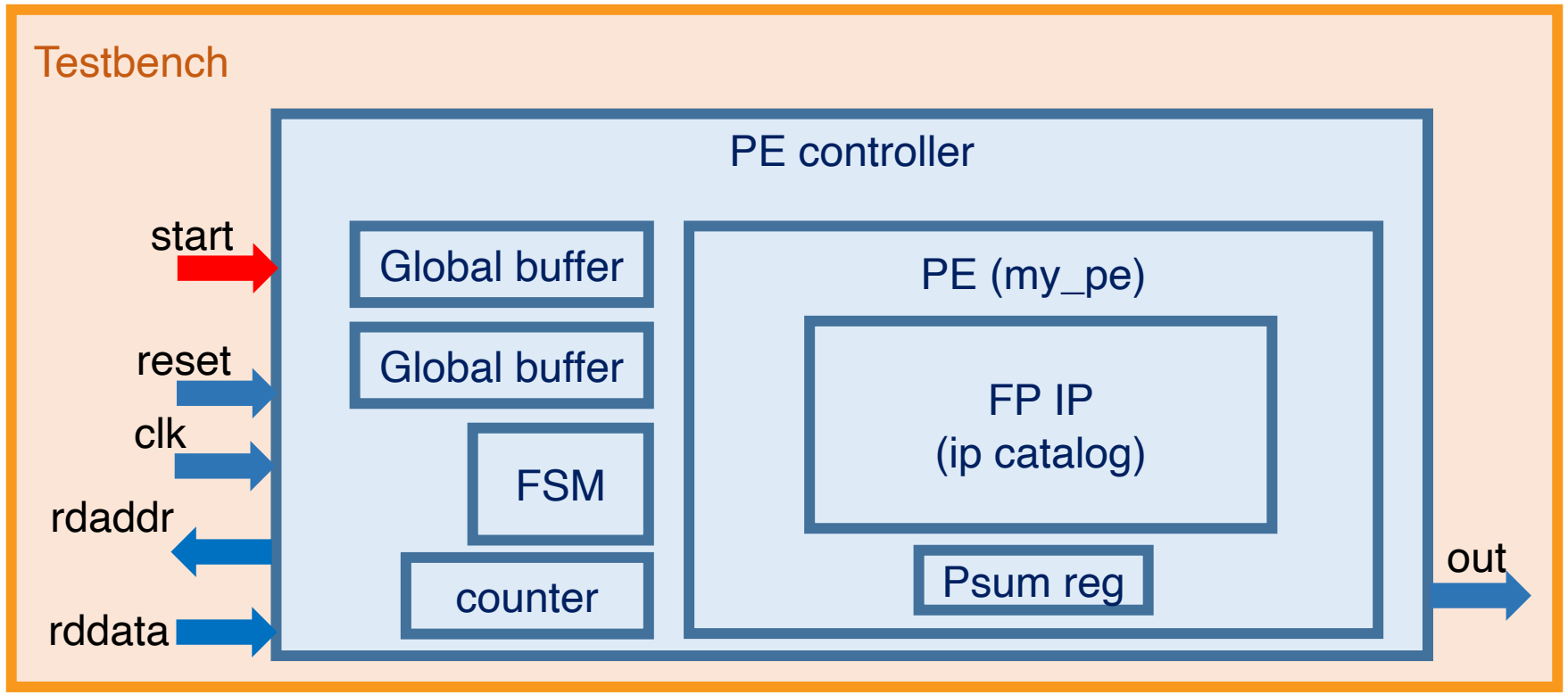


Practice

- **PE Controller FSM states**
  - **S_IDLE:**
    - Idle state. Starts operation with state transition to S_LOAD, when input ‘start==1’.
  - **S_LOAD:**
    - Loads each 16 data into each global buffer.
    - After completion of data loading, state moves to S_CALC.
  - **S_CALC:**
    - Calculate “inner product” with 16 data from one global buffer, 16 data from the other global buffer.
    - After calculation, state moves to S_DONE.
  - **S_DONE:**
    - Gives ‘done’ as 1.
    - After a few cycles(ex. 5 cycles), state moves to S_IDLE

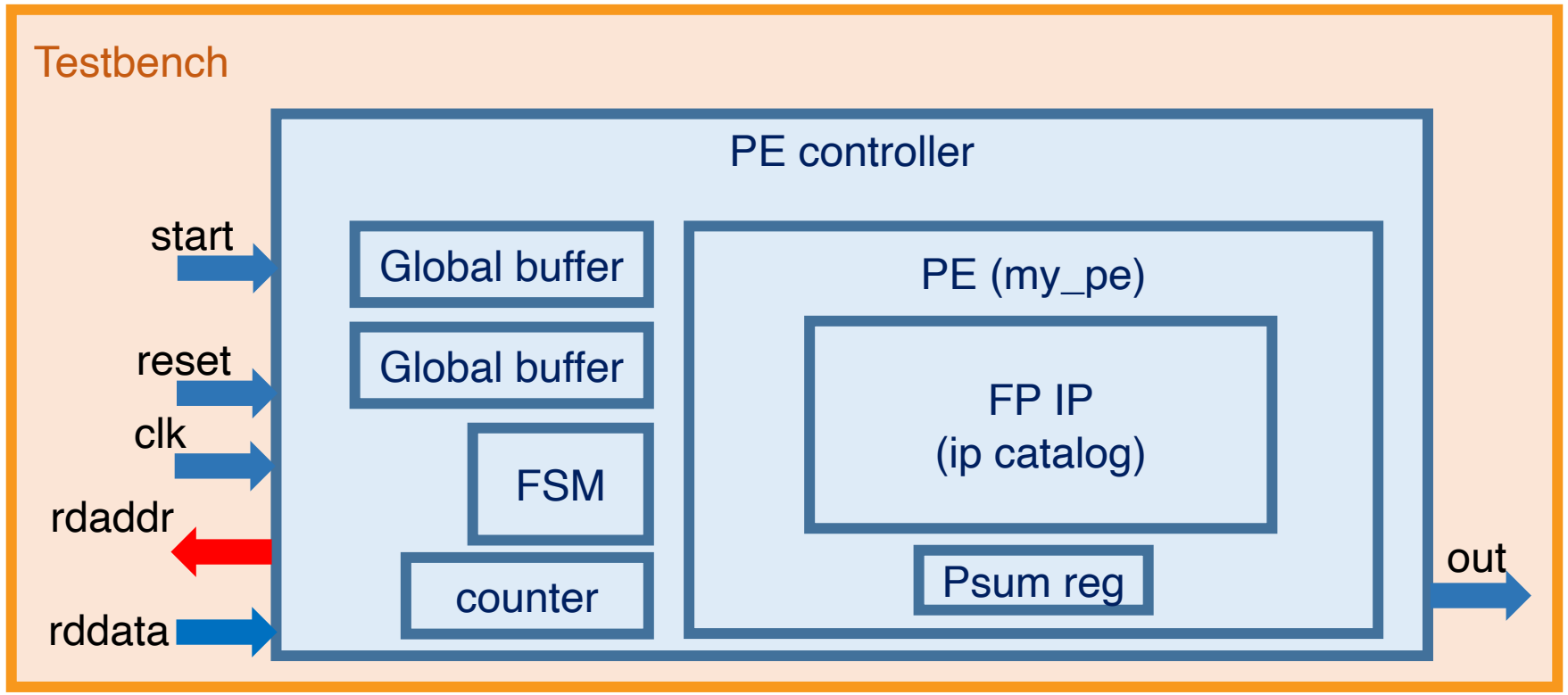
Idle state

- **S_IDLE:**
  - Idle state. Starts operation with state transition to S_LOAD, when input ‘start==1’.



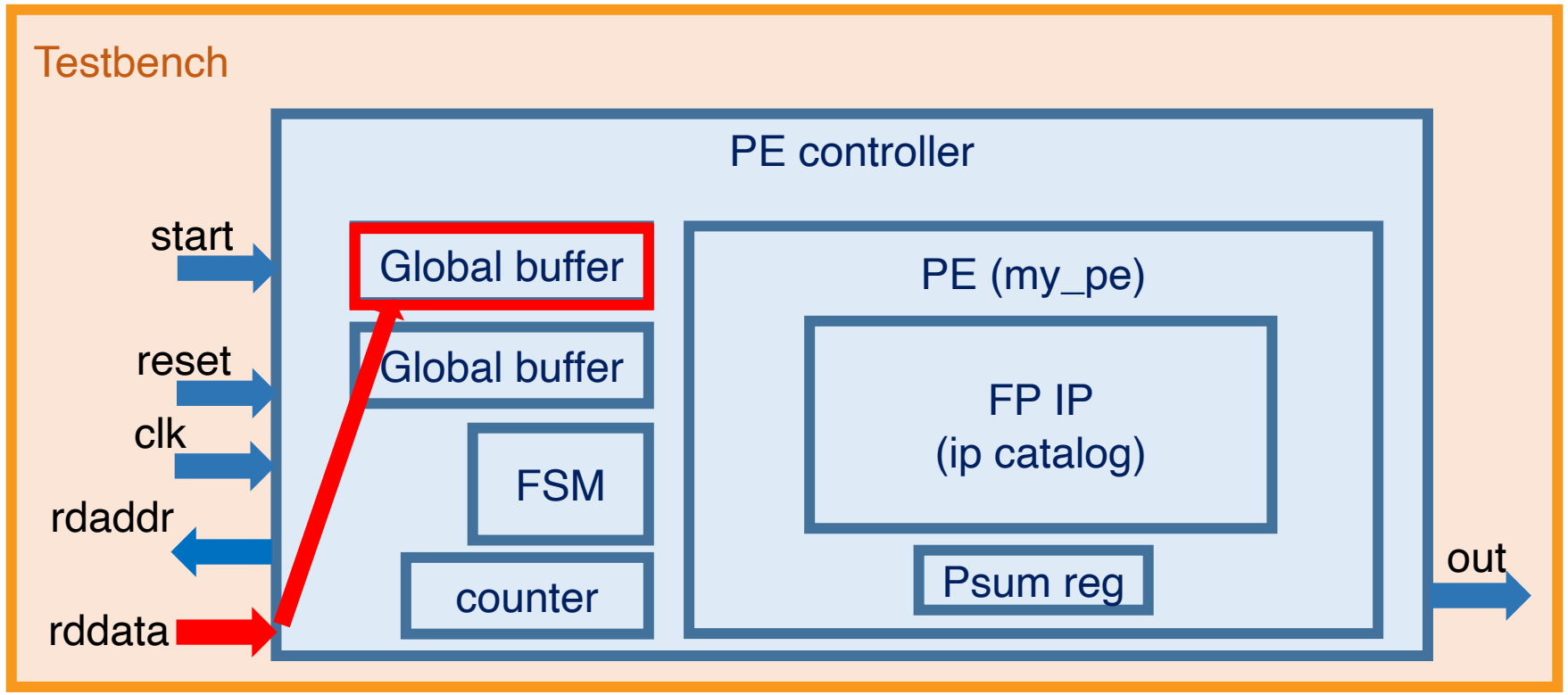
Load state

- **S_LOAD:**
  - Loads each 16 data into each global buffer.
  - After completion of data loading, state moves to S_CALC.



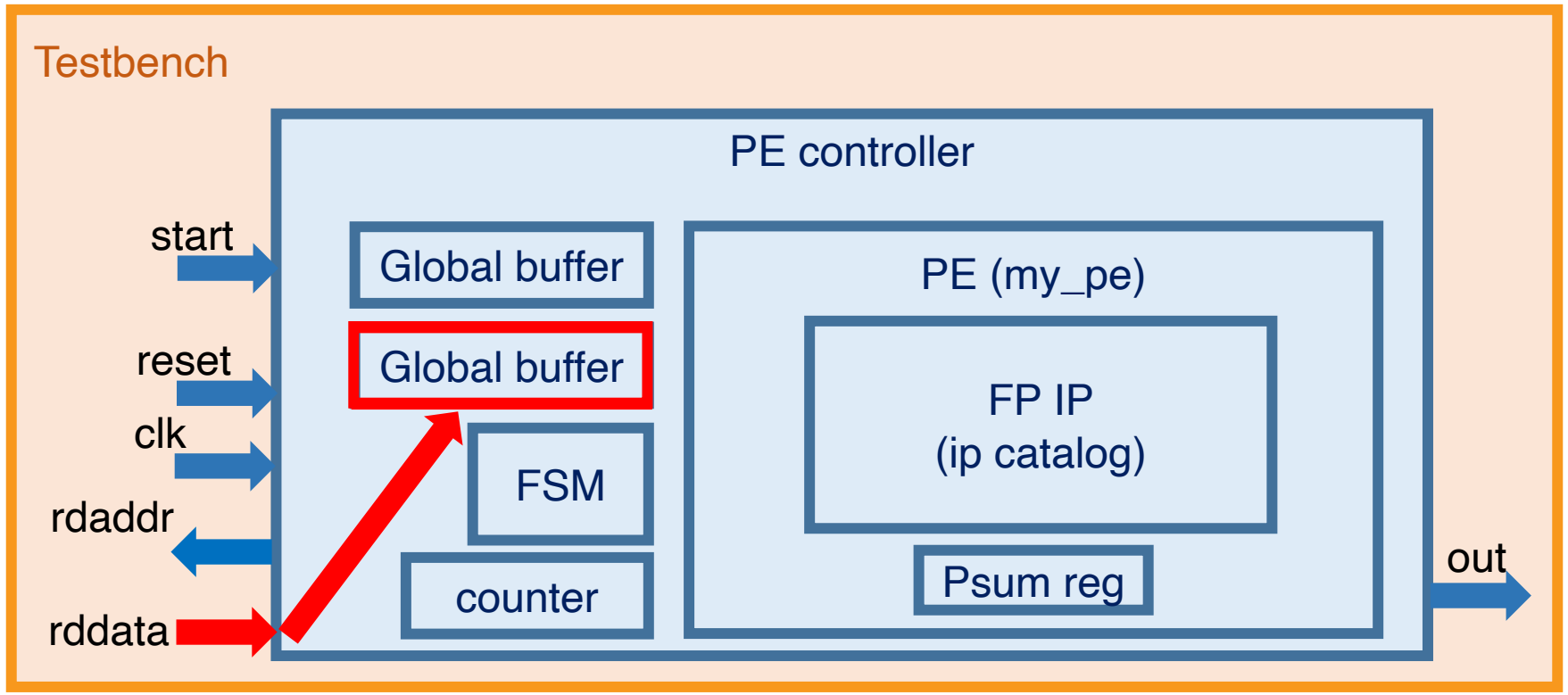
Load state

- **S_LOAD:**
  - Loads each 16 data into each global buffer.
  - After completion of data loading, state moves to S_CALC.



Load state

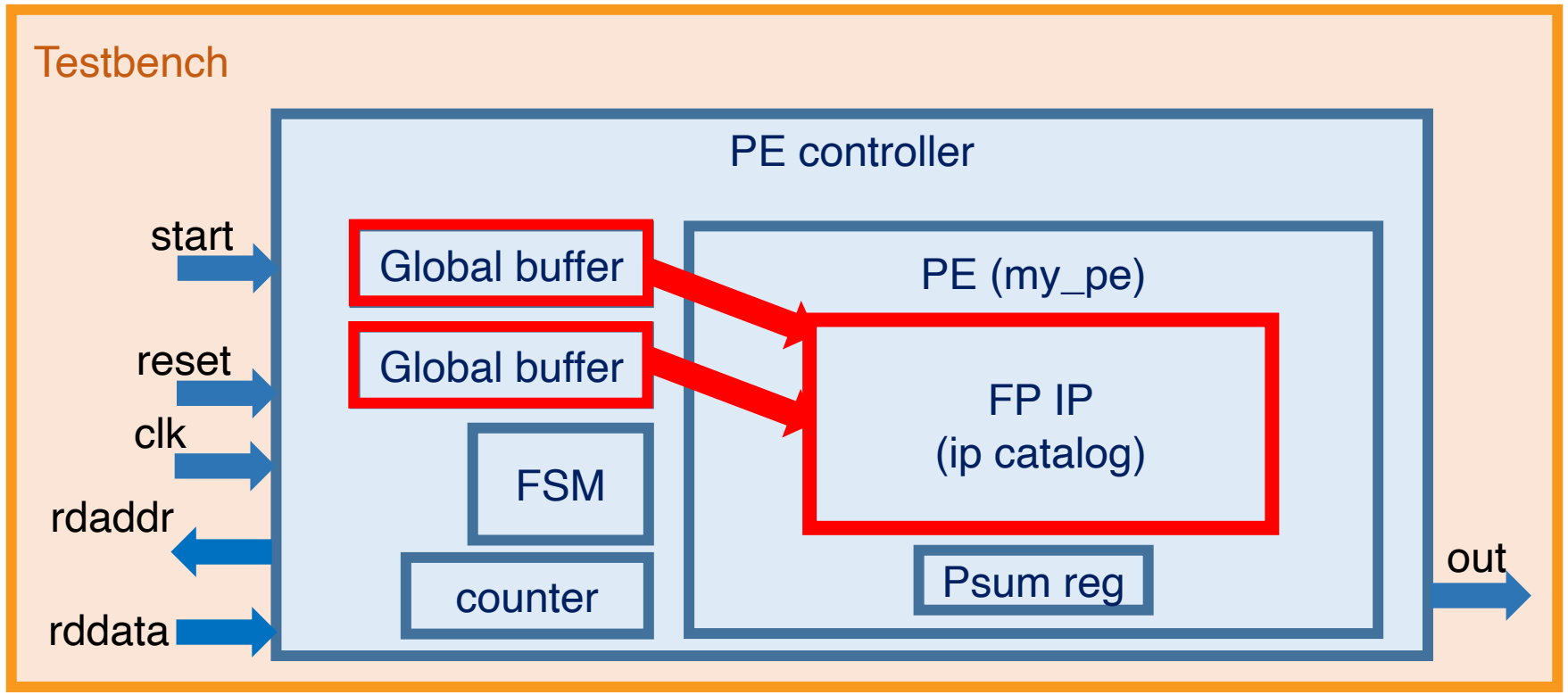
- **S_LOAD:**
  - Loads each 16 data into each global buffer.
  - After completion of data loading, state moves to S_CALC.



Calc state

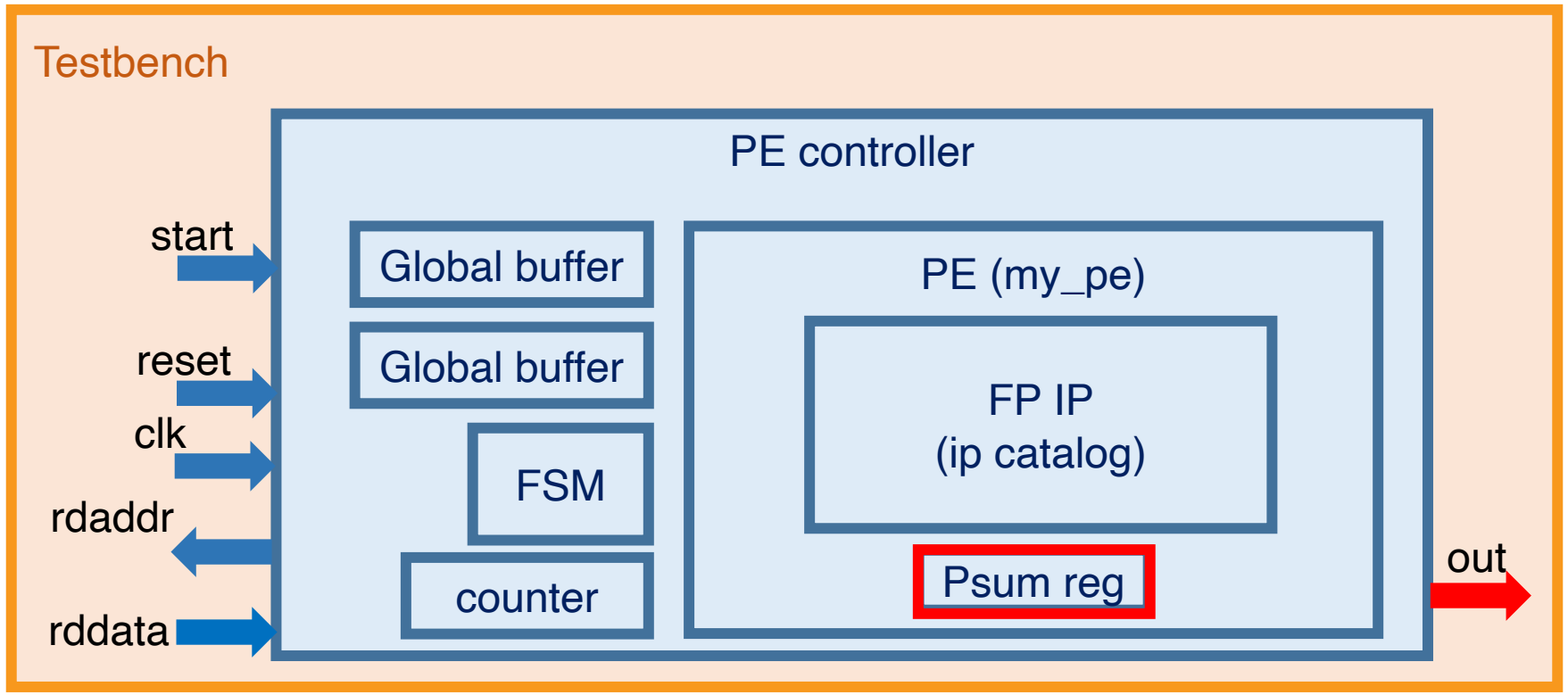
- **S_CALC:**
  - Calculate “inner product” with 16 data from one global buffer, 16 data from the other global memory.
  - After calculation, state moves to S_DONE.

$$result = (a_{in} * b_{in}) + result$$



Done state

- **S_DONE:**
  - Gives ‘done’ as 1.
  - After 5 cycles, state moves to S_IDLE



Homework

- Requirements
  - Result
    - Attach your project folder with all your verilog codes (e.g., PE Controller, test bench)
    - Attach your PE Controller waveform(simulation result) with [student_number, name]
      - Test the scenario in slide[9~14].
      - The correct waveform should be shown to confirm the operation of your code.
      - Refer to Practice3 about screenshot.
  - Report
    - Explain operation of PE Controller with waveform that you implemented
    - In your own words
    - Either in Korean or in English
    - # of pages does not matter
    - **PDF only!!**
  - **Result + Report to one .zip file**
- Upload (.zip) file on ETL
  - Submit one (.zip) file
    - zip file name : [Lab06]name1_name2.zip (ex : [Lab06]홍길동_홍동길.zip)
  - Due: 4/21(WED) 23:59
    - **No Late Submission**