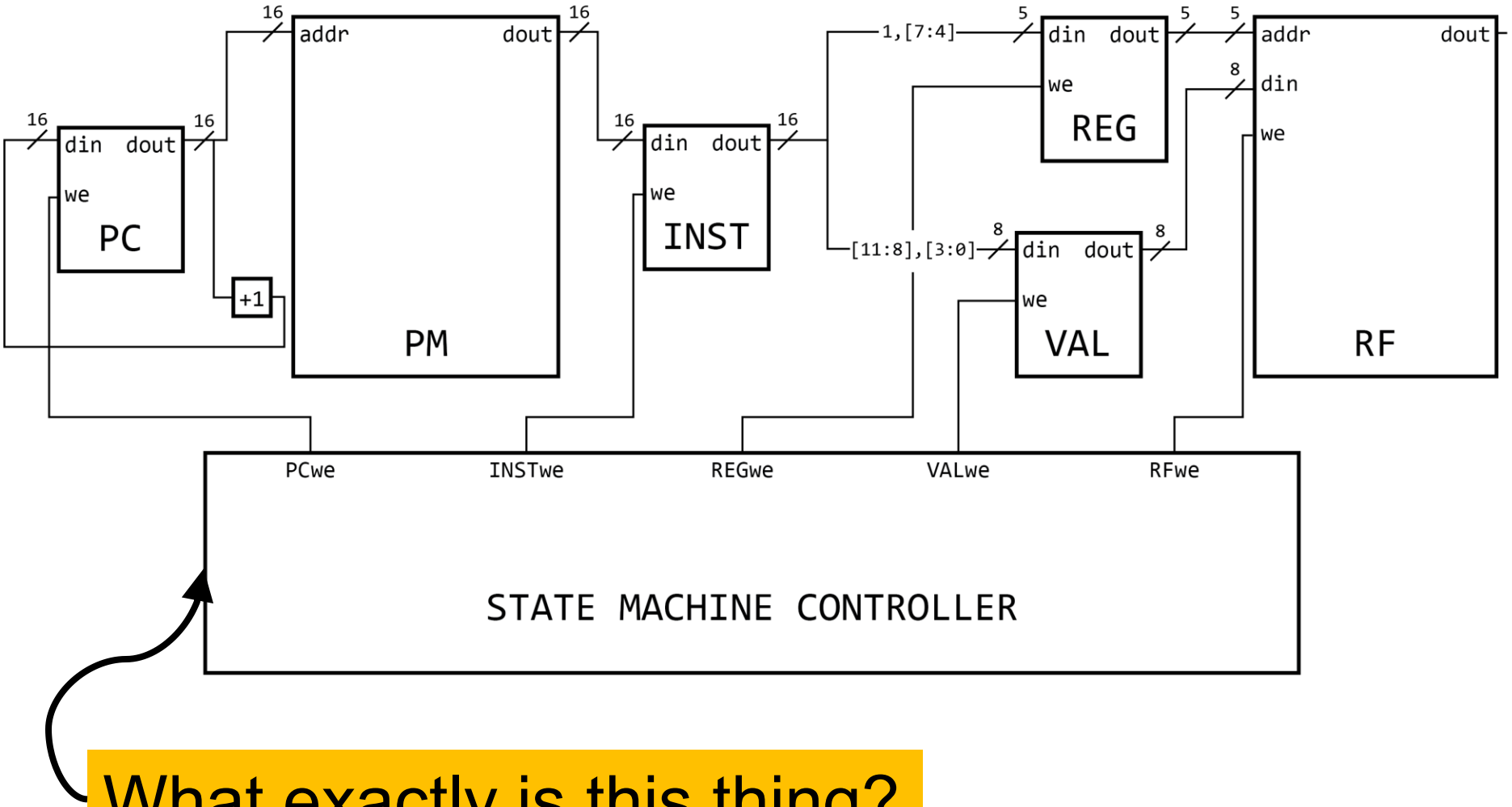
Today

- Last week
  - State machine
  - Circuit blocks
  - LDI computer
- Today
  - The entire machine!
  - Little bit more on idea of clocks

LDI Computer



What exactly is this thing?

The box implements the state transition table

	PC_we	INST_we	REG_we	VAL_we	RF_we
INST=PM[PC]	0	1	0	0	0
REG=1,INST[7:4]	0	0	1	0	0
VAL=INST[11:8],INST[3:0]	0	0	0	1	0
RF[REG]=VAL	0	0	0	0	1
PC=PC+1	1	0	0	0	0

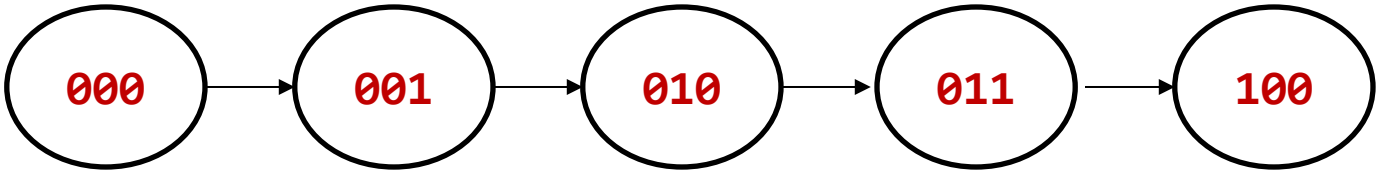
How to implement this as circuit blocks?

How is state transition diagram implemented?

1. States are represented as numbers
2. Build circuit blocks that emit outputs based on the state number
3. Build circuit blocks that perform logic for transitioning to next state

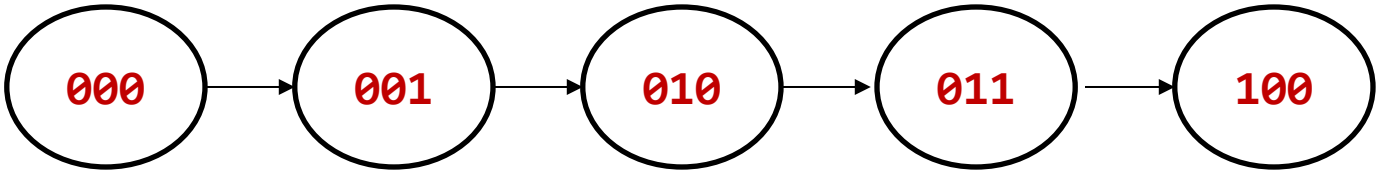
1) State transition table

$S_2S_1S_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0



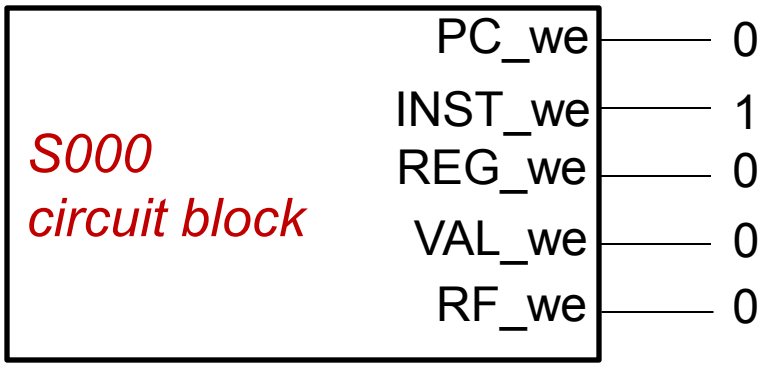
2) State transition table

$S_2S_1S_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0



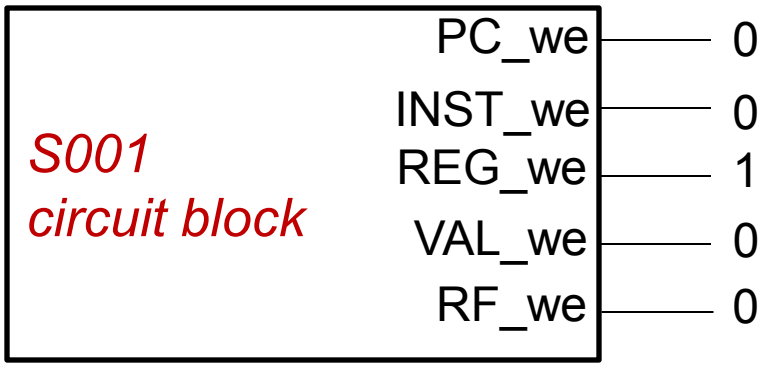
2a) State transition table

$s_2 s_1 s_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0



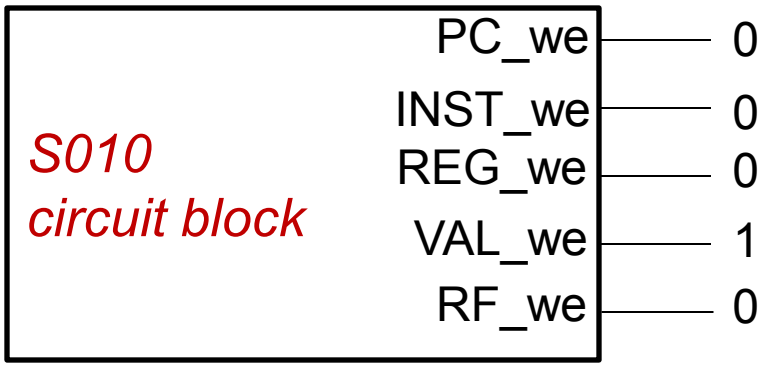
2b) State transition table

$S_2S_1S_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0



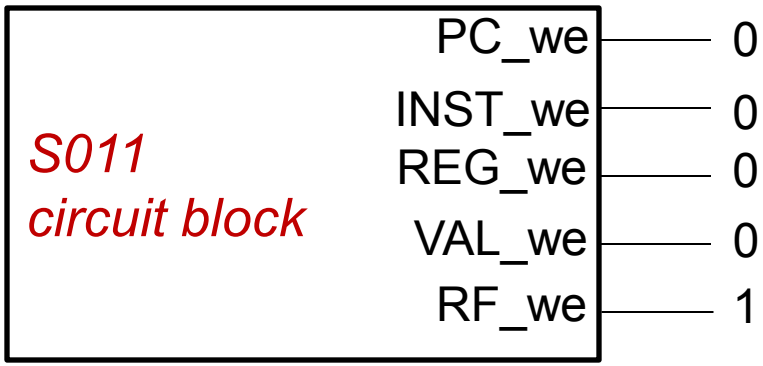
2c) State transition table

$S_2S_1S_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0



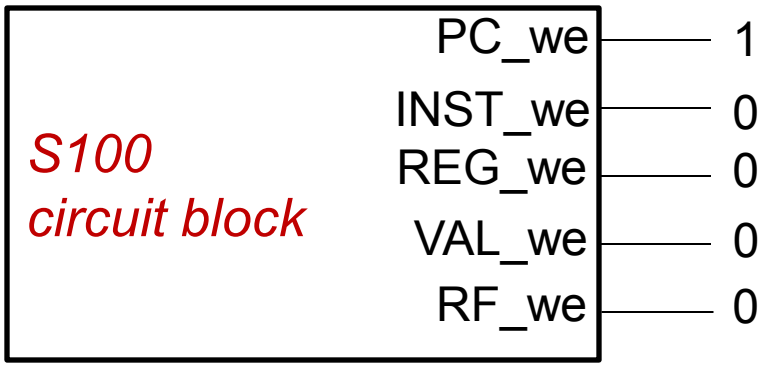
2d) State transition table

$S_2S_1S_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0

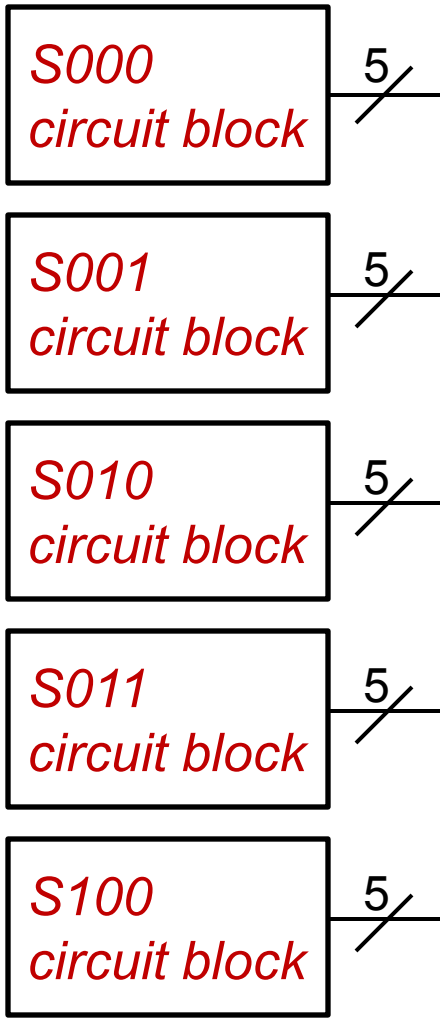


2e) State transition table

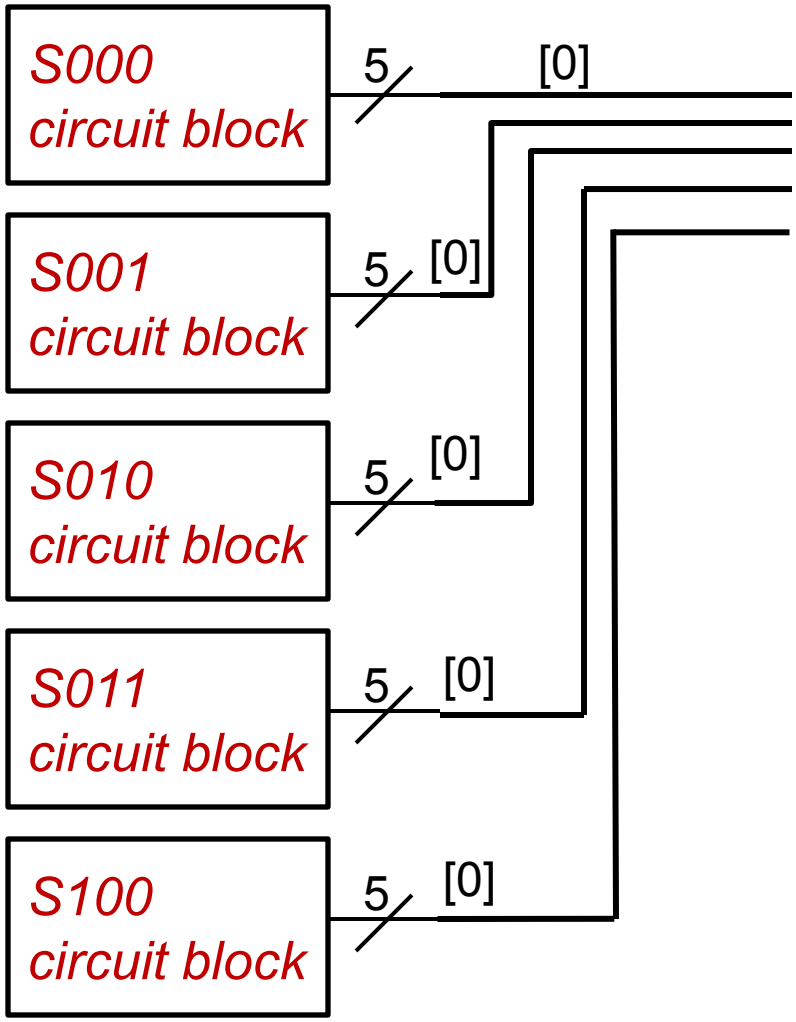
$S_2S_1S_0$		PC_we	INST_we	REG_we	VAL_we	RF_we
000	INST=PM[PC]	0	1	0	0	0
001	REG=1,INST[7:4]	0	0	1	0	0
010	VAL=INST[11:8],INST[3:0]	0	0	0	1	0
011	RF[REG]=VAL	0	0	0	0	1
100	PC=PC+1	1	0	0	0	0



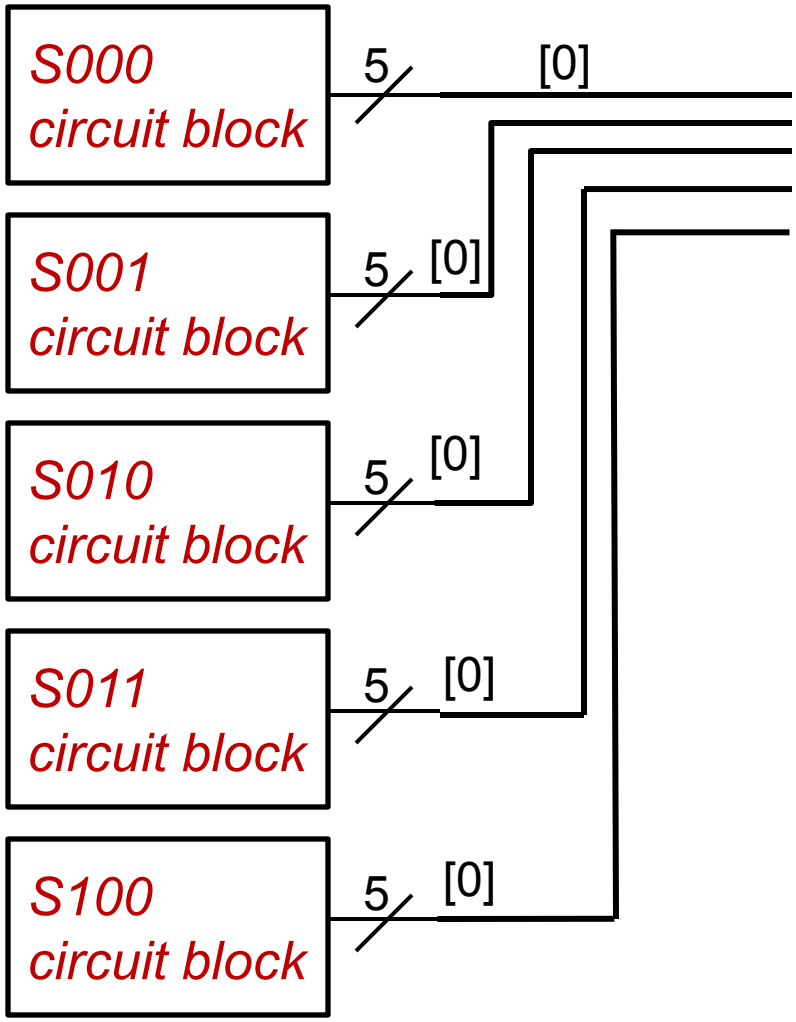
Putting it together

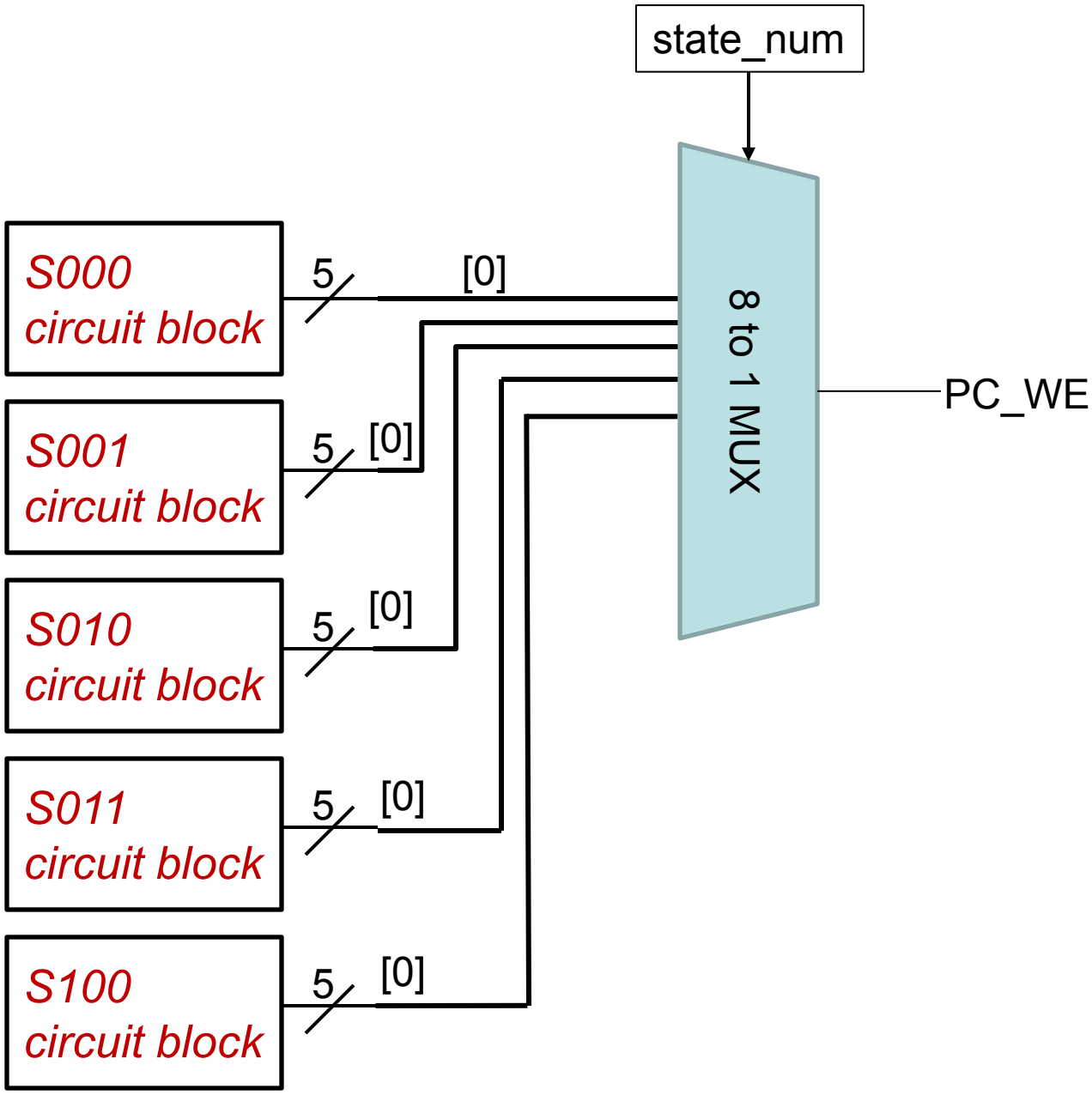


Putting it together



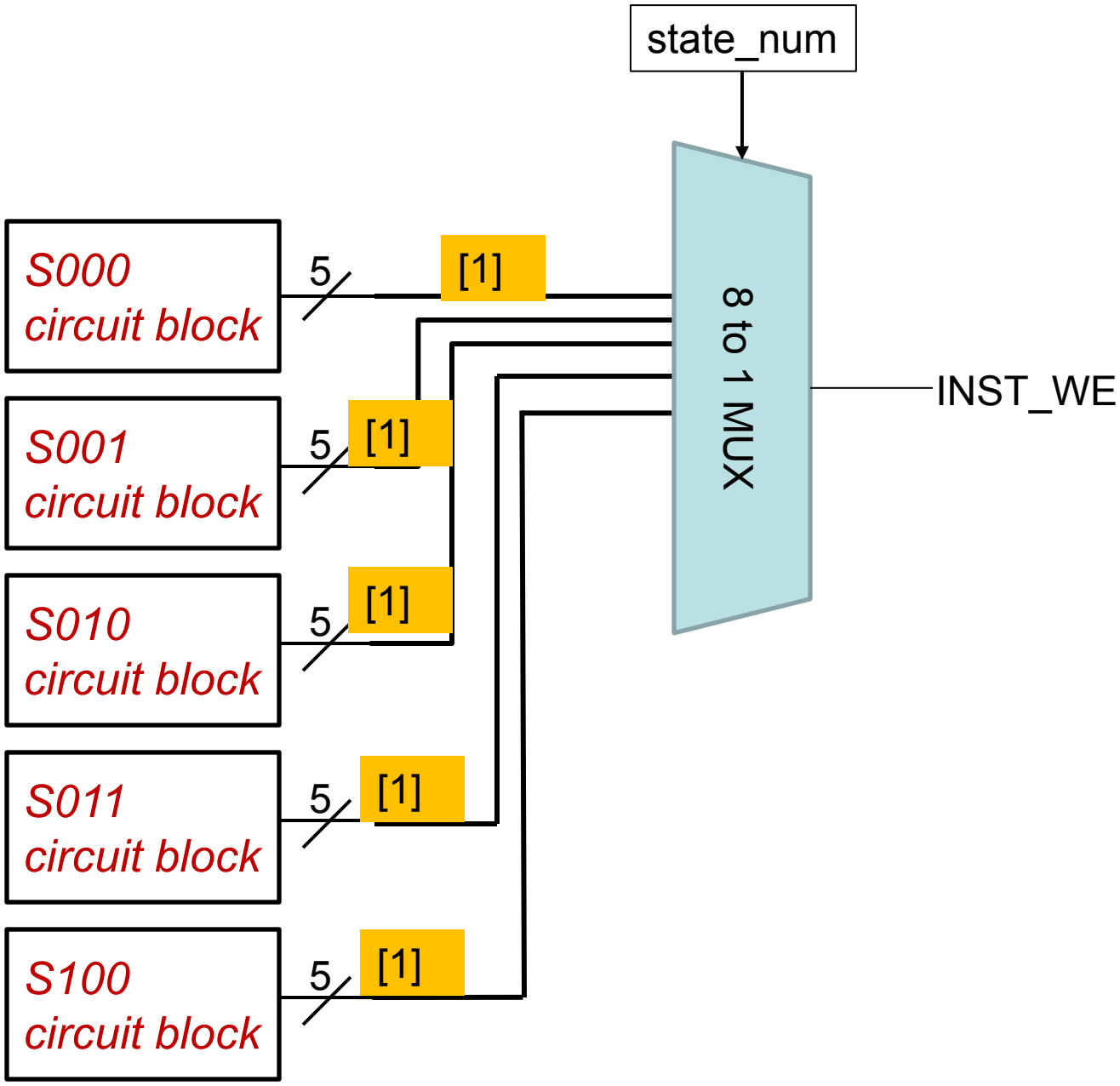
Putting it together



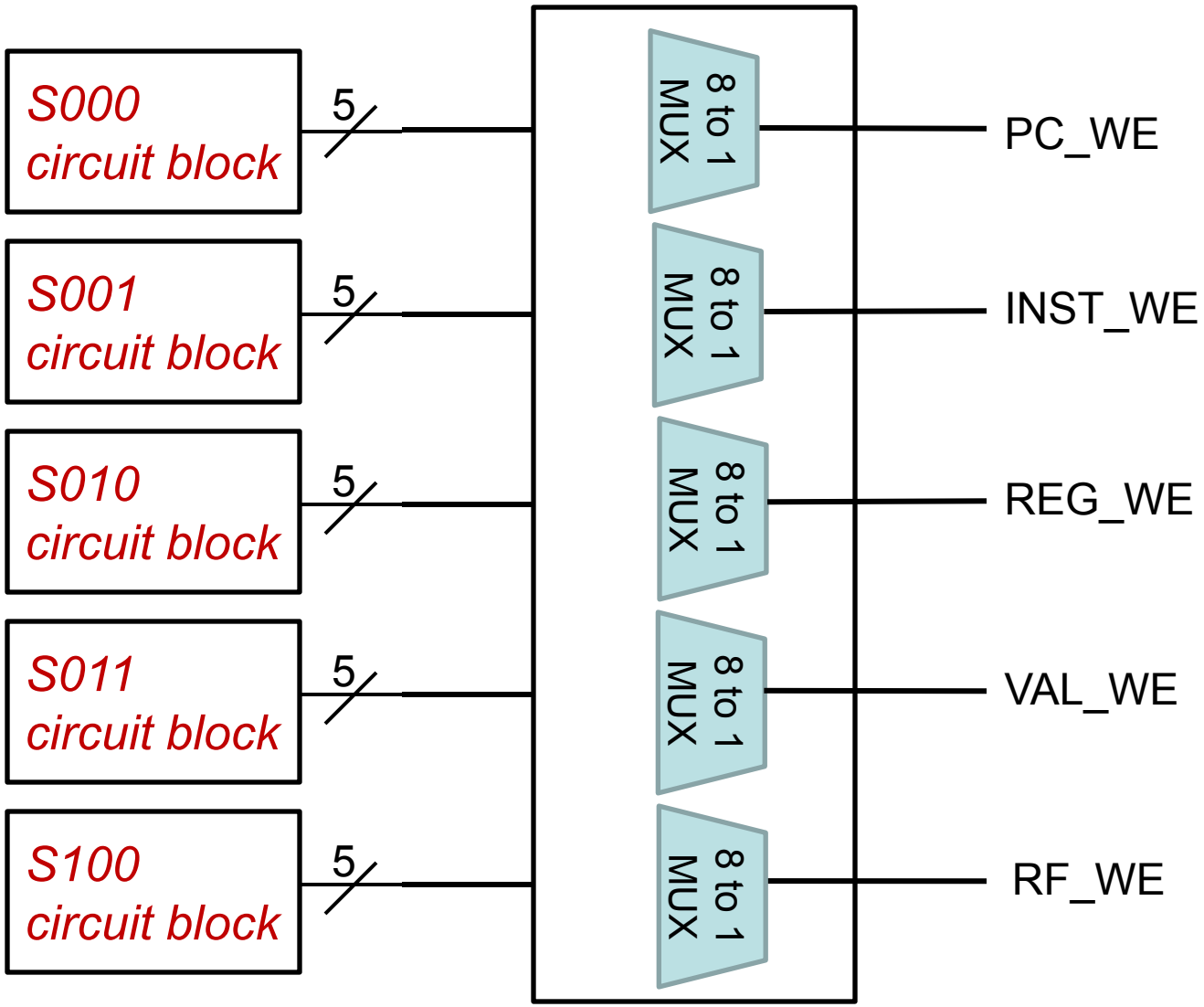

state_num
S000
circuit block
S001
circuit block
S010
circuit block
S011
circuit block
S100
circuit block
5
[0]
8 to 1 MUX
PC_WE

We have done one control signal

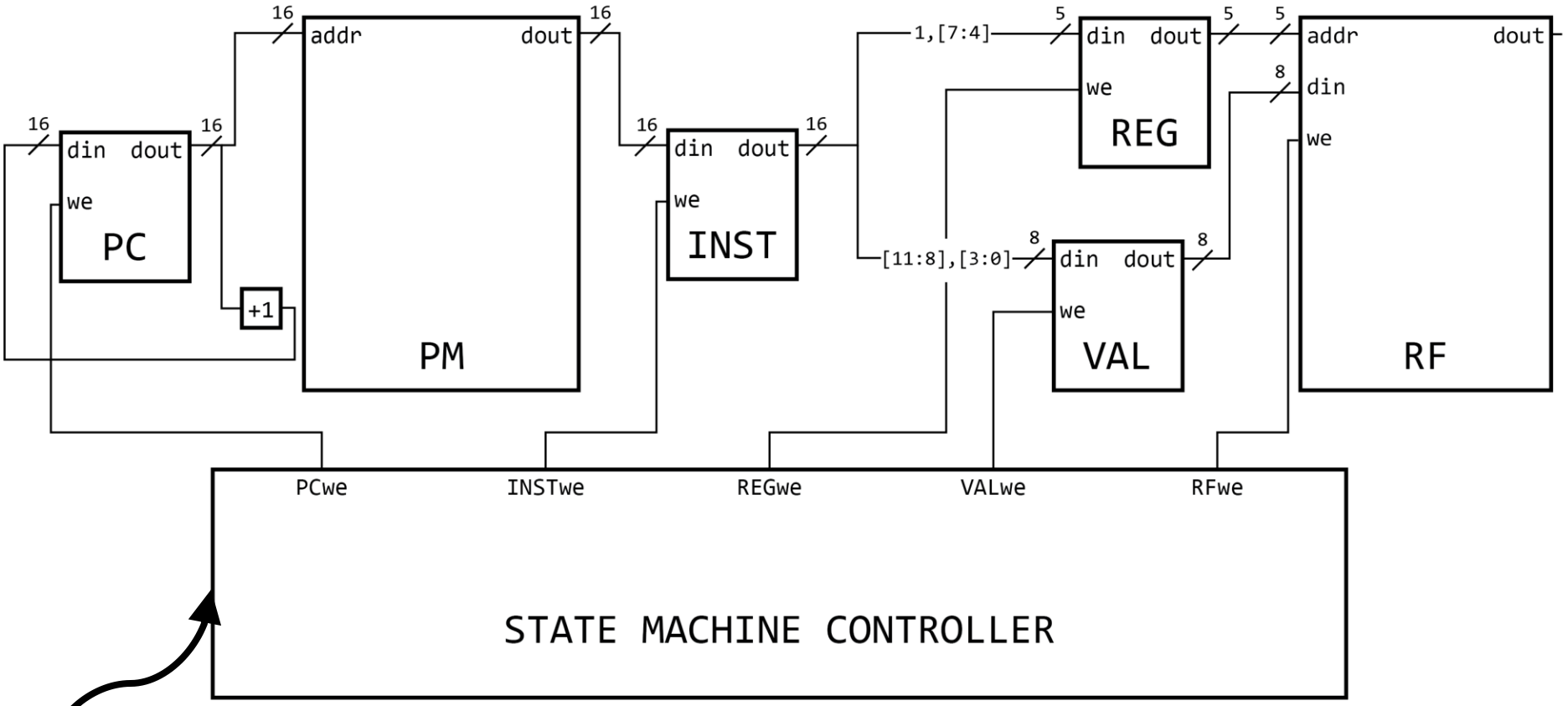
Repeat process for each


state_num
S000
circuit block
S001
circuit block
S010
circuit block
S011
circuit block
S100
circuit block
5
5
5
5
5
[1]
[1]
[1]
[1]
[1]
8 to 1 MUX
INST_WE

Putting it all together

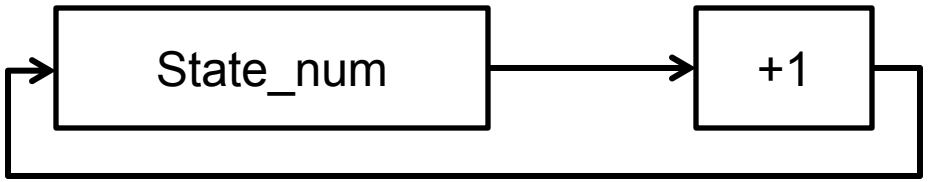


LDI Computer



That is exactly is this thing!

3) Next state logic

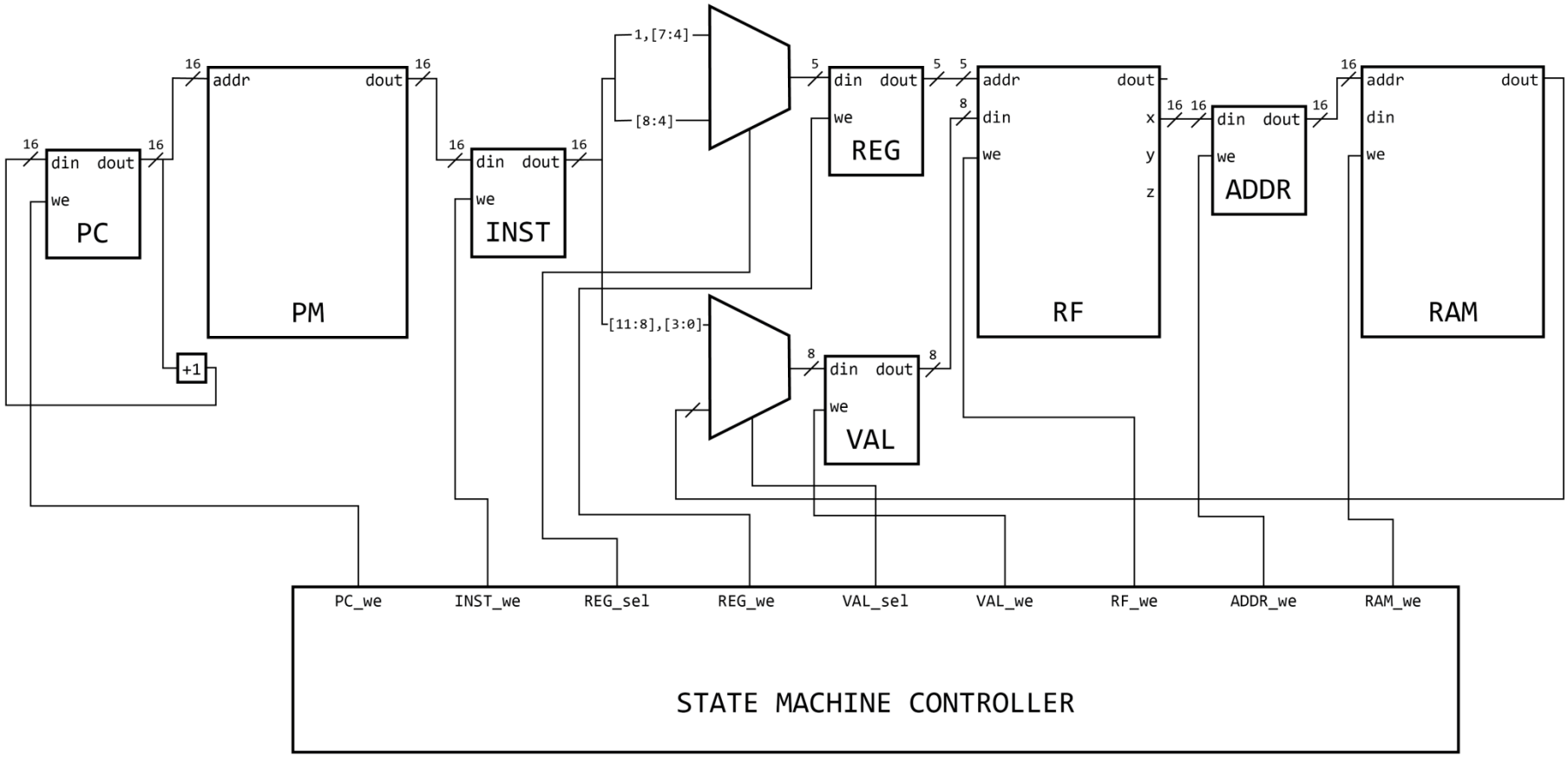



WooHoo!!!

How is state transition diagram implemented?

1. States are represented as numbers
2. Build circuit blocks that emit outputs based on the state number
3. Build circuit blocks that perform logic for transitioning to next state

We will do this for each instruction


PC
din
dout
we
PM
addr
dout
INST
din
dout
we
+1
1,[7:4]
[8:4]
[11:8],[3:0]
REG
VAL
RF
addr
din
we
dout
x
y
z
ADDR
RAM
PC_we
INST_we
REG_sel
REG_we
VAL_sel
VAL_we
RF_we
ADDR_we
RAM_we
STATE MACHINE CONTROLLER

	PC_we	INST_we	REG_we	REG_sel	VAL_we	VAL_sel	RF_we	ADDR_we	RAM_we
INST=PM[PC]	0	1	0		0		0	0	0
REG=1,INST[7:4]	0	0	1	0	0		0	0	0
REG=INST[8:4]	0	0	1	1	0		0	0	0
ADDR=X	0	0	0		0		0	1	0
VAL=INST[11:8],INST[3:0]	0	0	0		1	0	0	0	0
VAL=RAM[ADDR]	0	0	0		1	1	0	0	0
RF[REG]=VAL	0	0	0		0		1	0	0
PC=PC+1	1	0	0		0		0	0	0

Today

- Last week
  - State machine
  - Circuit blocks
  - LDI computer
- Today
  - The entire machine!
  - Little bit more on idea of clocks

So far:

- State machine
- Circuit blocks
- LDI computer

Today

- The machine
- Clocks!

So, what is this state machine?

- It is an implementation of the state transition table!
- How?
    - States are numbers
    - Build circuit blocks that emit outputs based on the state number
        - e.g. 5 states, number them 0, 1, 2, 3, 4
    - Build circuit blocks that perform logic for transitioning to next state
    - Build a circuit block for each state too
        - A logical 1 is just supply voltage
        - All these have no inputs, but constantly output something
    - We run all of these lines from each stage circuit block out together
    - These lines go in to a MUX!
    - Keep pulling out wires and MUX'ing them to get each signal out
- Now, we need the next state logic
    - We just need to increment the state_num every cycle.

So now, timing

- We need things to get to the reg's before we do the operations
- Next state logic gets more complicated, needing the opcode too
- The outputs from the states are more, but not much different besides increasing in size.
- When do we stop or make sure we don't go to the next stage before the current stage is done?
    - Only write the new state_num on the rising clock edge
- Moore's law! Twice as fast for the same circuit
    - For friday: parallel blocks
-