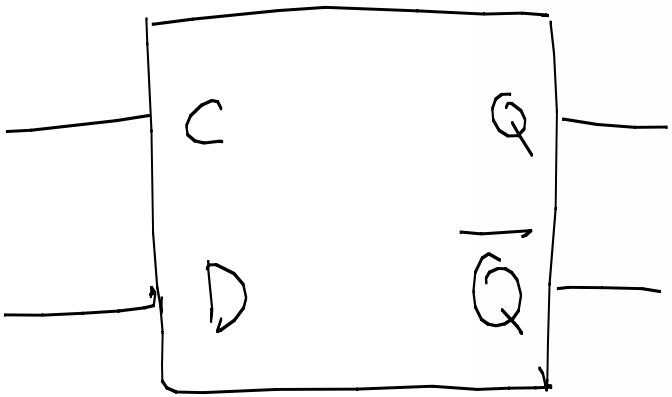
lecture 6

Sequential circuits 2

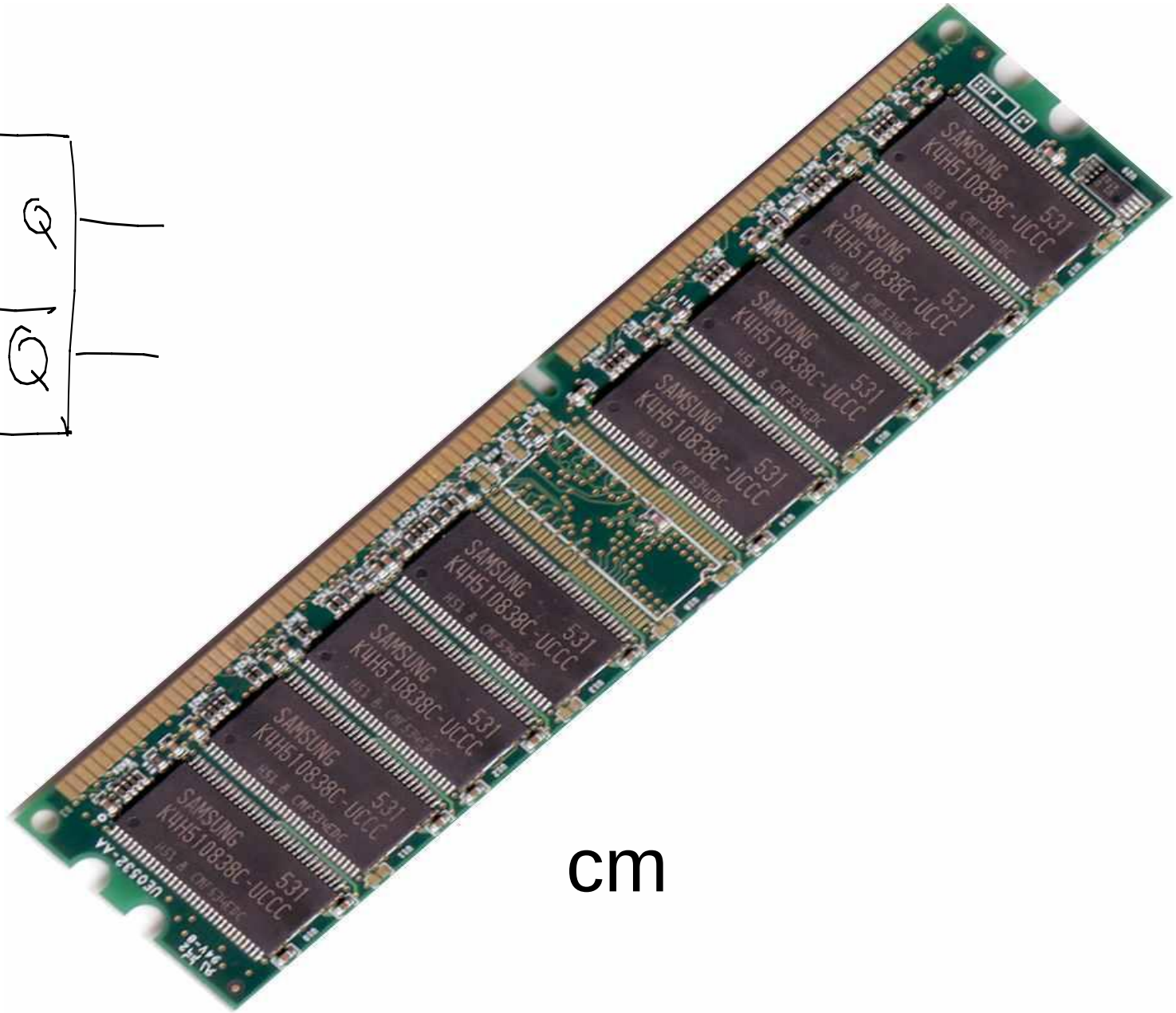
- T flipflops, counters and timers (finishing last lecture)
- register array
- RAM

January 27, 2016

TODAY: from flip flops to RAM

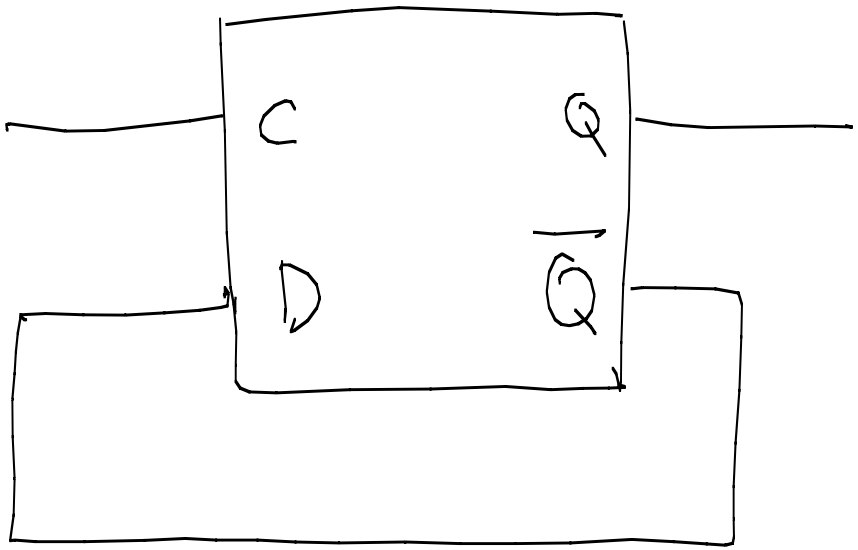


µm

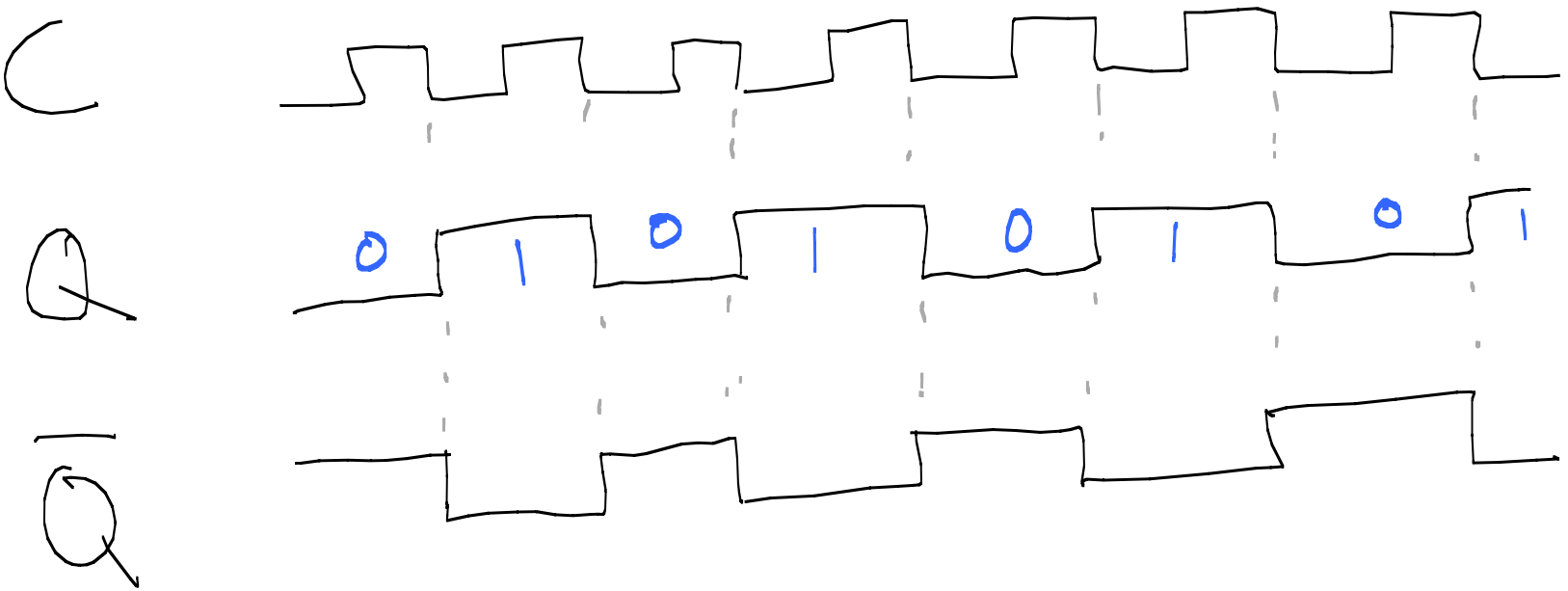


cm

T flip flop (toggle)



Assume falling edge triggered.

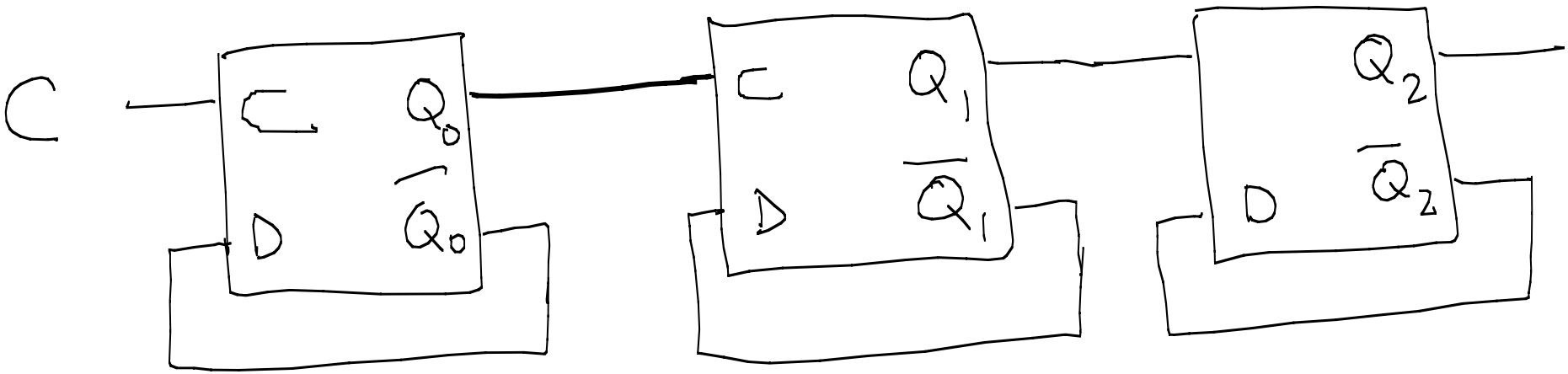


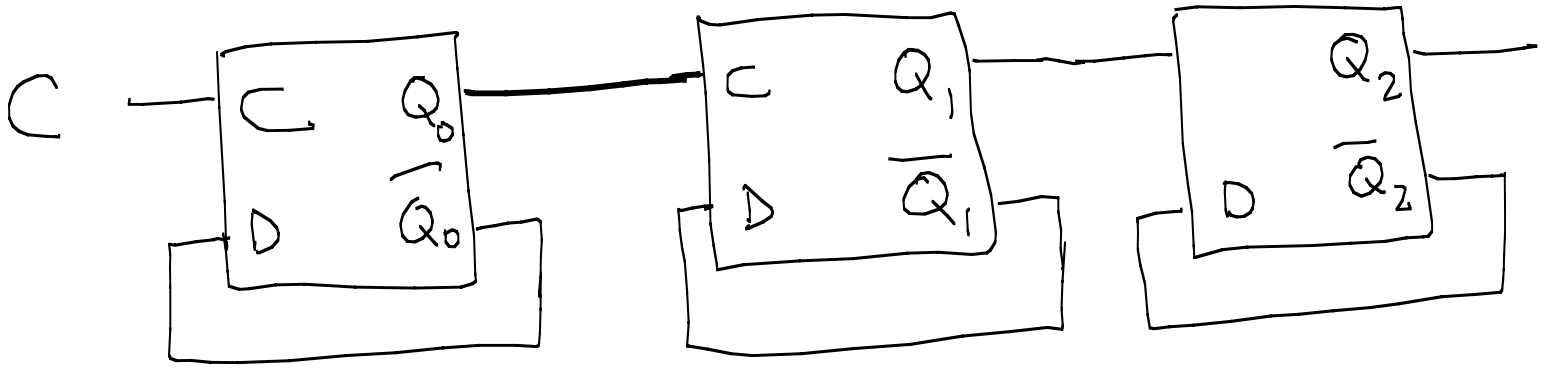
Q: What does this circuit do ?

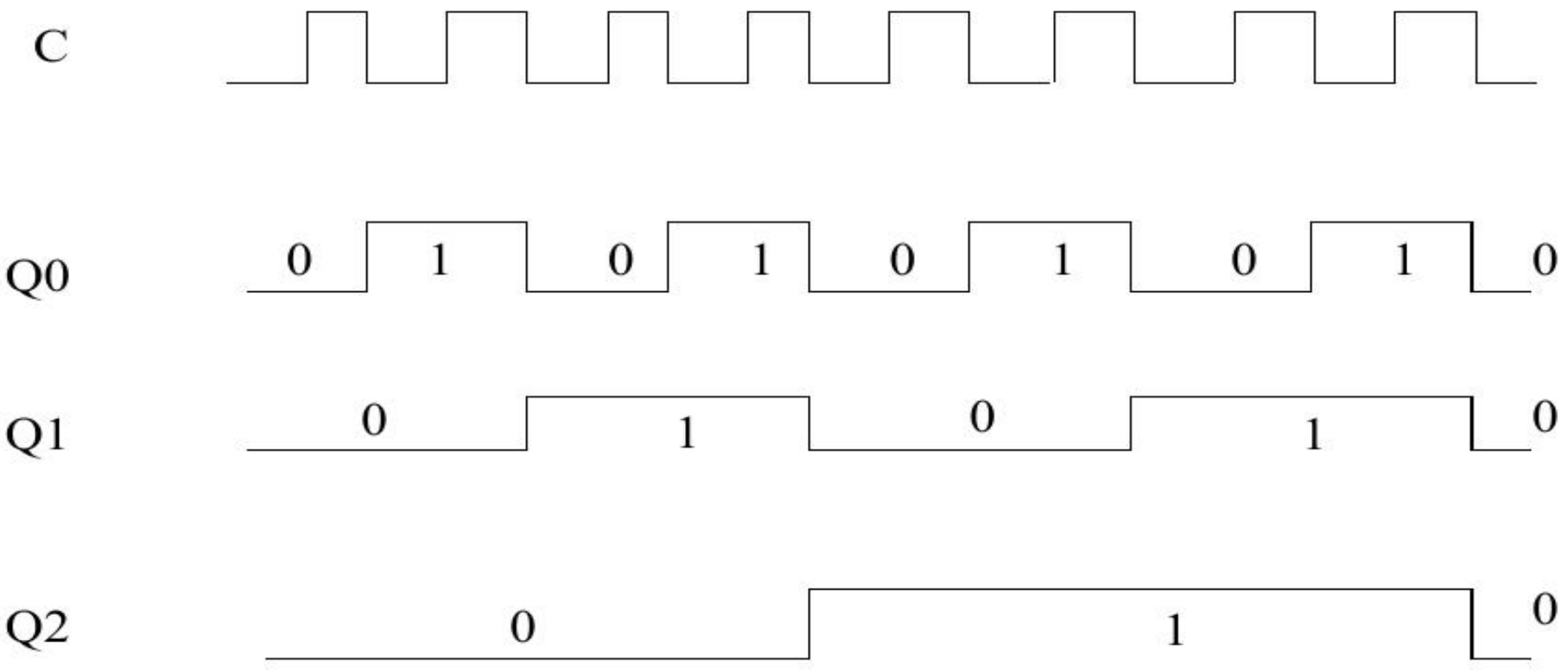
Assume falling edge triggered flip flops.

Qi is the clock input for flip flop i + 1.

i increases from left to right.



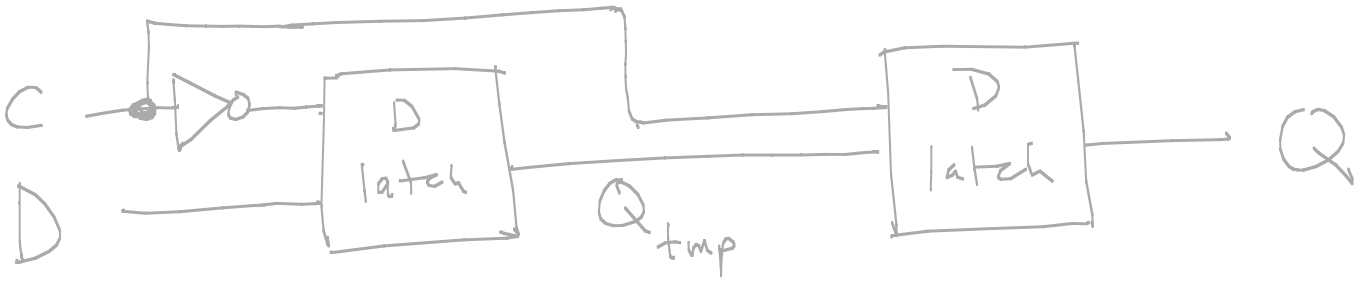




0 1 2 3 4 5 6 7 0

Correction of incorrect claim made from last lecture (see below).

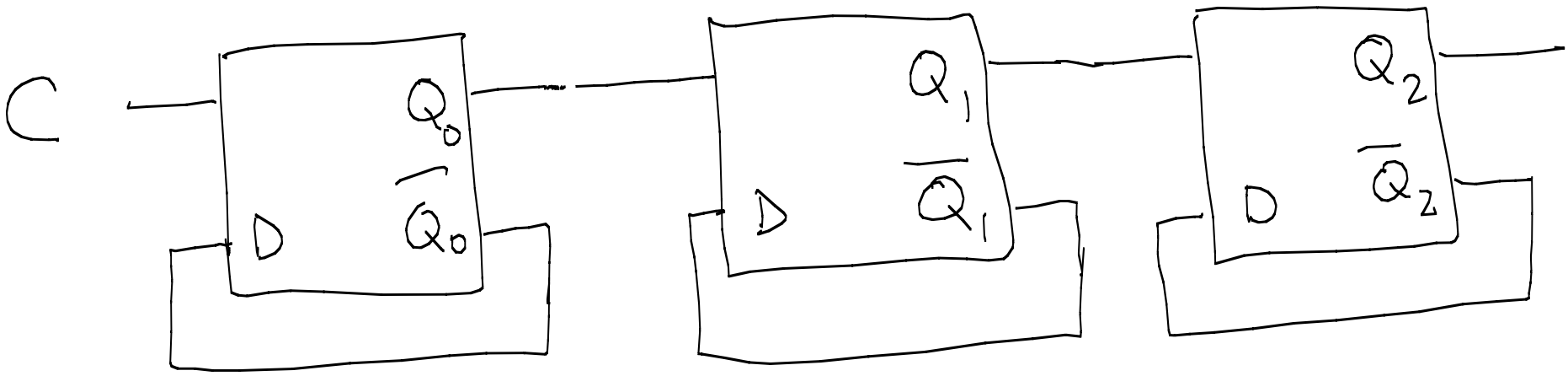
D flip flop ("rising edge triggered")

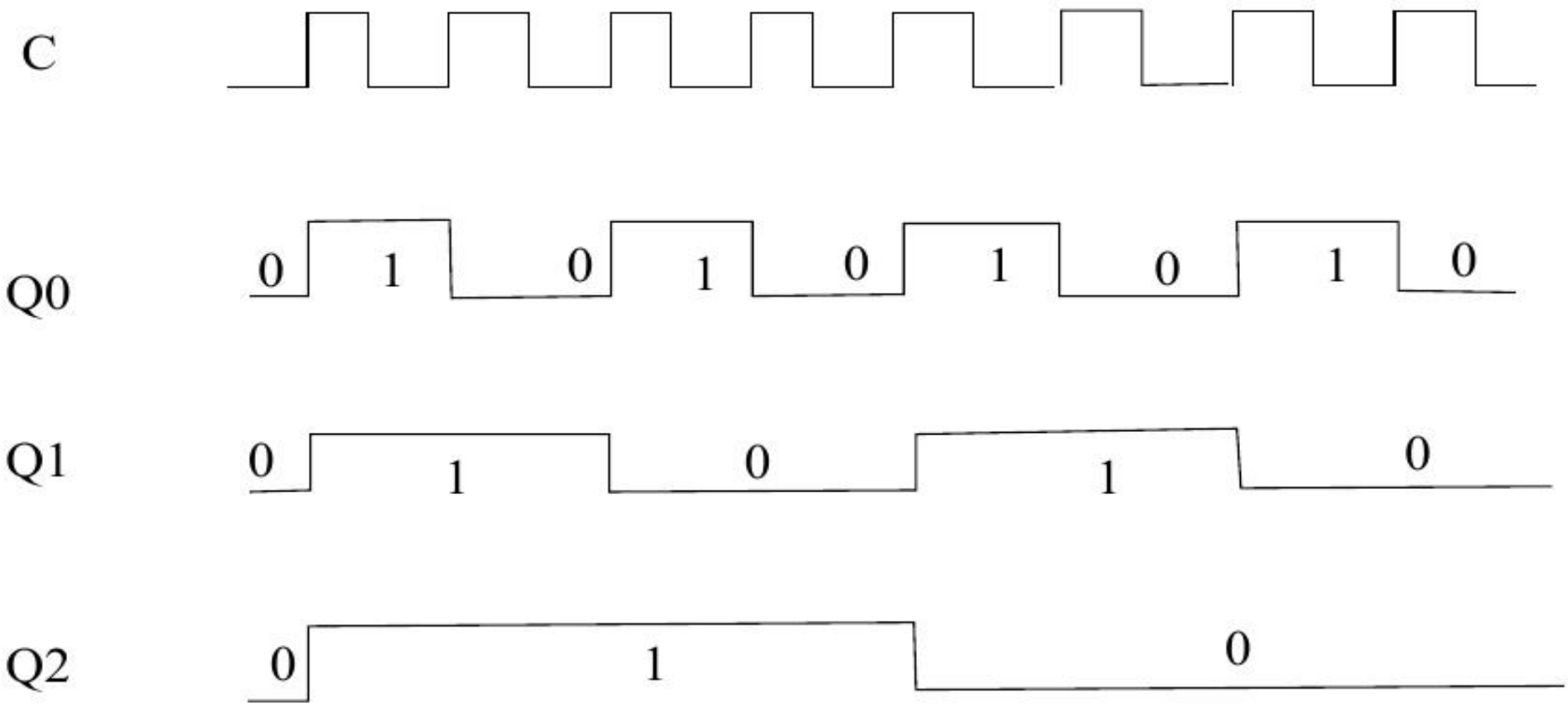


By putting the inverter on the first D latch, we would make Q change its value on the rising edge of the clock. There is no advantage to this, so for simplicity we will always work with falling edge triggered.

Assume **rising** edge triggered.

Q: What does the circuit do ?





A: 0 7 6 5 4 3 2 1 0

Timer (count down)

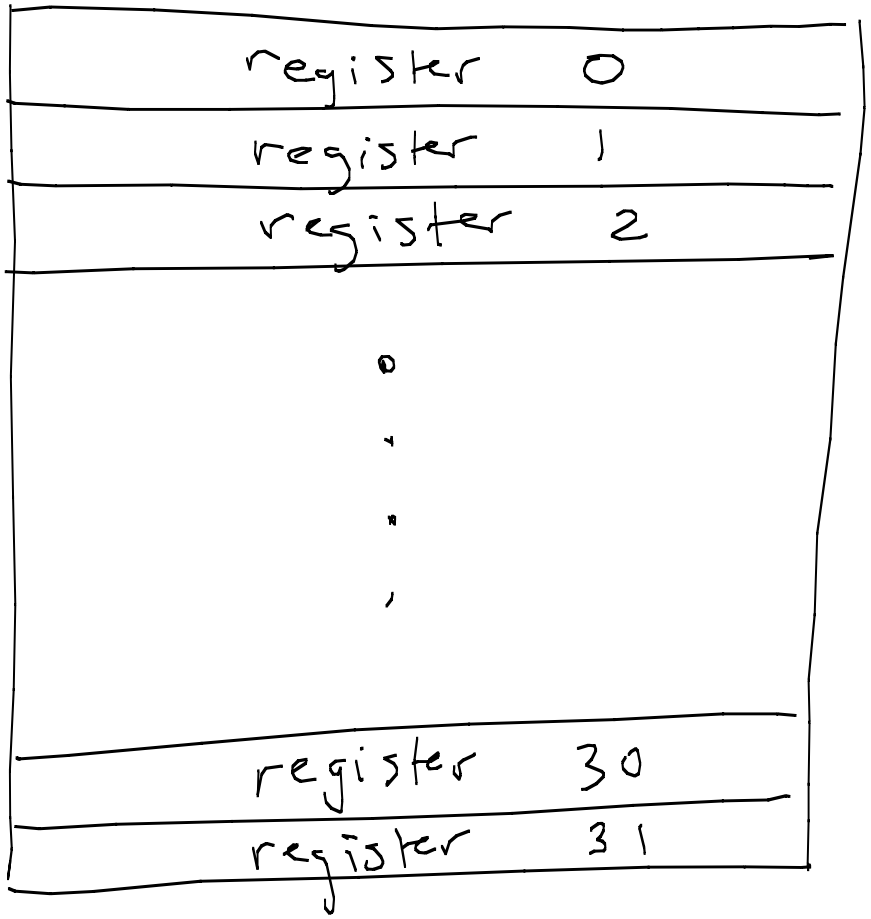
lecture 6

Sequential circuits 2

- T flipflops, counters and timers
- register array [recall what a register is]
- RAM

January 27, 2016

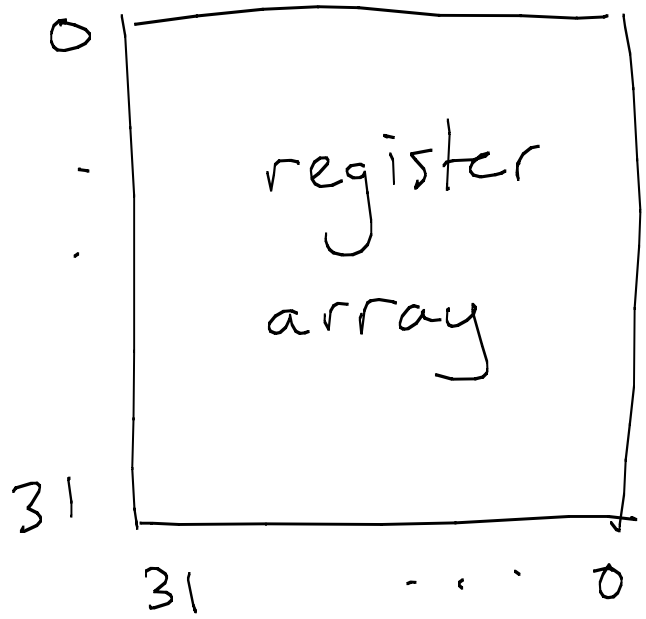
Register Array



In MIPS, there are 32 registers, and each is 32 bits. There is no signficance to the fact that the number of registers is the same as the number of bits per register.

$$x = y + z$$

Suppose the variables x, y, z are stored in registers.



How to read y and z ?

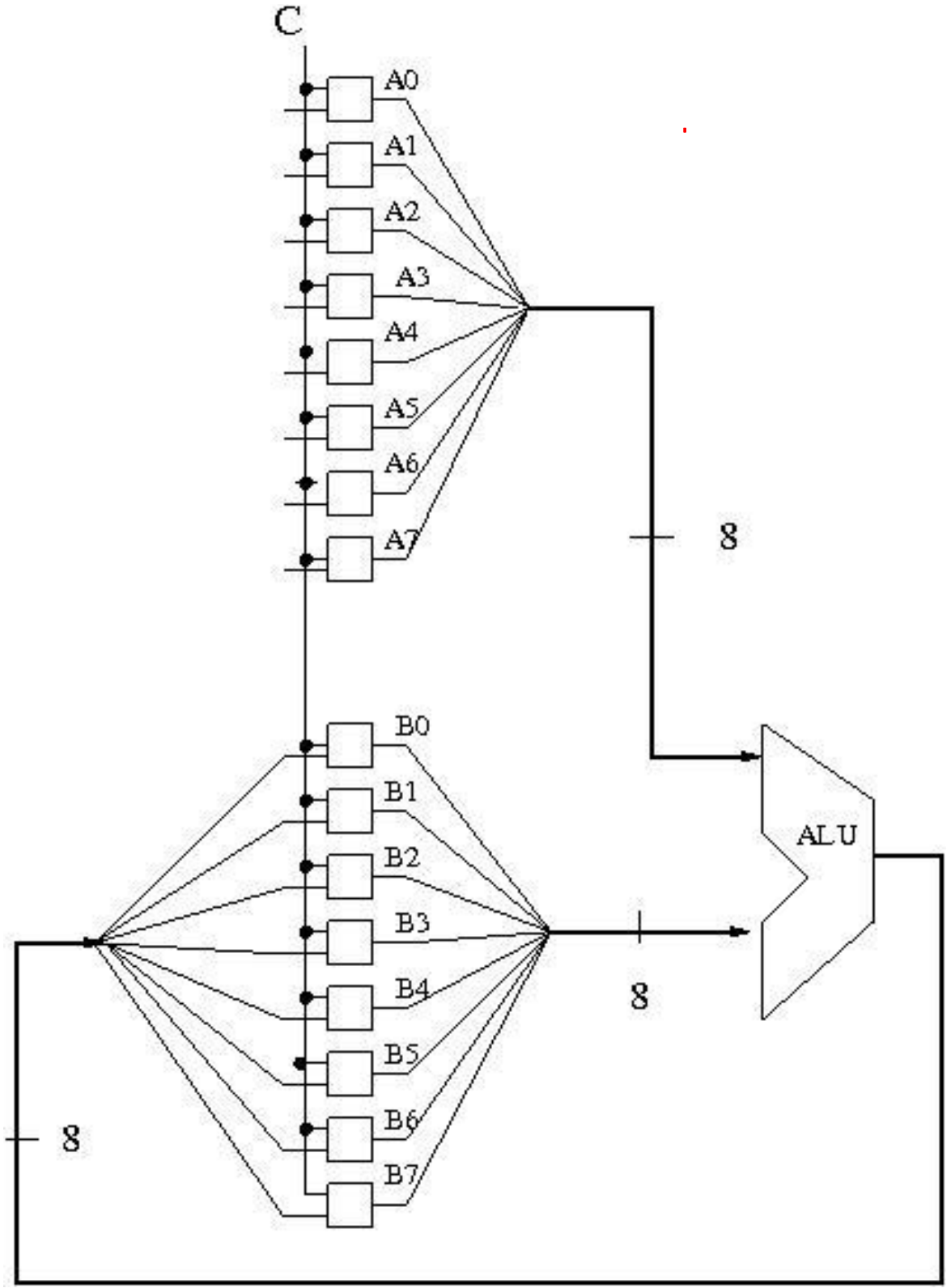
How to write the result into x ?

Recall idea from last lecture:

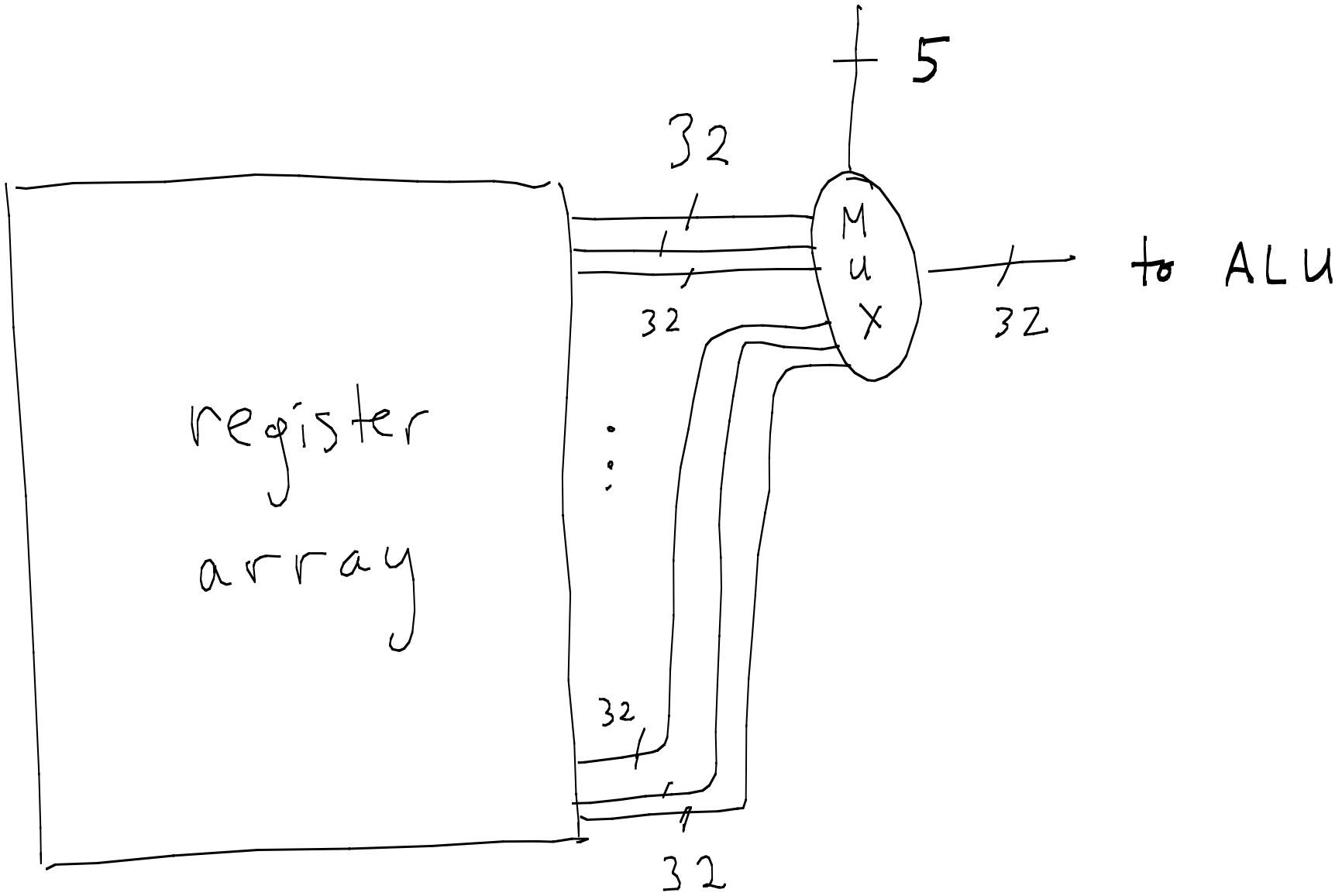
Use sets of D flip flops (register) to store numbers A and B.

Compute A + B using the circuit shown, and write the new value back into B.

For the next slide, we will wil rotate A and B so they are horizontall oriented and have 32 of them (not 2), each 32 bits (not 8).



ReadReg

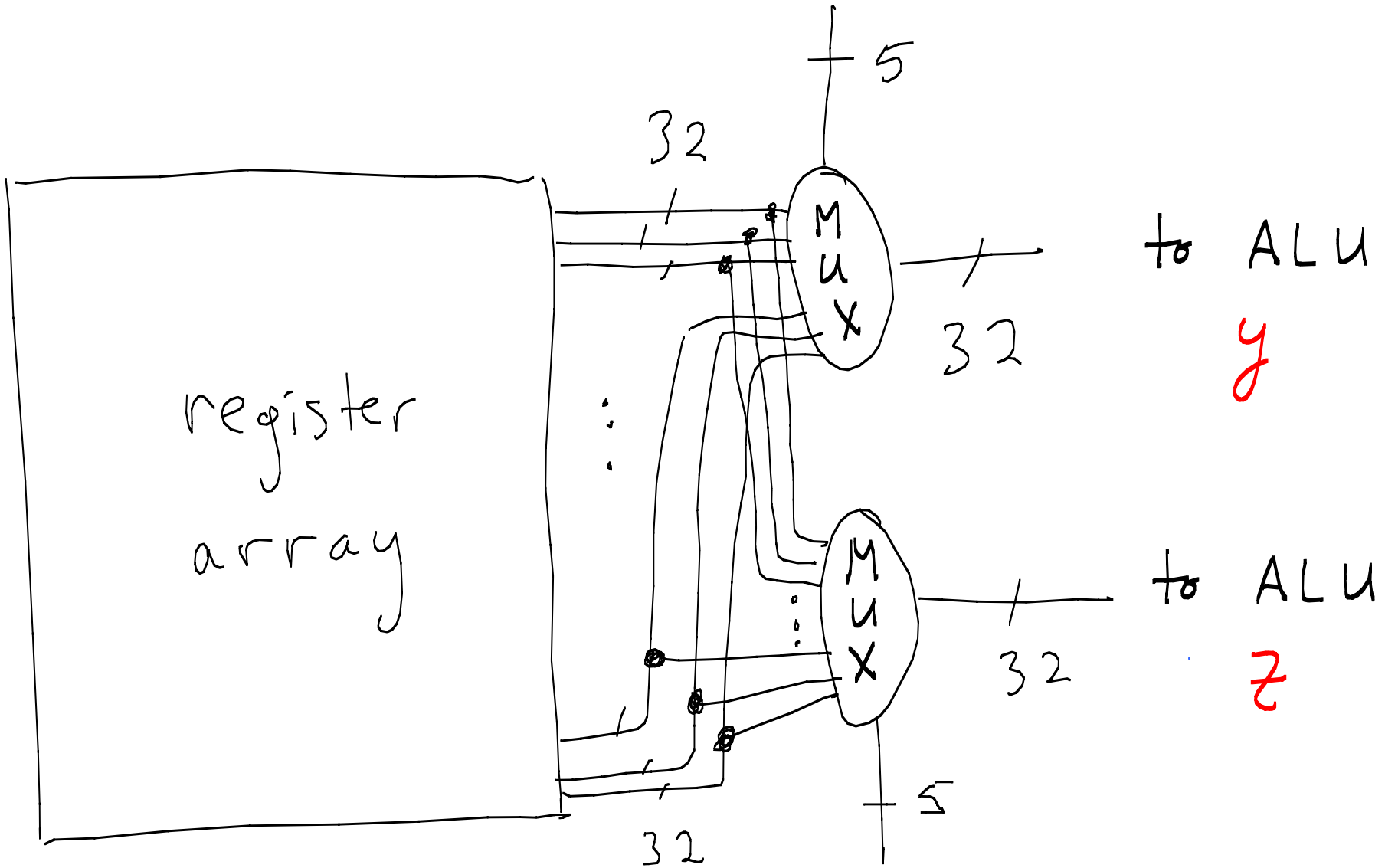


There are 32 x 32 = 1024 data wires fed into the multiplexor. In fact, there are 32 separate 1 bit multiplexors, each using the same 5 bit selector code.

$x = y + z$

ReadReg1

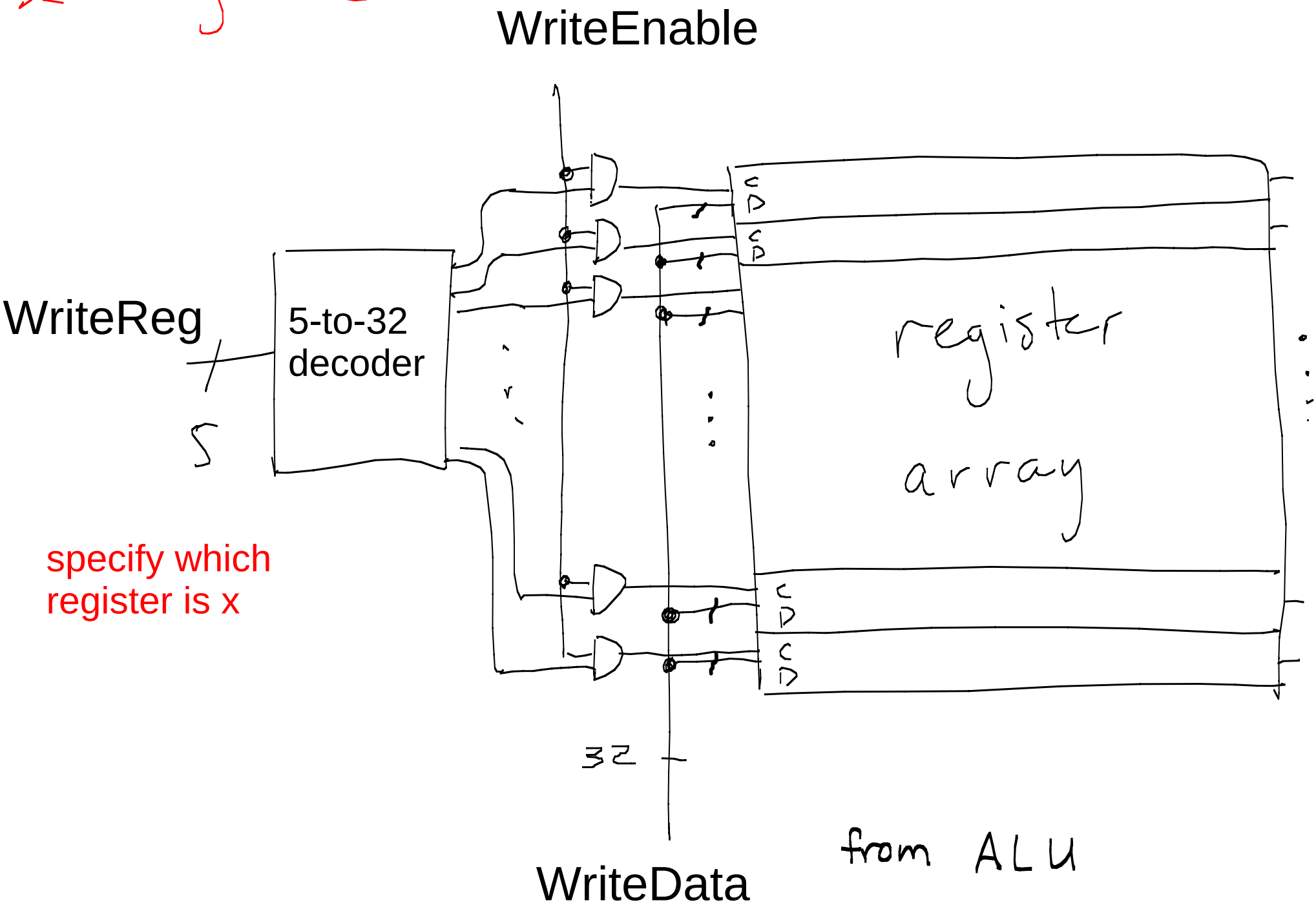
specify which register is y



ReadReg2

specify which register is z

$x = y + z$



specify which
register is x

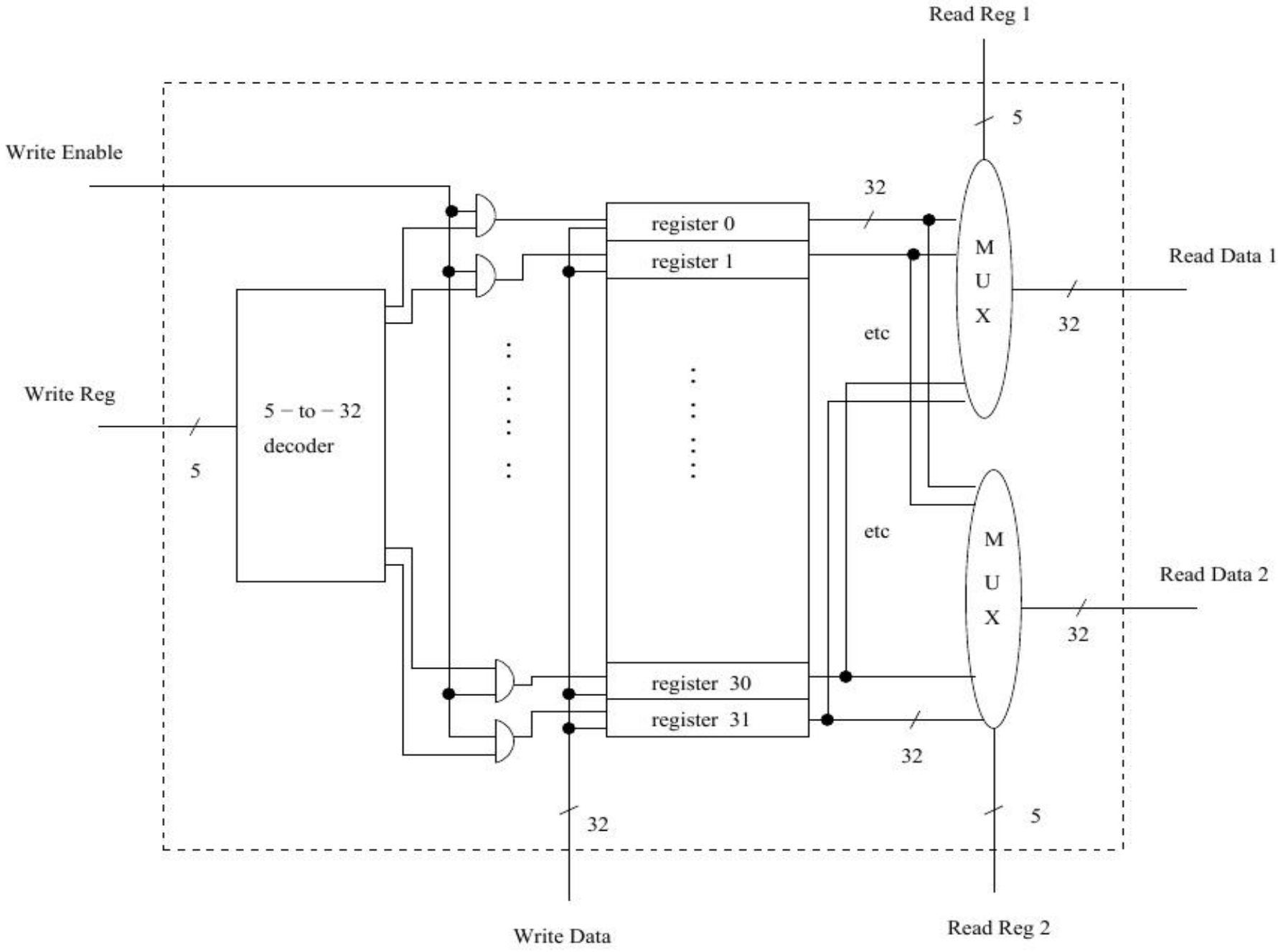
[ADDED SLIDE]

I neglected to mention in the lecture that the clock signal C is embedded in the WriteEnable signal. That is, WriteEnable = 1 if the clock C is 1. Note this is not an "if or only if", rather

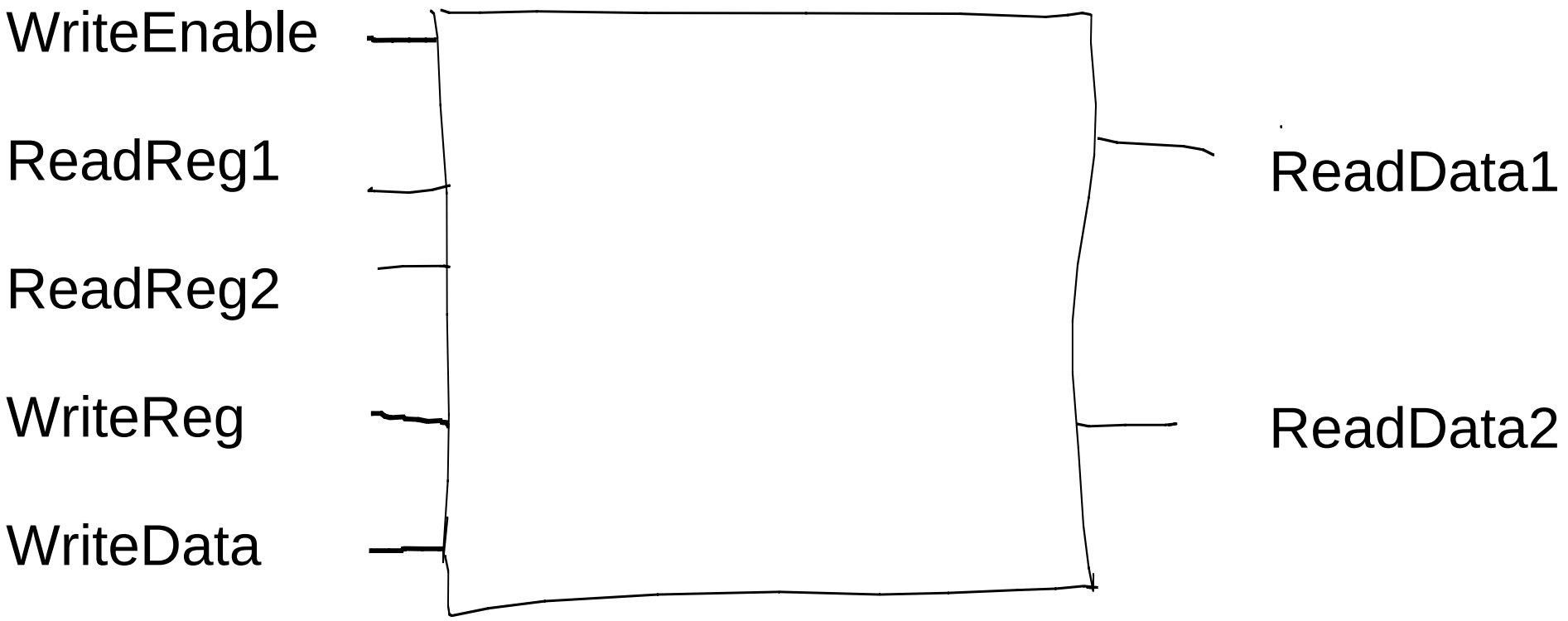
WriteEnable = AND(C, ...)

I won't write the clock explicitly in the future since it is understood that when we are working with registers, we always need a clock to synchronize the writes.

... putting those last two slides together....



Sometimes we write as follows (inputs on left, outputs on right).

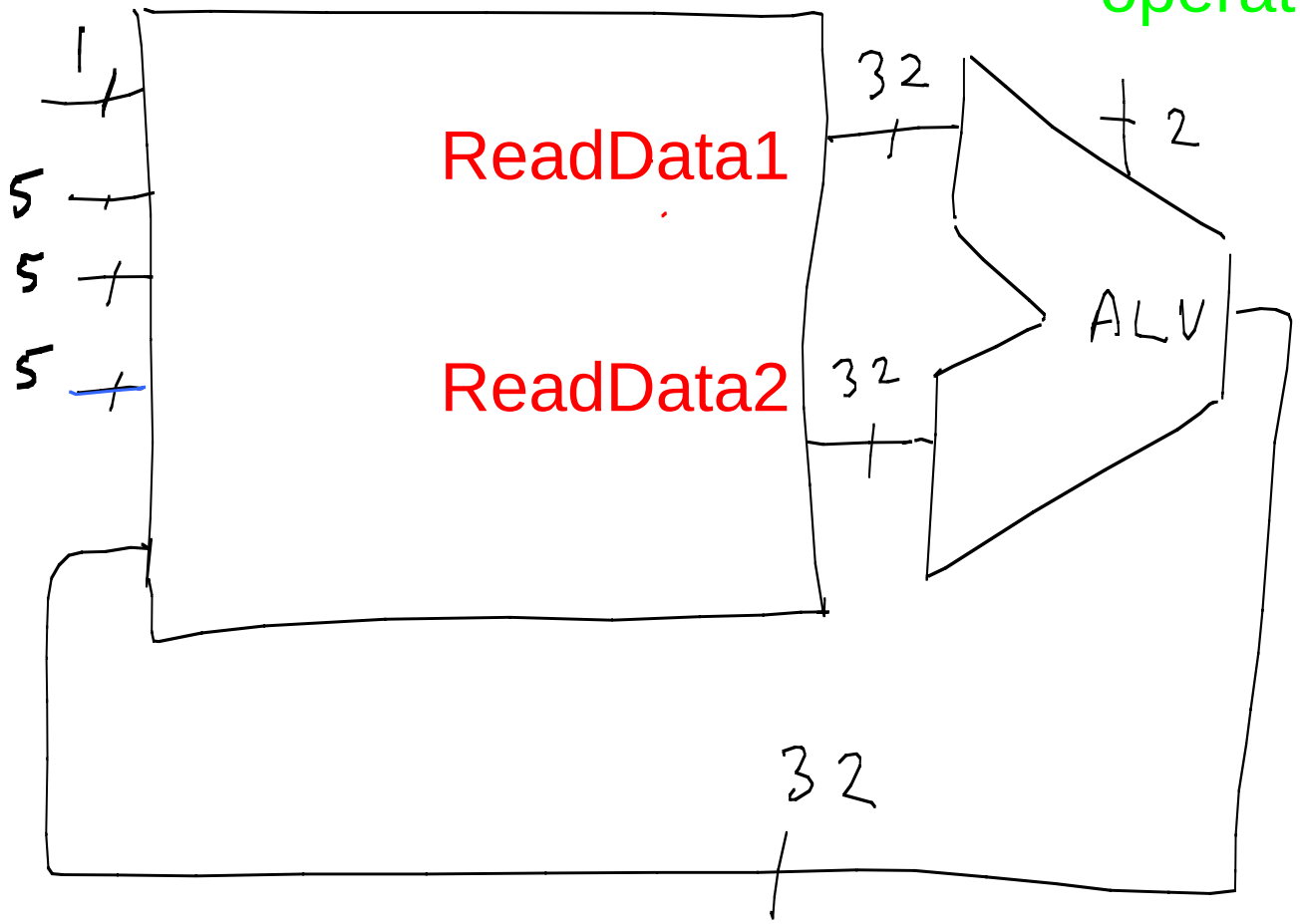


operation

WriteEnable

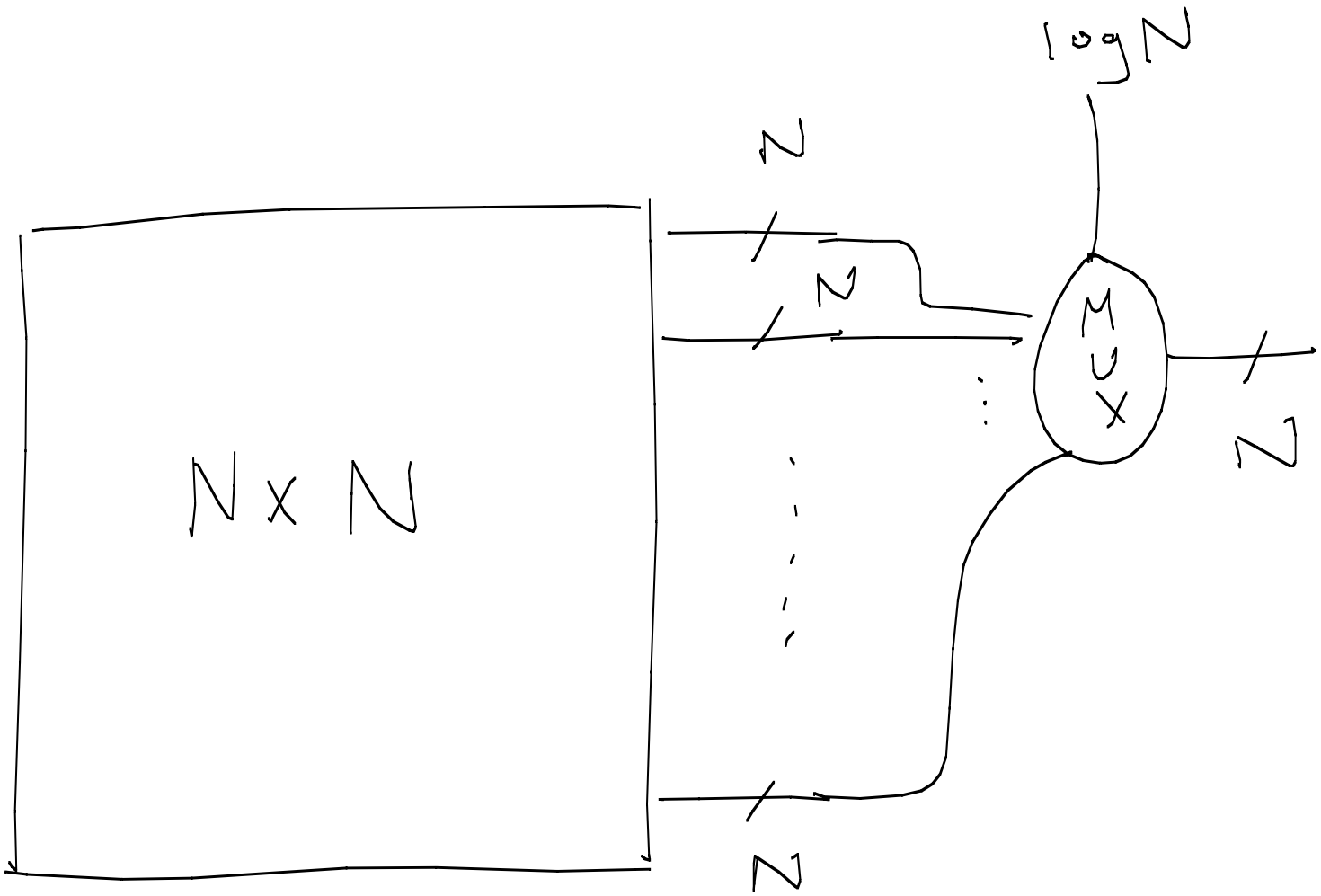
ReadReg1
ReadReg2
WriteReg

WriteData



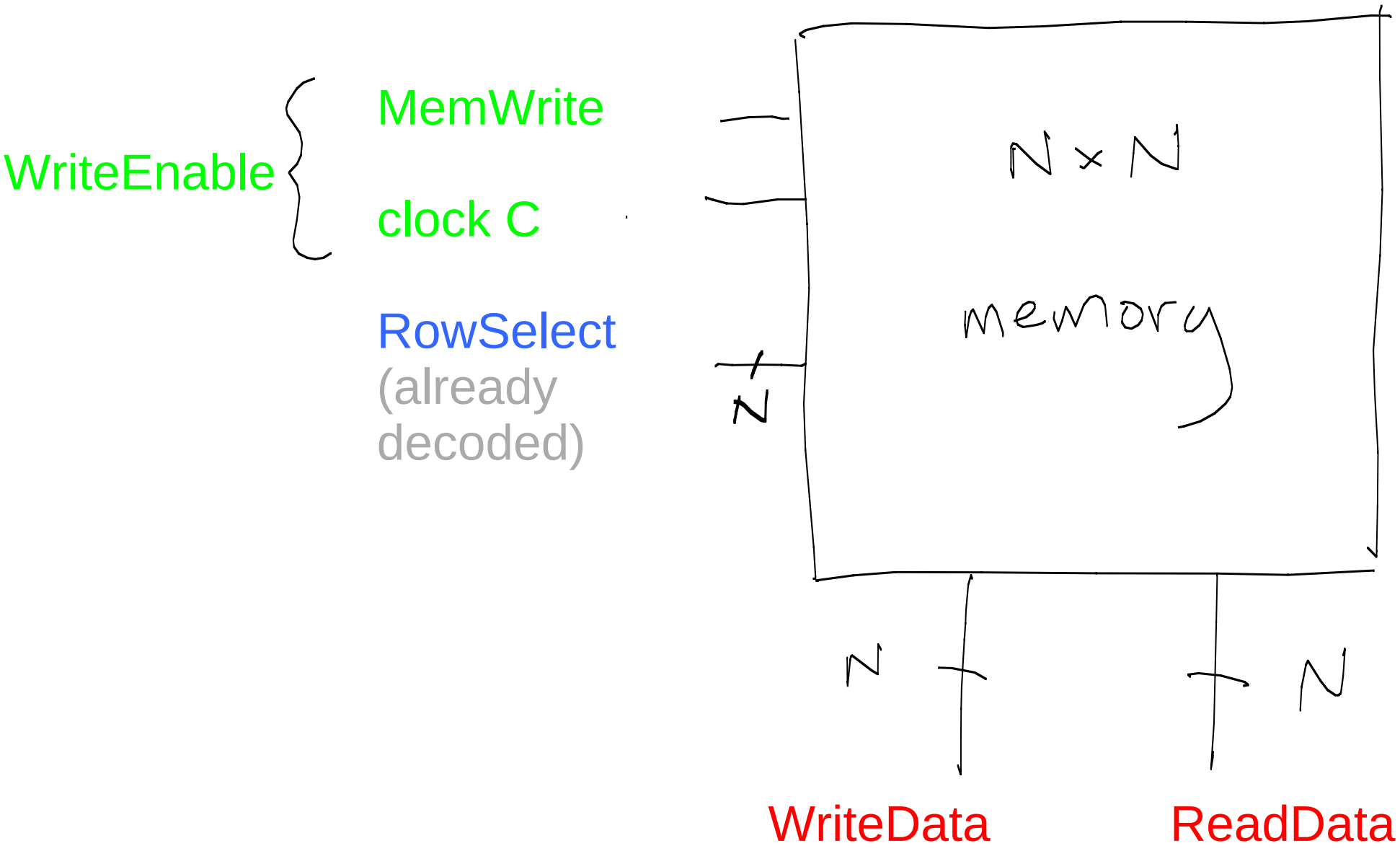
address
data
control

types of signals



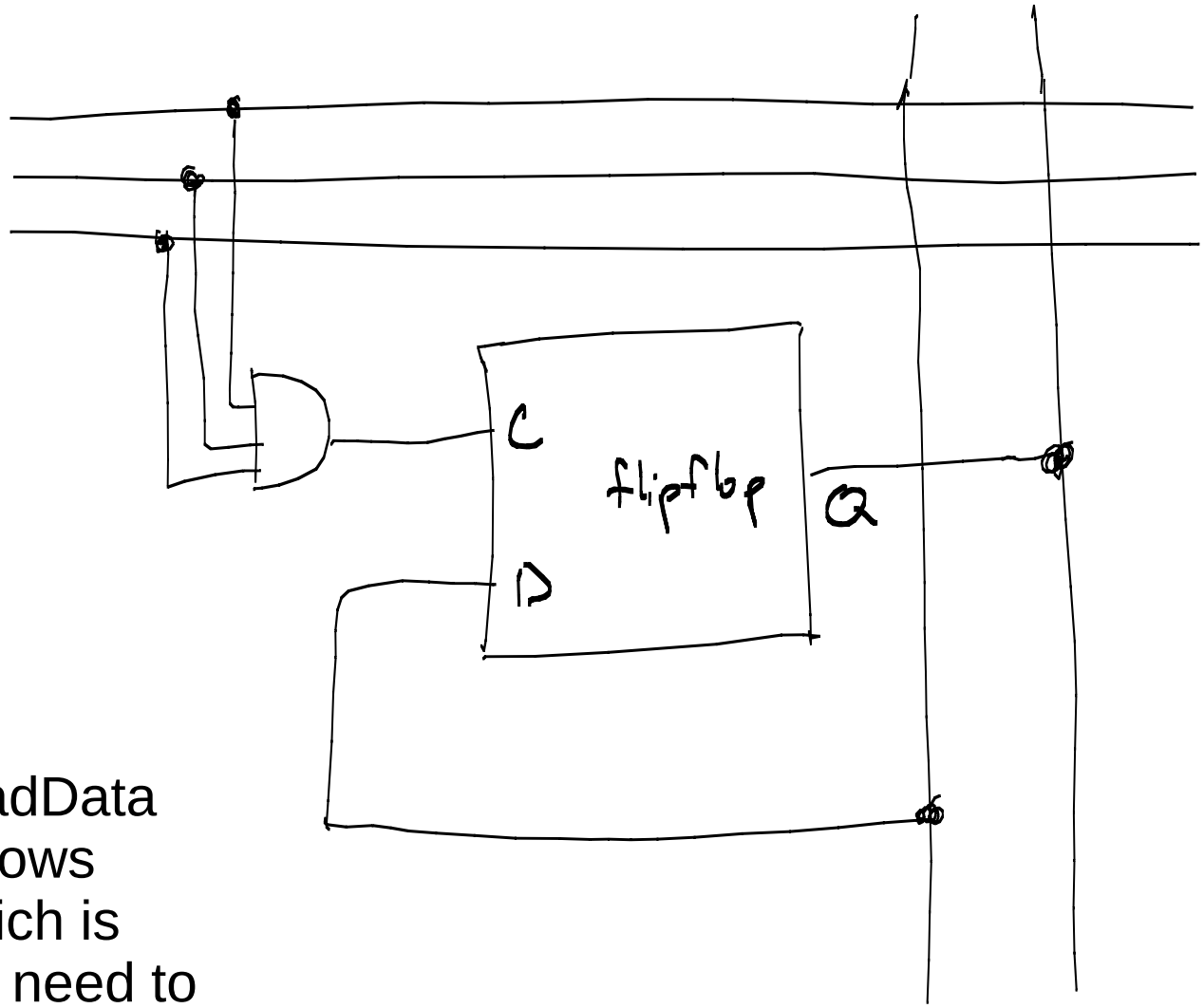
For larger memory arrays, the multiplexor design is not physically feasible. You need N^2 wires coming out. But the side of the square array only grows with N.

An alternative approach ? .... somehow have only 2N wires, namely a read and a write wire for each column.



Consider what happens for each of the N^2 flip flops.
All flip flops in a row (column) share the same horizontal (vertical) wire.

MemWrite
clock C
RowSelect

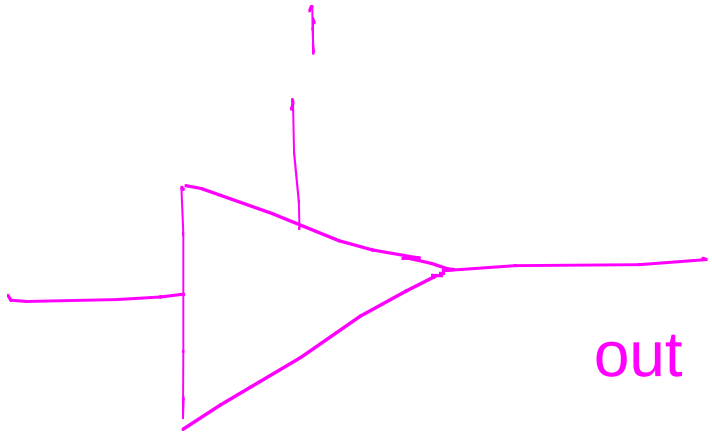


There is a problem, however. The ReadData line reads from all rows simultaneously, which is not allowed. We need to select one. How ?

WriteData ReadData

Tri-state Gate (not a logic gate)

- also known as a 'tri-state buffer'
- output can have values 0, 1, or none
  (voltages are low, high, or zero)

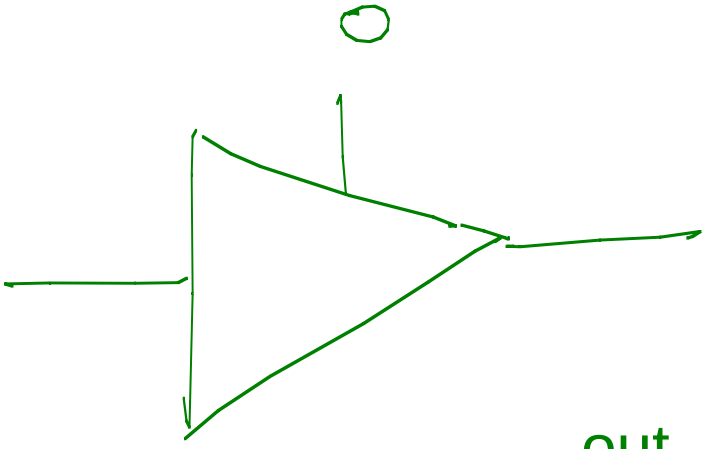


in

out

means

connect



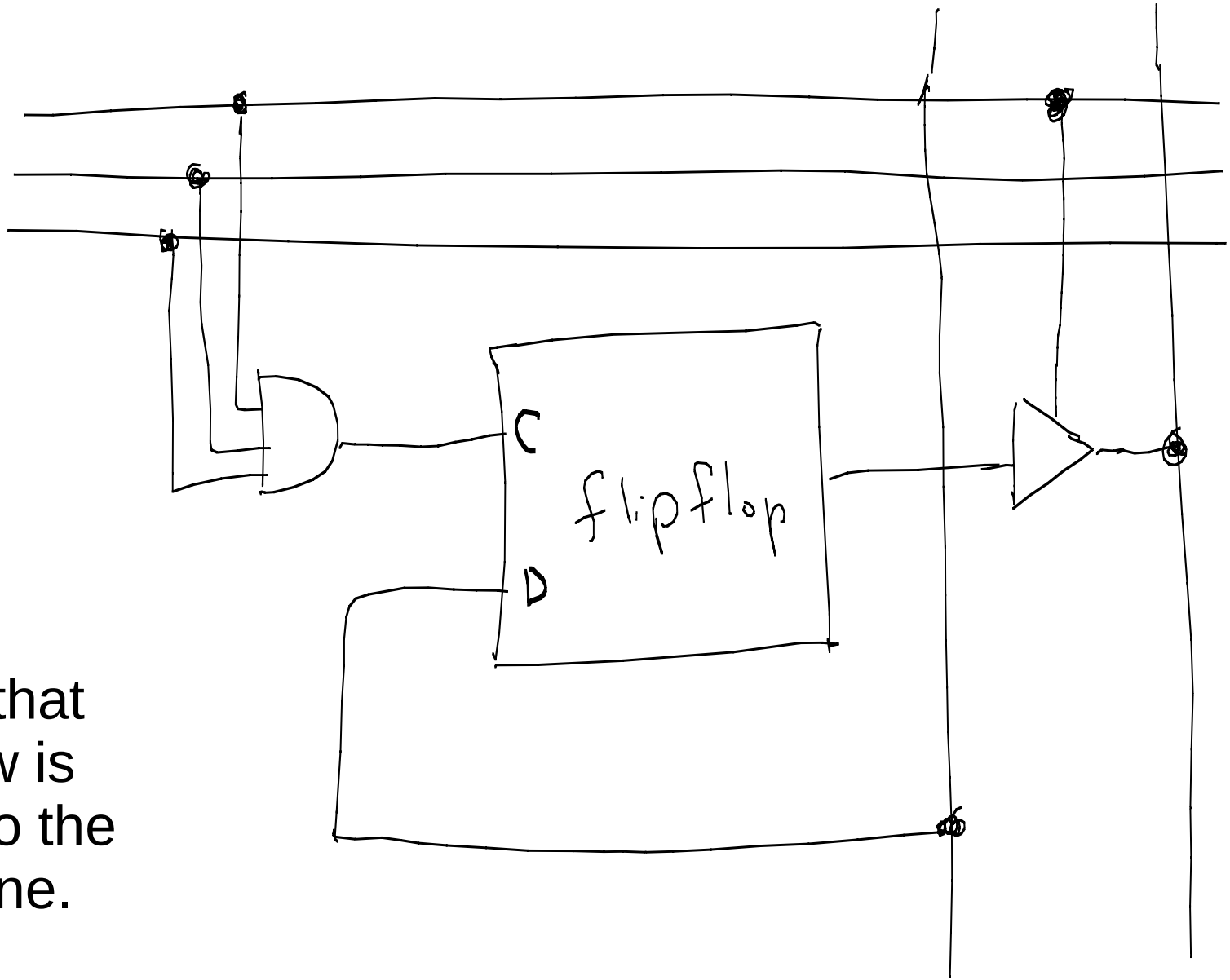
in

out

means

disconnect

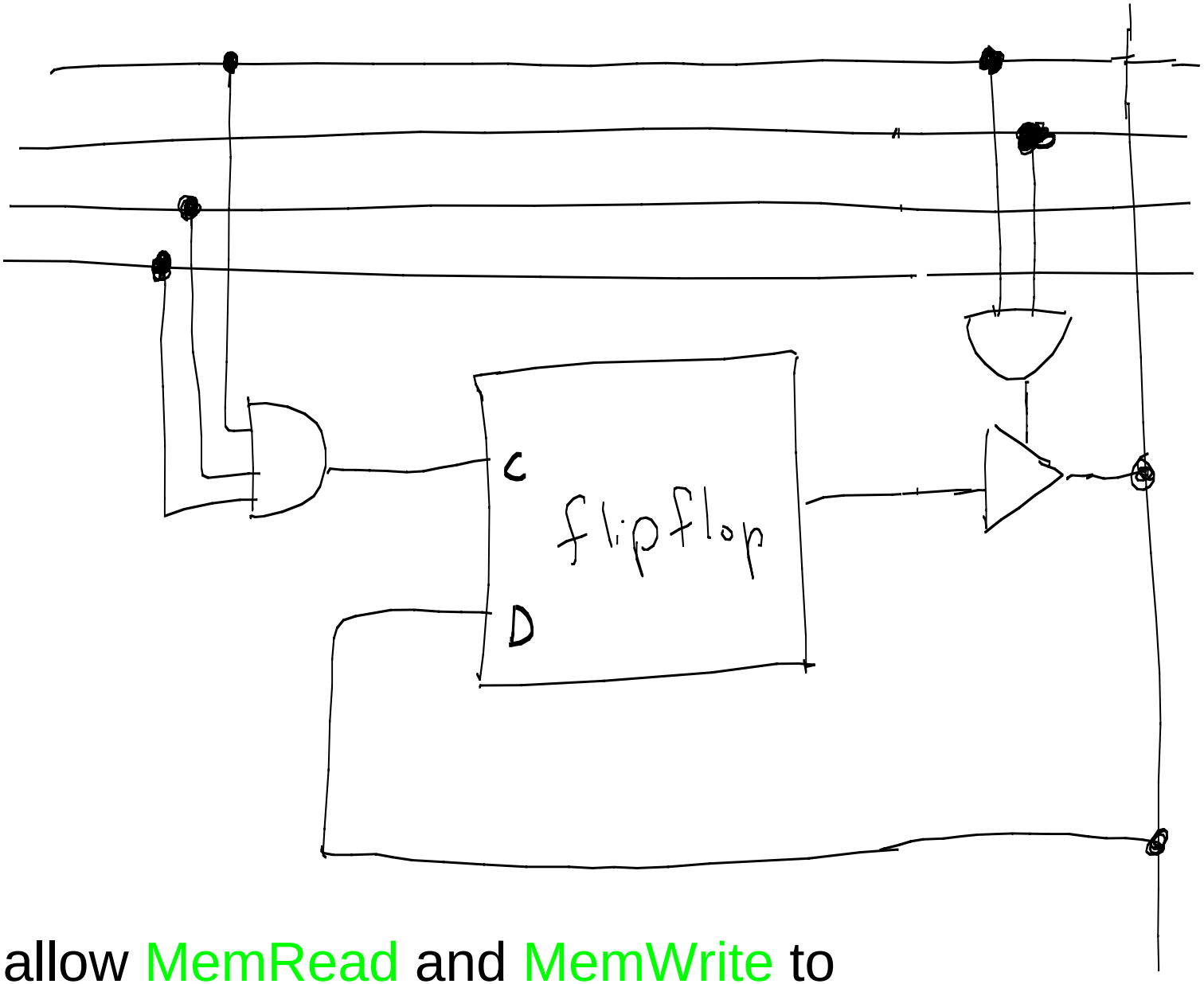
RowSelect
MemWrite
clock C



The idea is that only one row is connected to the ReadData line.

WriteData ReadData

RowSelect
MemRead
MemWrite
clock C



We would not allow MemRead and MemWrite to both be 1 (not shown in circuit). Also, sometimes neither would be 1.

Data

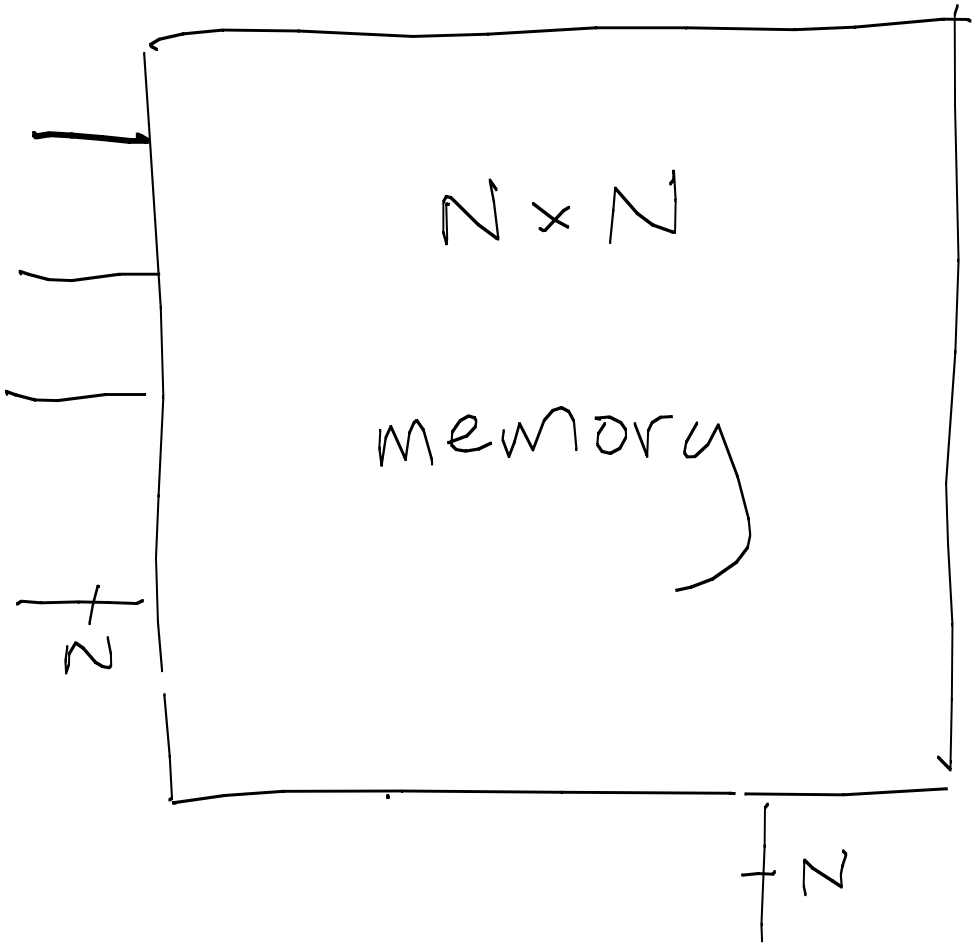
We have been thinking of reading or writing an entire row.

MemWrite

MemRead

clock C

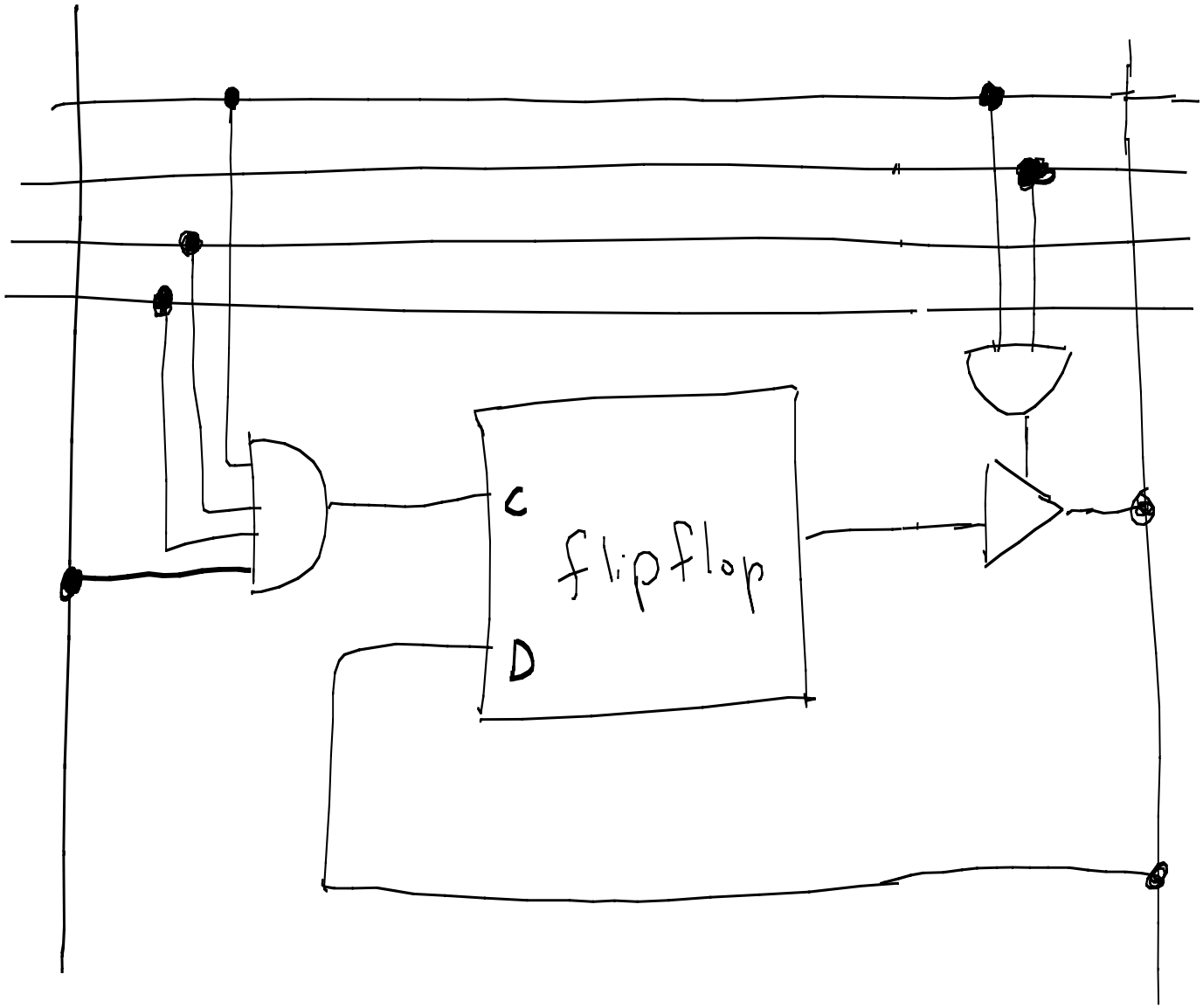
RowSelect
(already decoded)



Data

Let's next select only a single row and column.

RowSelect
MemRead
MemWrite
clock C



ColumnSelect

Data

1 bytes = 8 bits

1 KB = 2^10 bytes ("kilo")

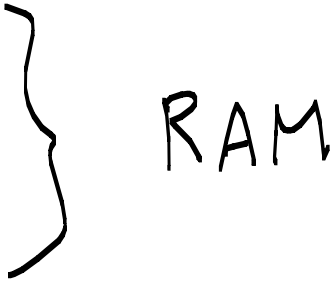
1 MB = 2^20 bytes ("mega")

1 GB = 2^30 bytes ("giga")

1 TB = 2^40 bytes ("tera")

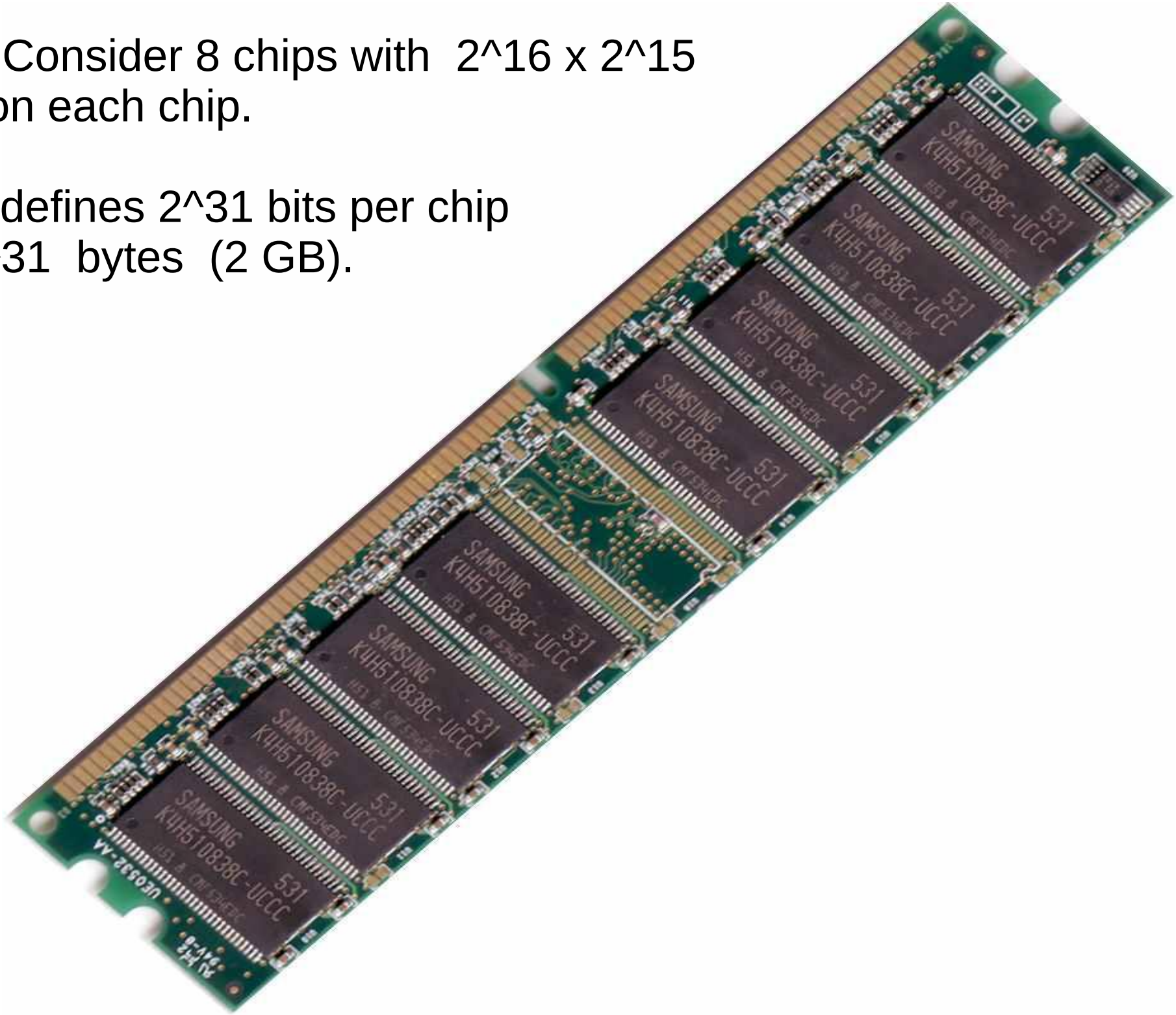
1 PB = 2^50 bytes ("peta")

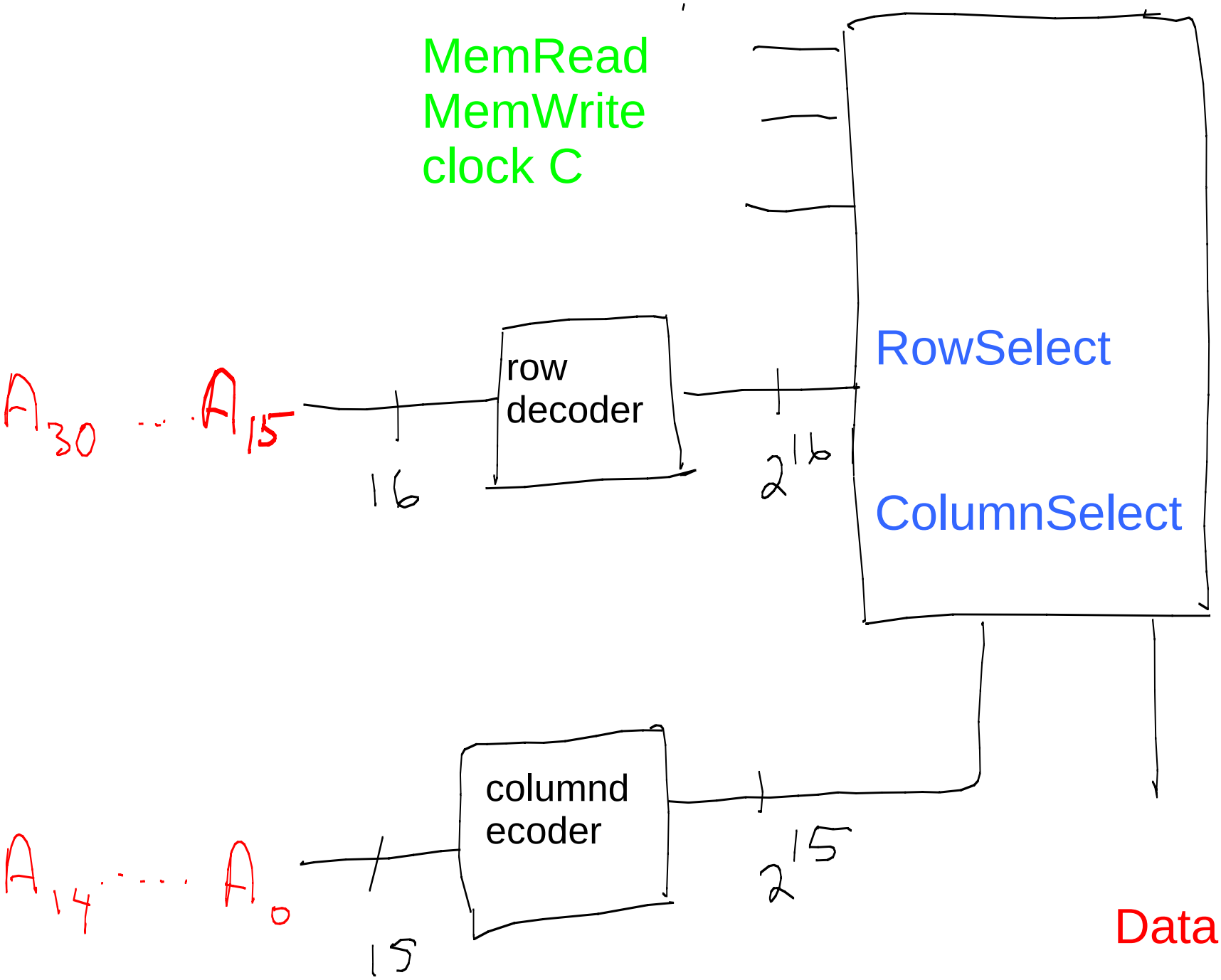
1 EB = 2^60 bytes ("exa")



e.g. Consider 8 chips with 2^16 x 2^15 bits on each chip.

This defines 2^31 bits per chip or 2^31 bytes (2 GB).




MemRead
MemWrite
clock C
RowSelect
ColumnSelect
row
decoder
columnd
ecoder
$A_{30} \cdots A_{15}$
$A_{14} \cdots A_0$
16
15
2^{16}
2^{15}
Data

Announcements

- Quiz 1 and yellow stickies

- Quizzes: who writes in Arts 145 ? (70 seats)

  Quiz 2 A-H (lastname starts with...)

  Quiz 3 I - P

  Quiz 4 I - P

  Quiz 5 R-Z

  Quiz 6 R-Z

- Assignment 1 posted ~next Monday,
  download 'logisim'
  (there will be a demo on class on Monday)