Exam 1 is live

Registers and Register Files

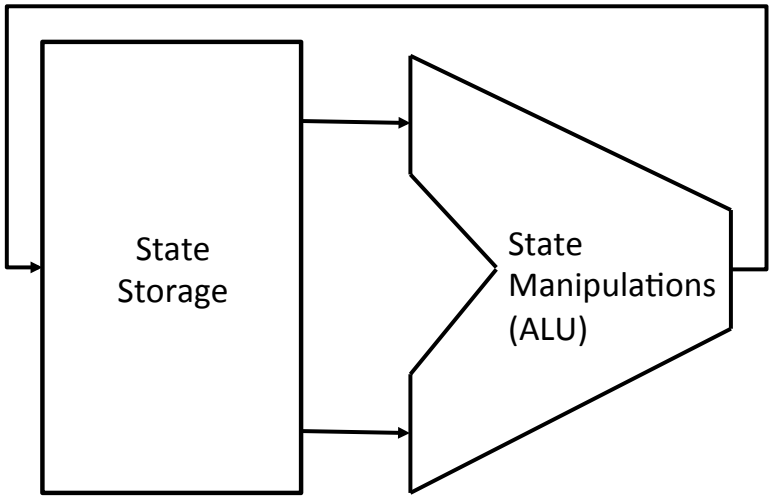
Pick up HANDOUT.

State – the central concept of computing

Computer can do 2 things

1)Store state (How do we actually store bits?)

2)Manipulate state

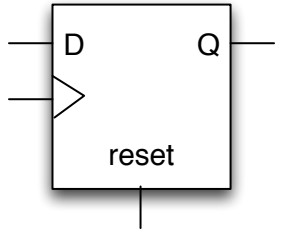


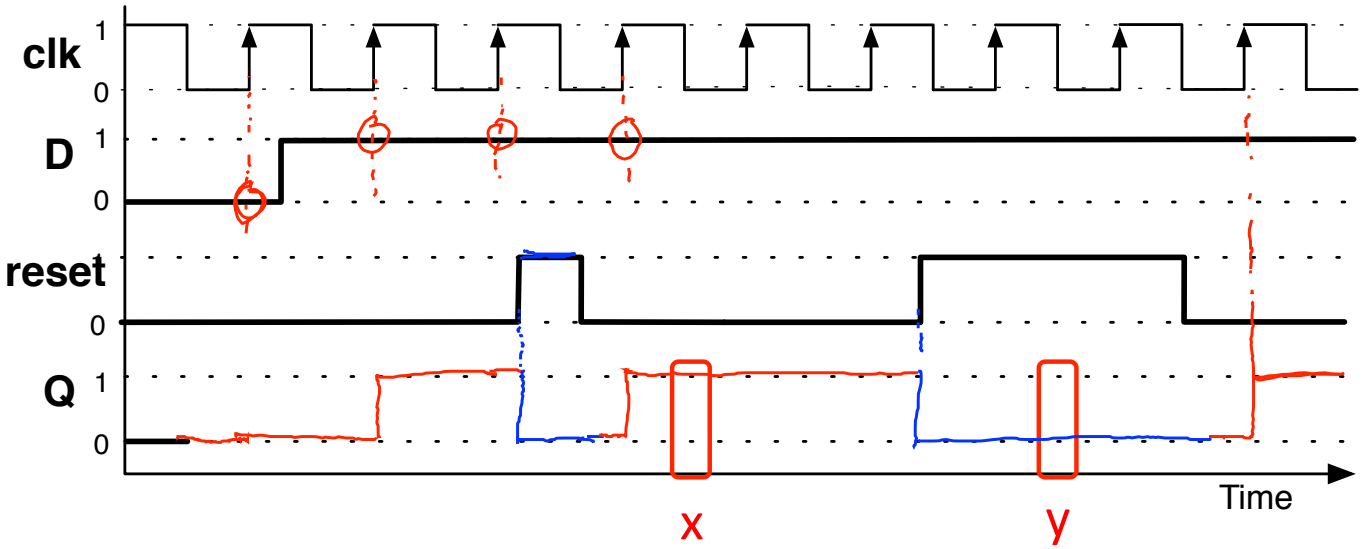
Today's lecture

- More D Flip flops
  - Asynchronous reset
  - Enable
- Random Access Memory (RAM)
  - Addressable storage
- Register Files
  - Registers
  - Decoders

Asynchronous reset immediately resets a flip-flop to 0

- Asynchronous = pertaining to operation without the use of fixed time intervals (opposed to synchronous).





x,y =

a) 0,0

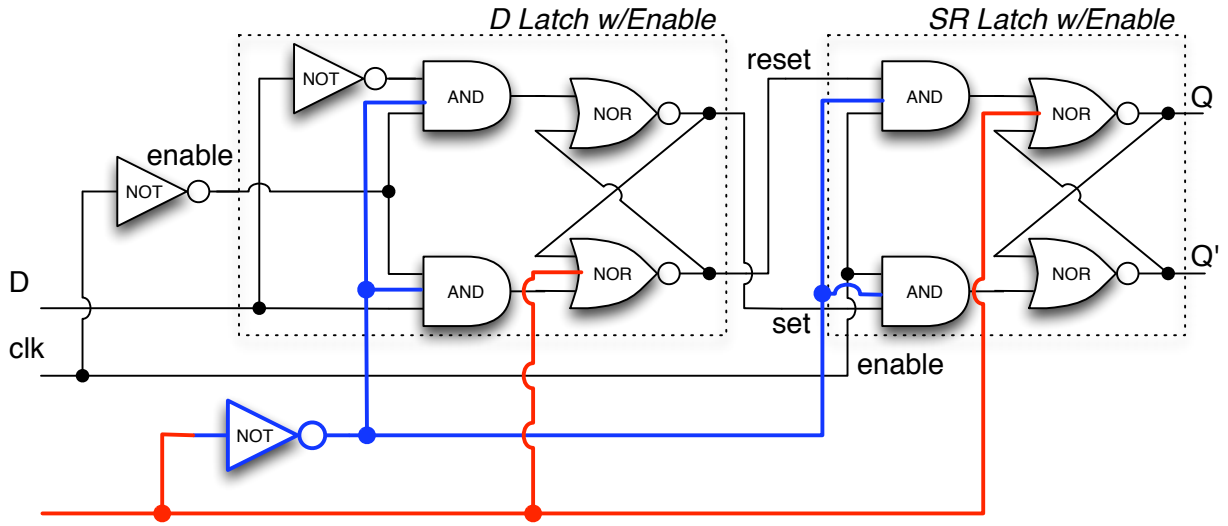
b) 0,1

c) 1,0

d) 1,1

Asynchronous Reset implementation

One example possible implementation



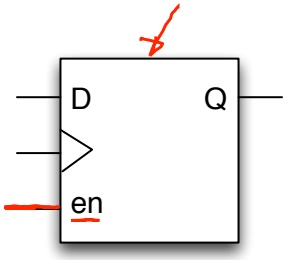
Ignores inputs and current state.

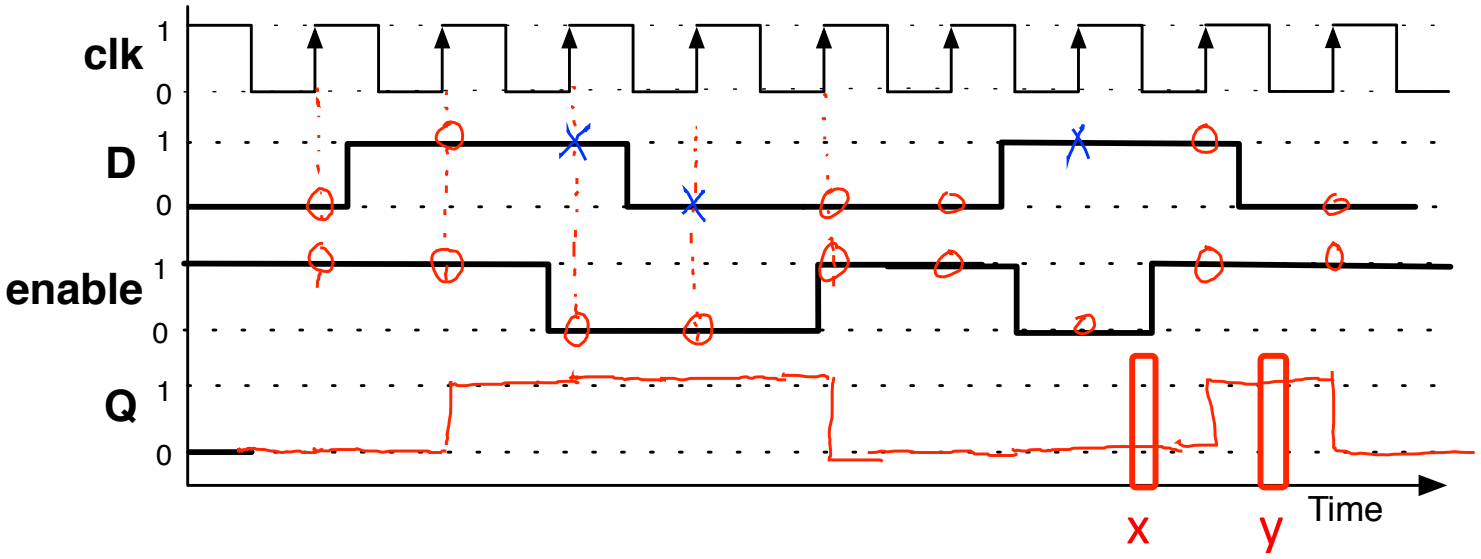
Forces Q output to zero.

(Not required material)

When enable is 0, the flip flop doesn't change on the rising edge

- Behaves normally when enable=1



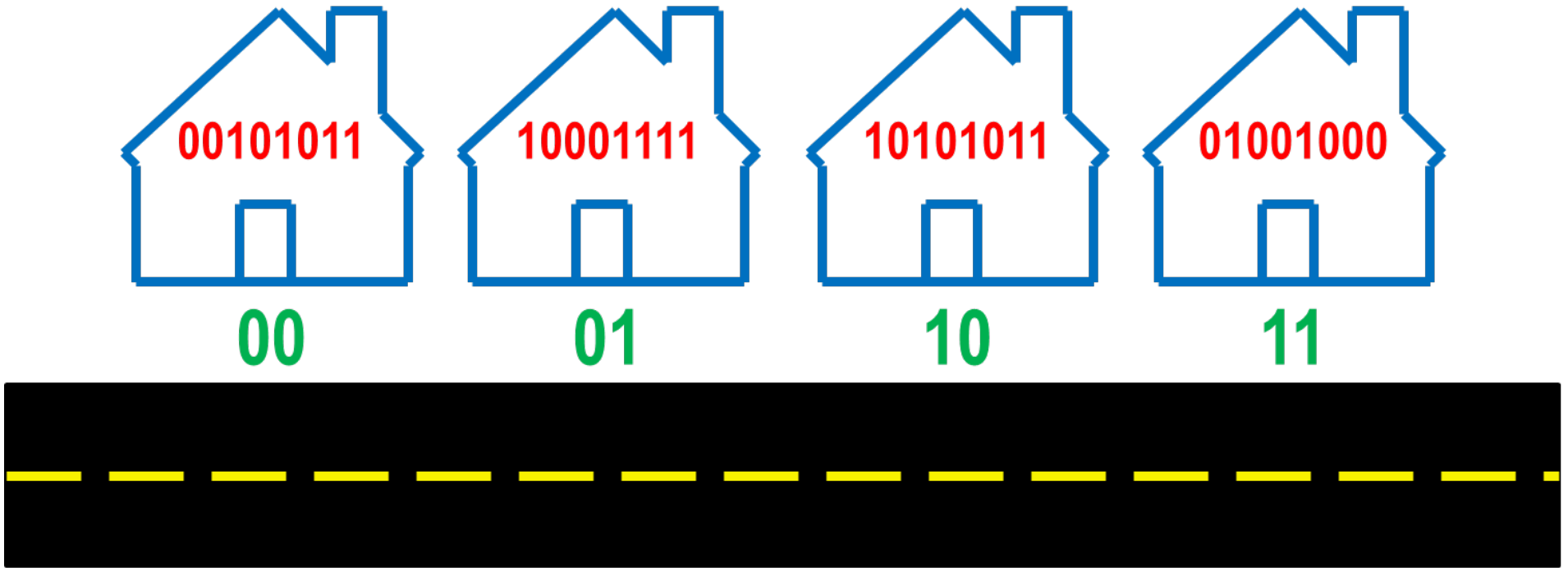


x,y =
a) 0,0
b) 0,1
c) 1,0
d) 1,1

$2^2 = 4$

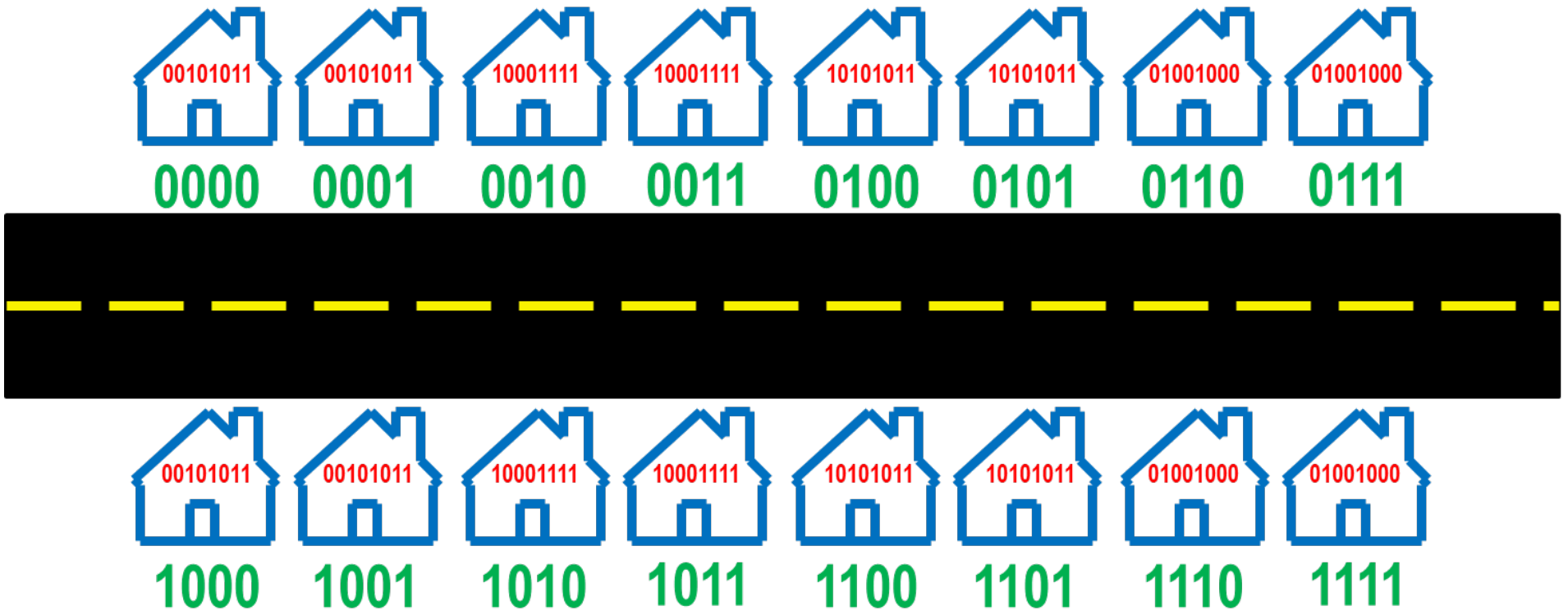
2 × 8

Address bits × Data bits



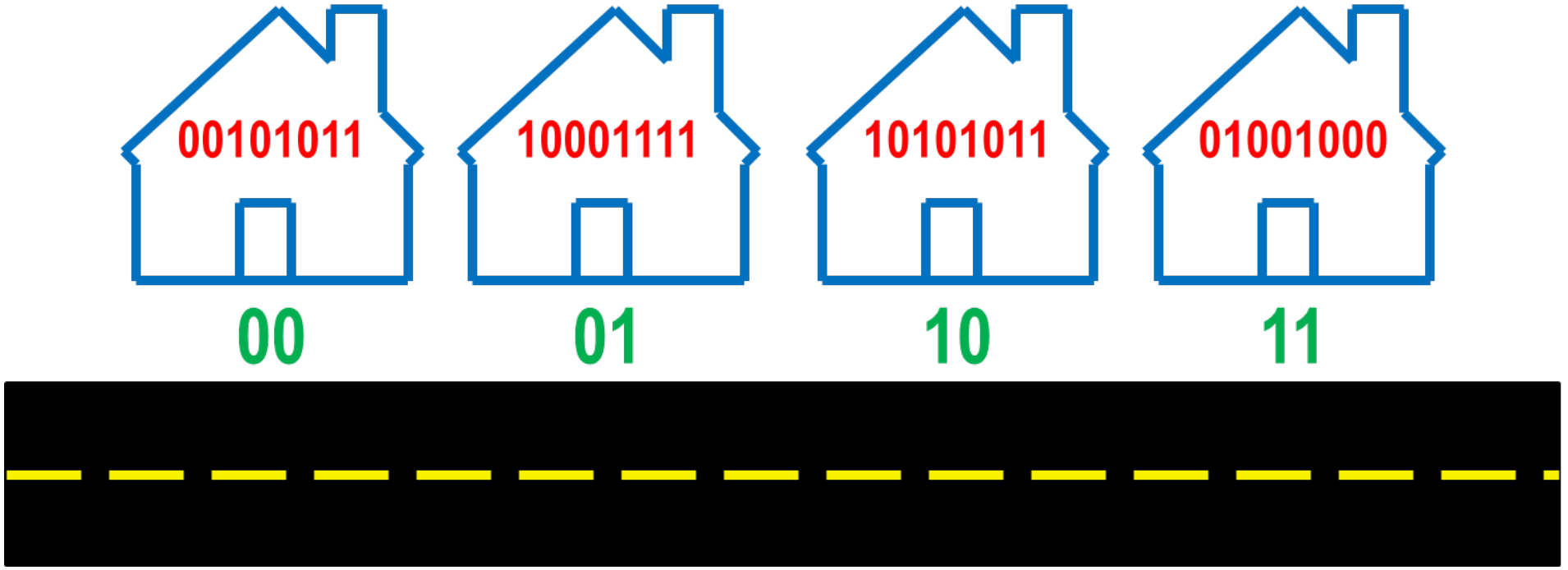
3 × 8

Address bits × Data bits


4 × 8
Address bits × Data bits
00101011
00101011
10001111
10001111
10101011
10101011
01001000
01001000
0000 0001 0010 0011 0100 0101 0110 0111
00101011
00101011
10001111
10001111
10101011
10101011
01001000
01001000
1000 1001 1010 1011 1100 1101 1110 1111

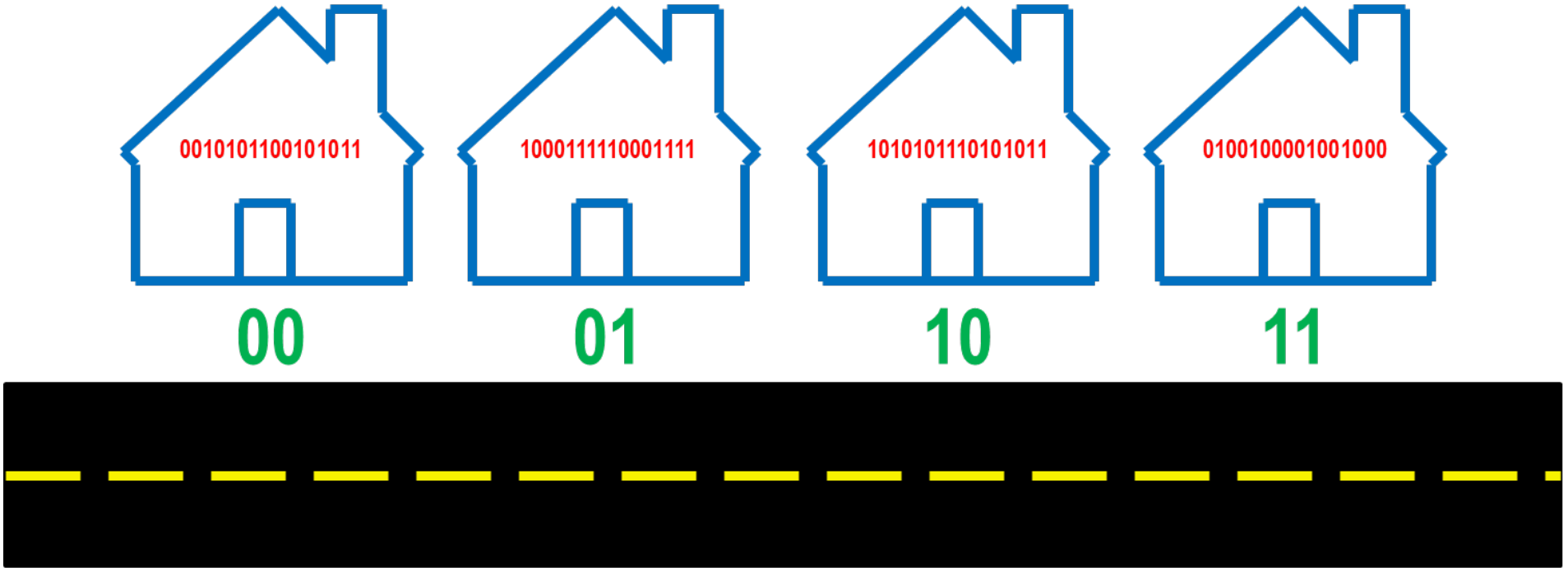
2 × 8

Address bits × Data bits



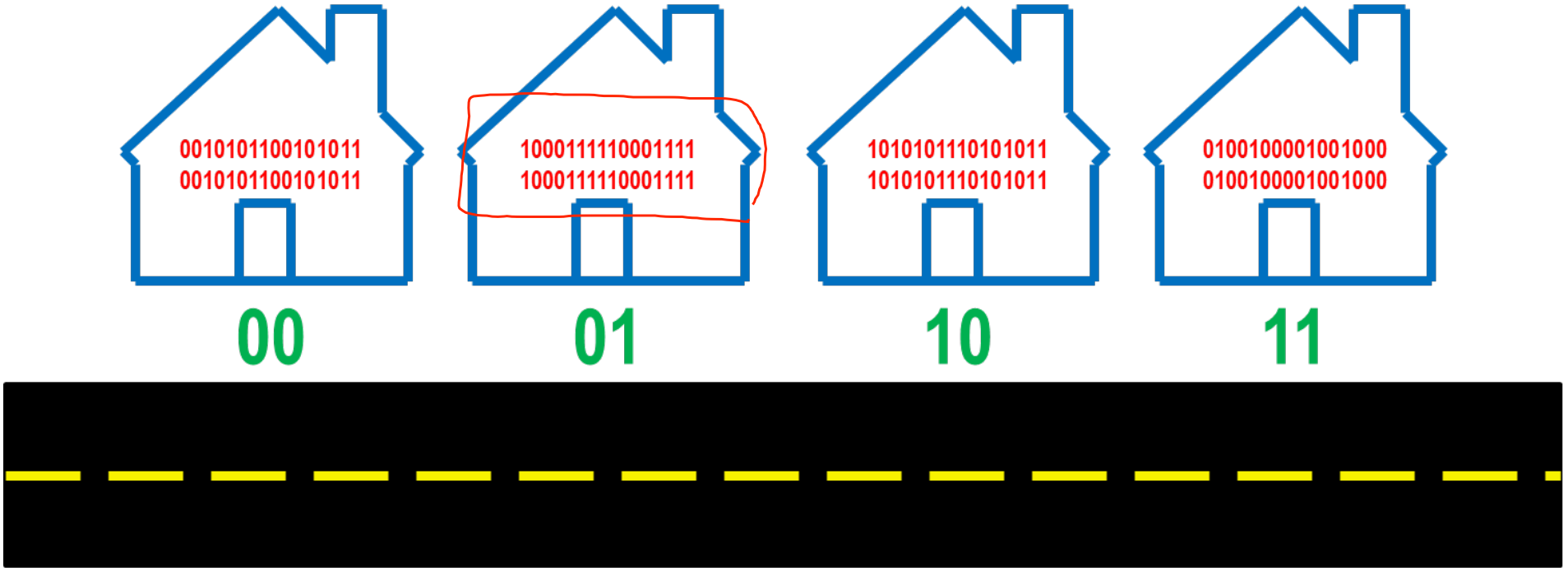
2 × 16

Address bits × Data bits



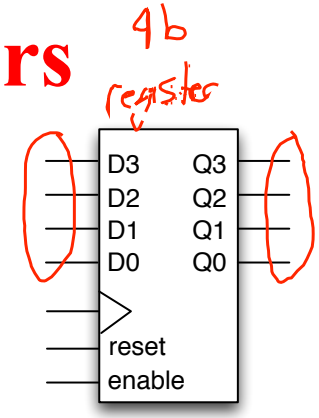
2 × 32

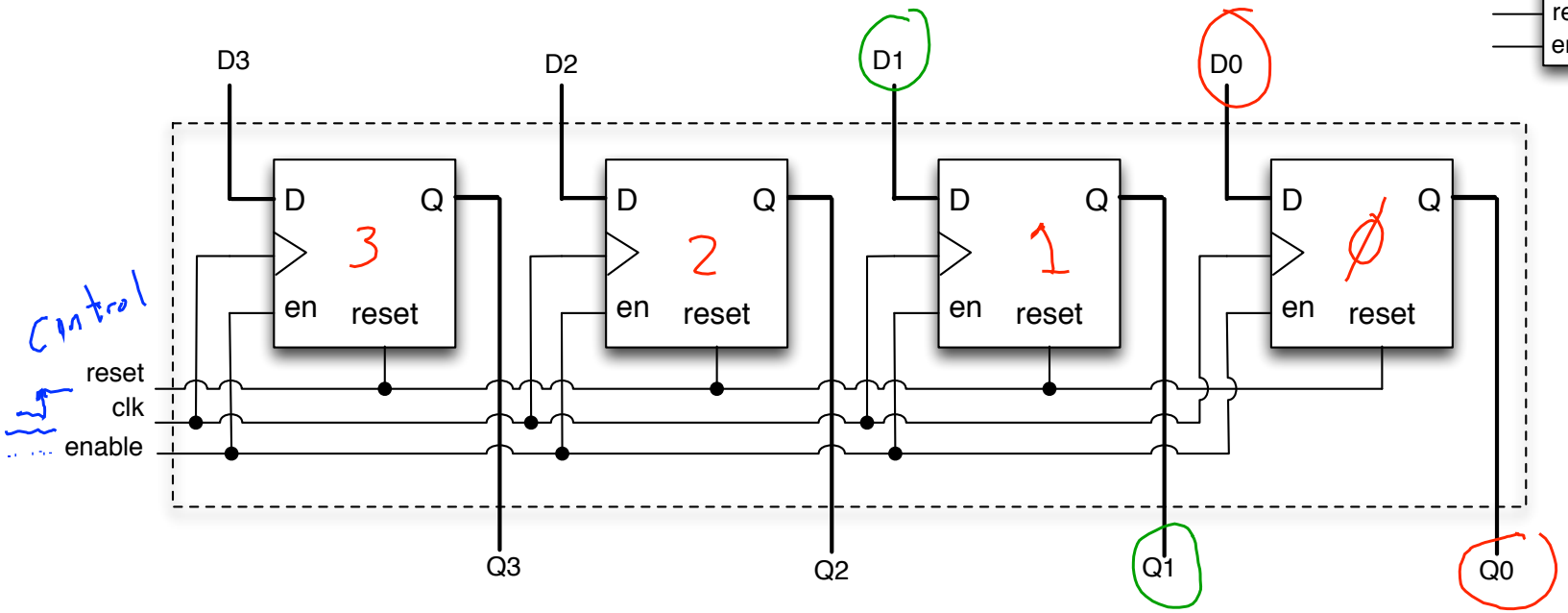
Address bits × Data bits



Use multiple flip flops to build registers

- Example 4-bit register made of four D flip flops
  - All control signals use the same input





Random Access Memory (RAM) is the hardware equivalent of an array

Addr

- RAM stores data at addresses
- All data has the same bit width
- Use brackets to access data at any address

M[Addr]

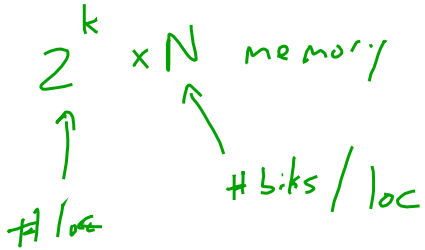
idx

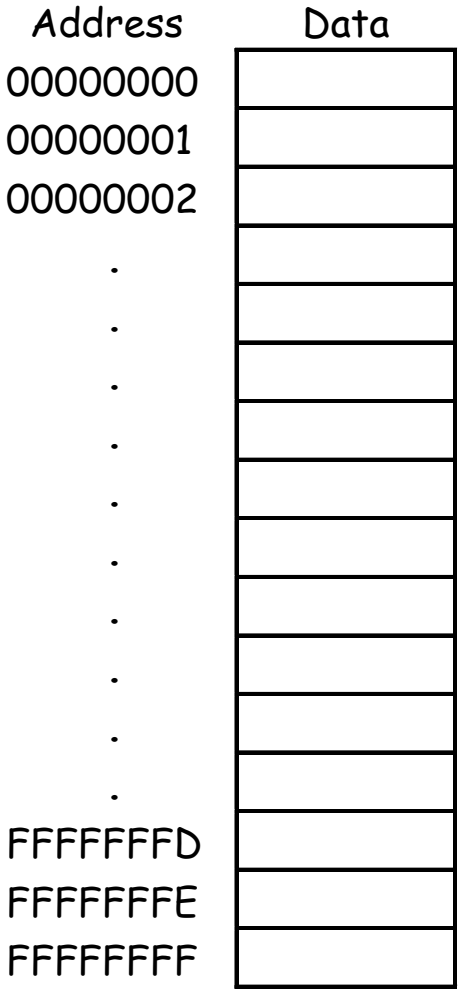
- Arrays store data at indices
- All data has the same type
- Use brackets to access data at any index

Array[idx]

RAM is the hardware equivalent of an array

- The address is an array index.
  - A *k*-bit address can specify one of 2^k words
- Each address refers to one word of data.
  - Each word can store *N* bits





A RAM should be able to

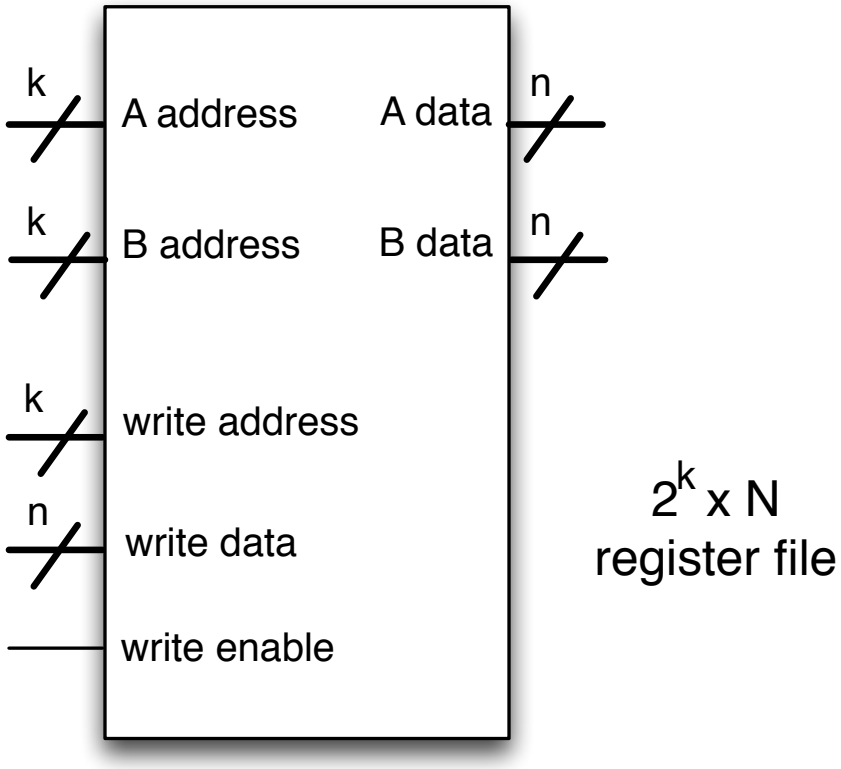
1. Store many words, one per address
2. Read the word that was saved at a particular address (??? = M[Addr])
3. Change the word that's saved at a particular address (M[Addr] = ???)

A Register File is a synchronous RAM

Use the letter R to indicate that the RAM is a register file rather than a generic memory (M)

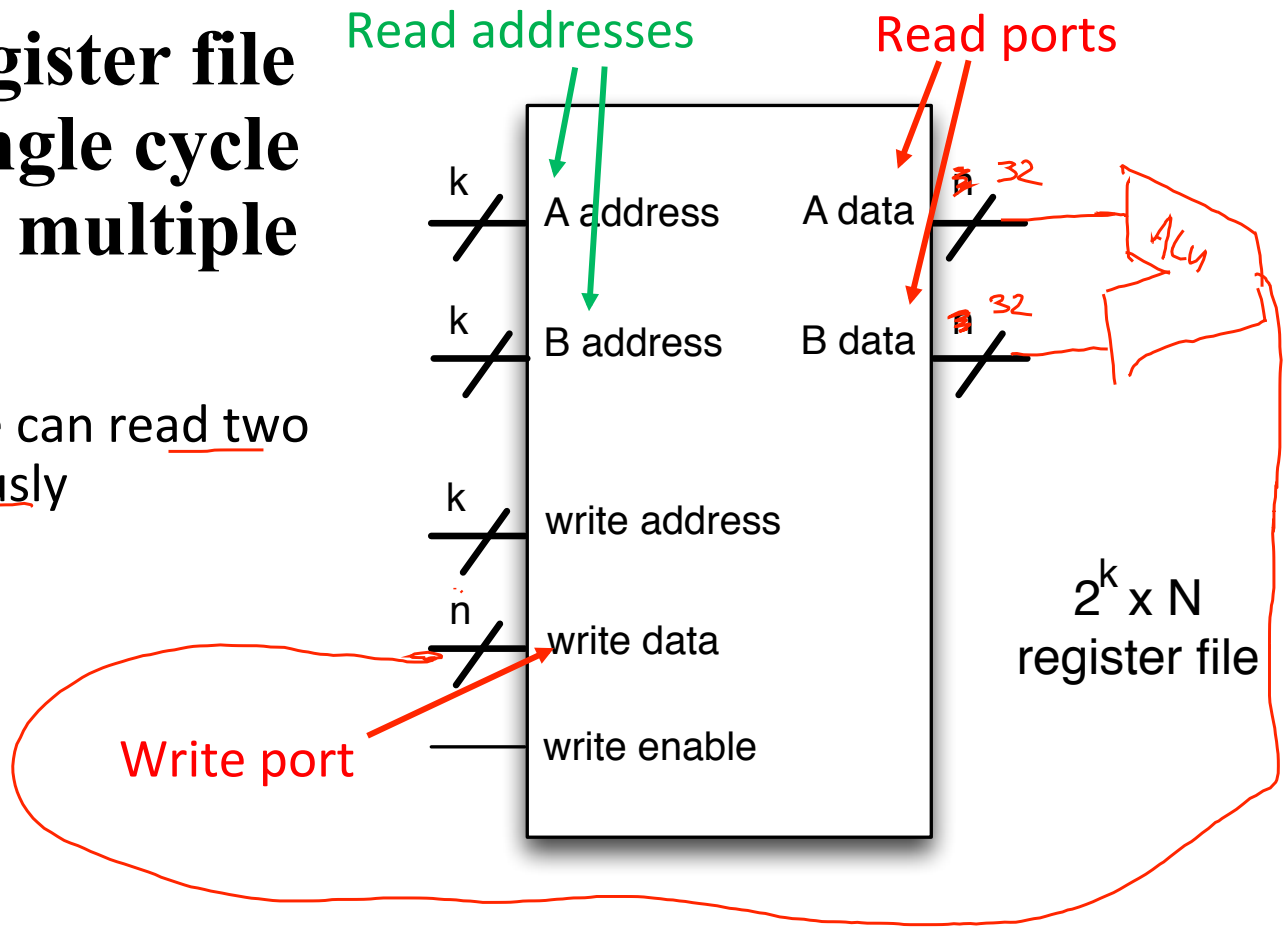
R[7]

R[Addr]



Our MIPS register file will enable single cycle operations on multiple data

- 2 read ports, so we can read two values simultaneously
- 1 write port



n = 32b

32 regs

$2^k = 32$

clicker question

2^7

We need to build a RAM that can store 128, 64-bit words and has one write port and three read ports

How many address bits does my RAM need for its **write** port?

A) 1

B) 6

C) 7

D) 64

E) 128

clicker question

We need to build a RAM that can store 128, 64-bit words and has one write port and three read ports

How many data bits does my RAM need for each **read** port?

A) 3

B) 6

C) 7

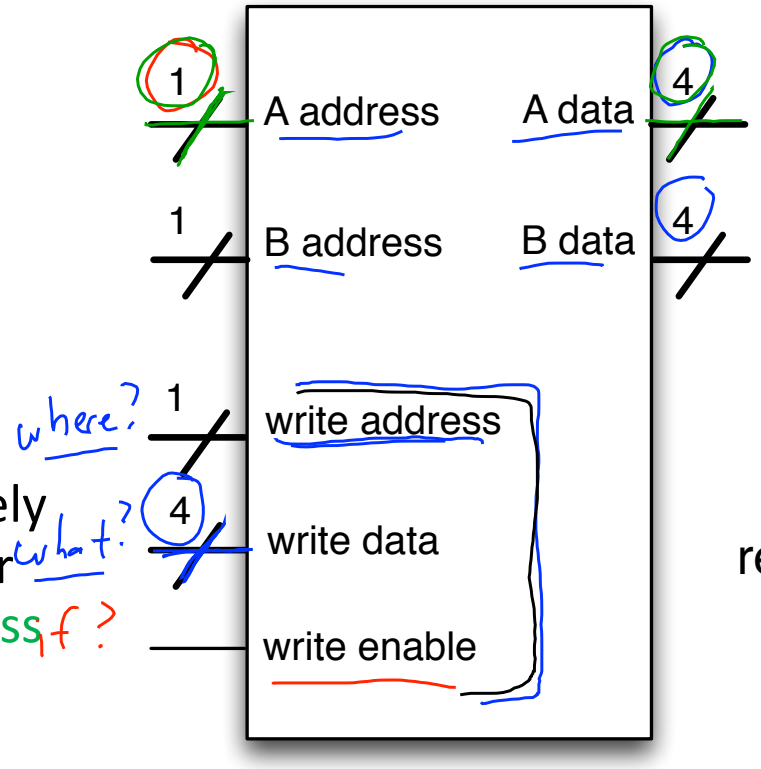
D) 64

E) 128

Let's build a 2-word memory with 4-bit words

A register file has 3 parts

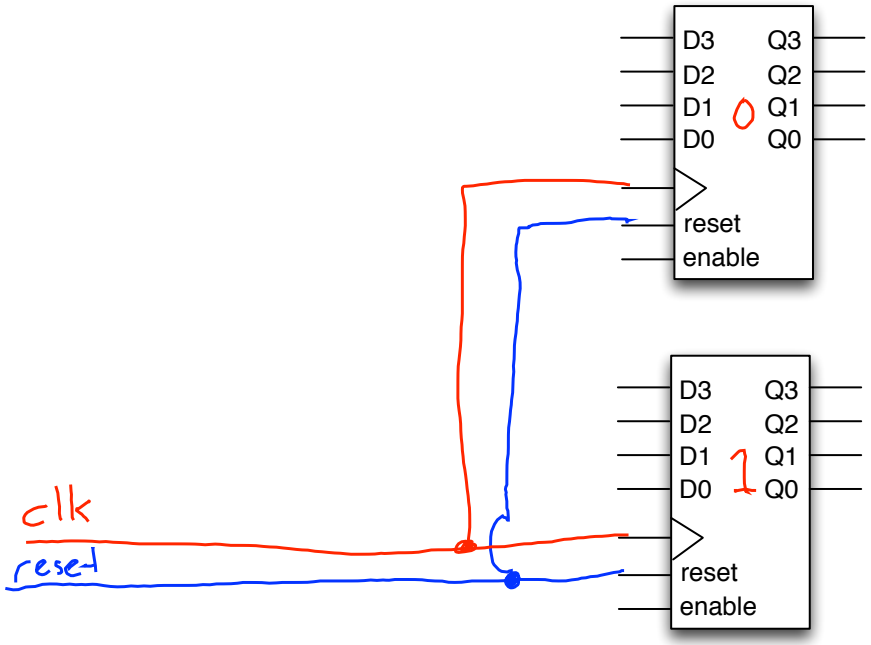
1. **The Storage**: An array of registers
2. **The Read Ports**: Output the data of the register indicated by read addresses
3. **The Write Port**: Selectively write data to the register indicated by write address



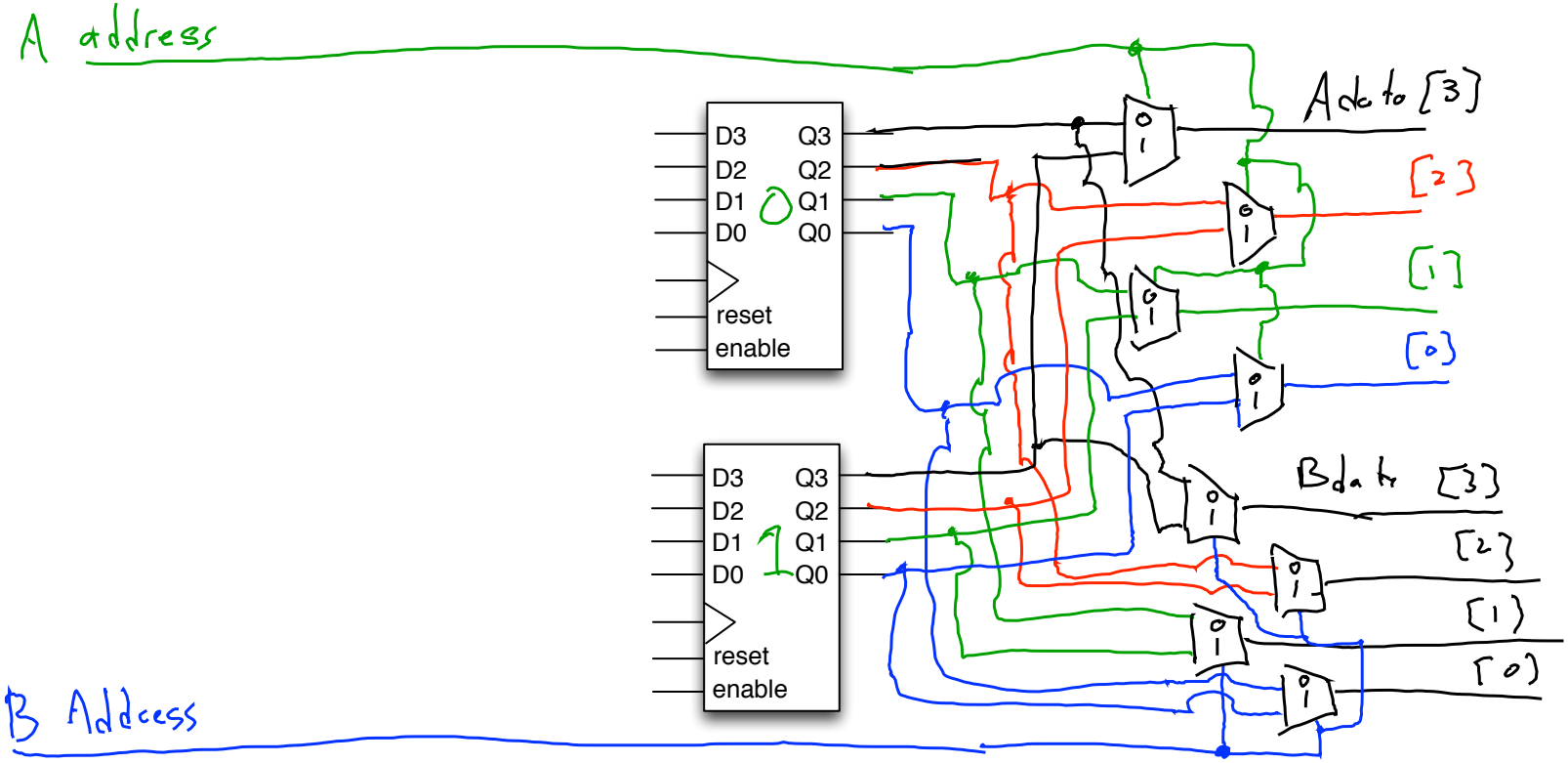
2^1 x 4 register file

Step 1: Allocate 1 register per address (2^1x4)

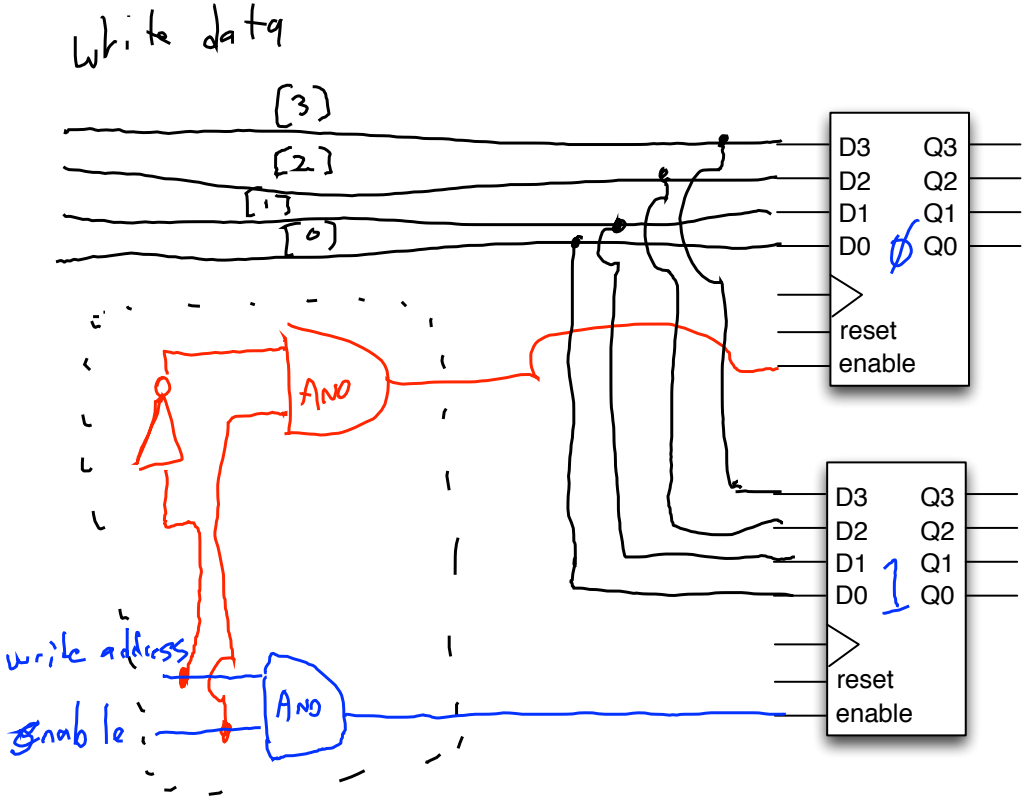
- Wire clocks and resets together to maintain synchronization



Step 2: Read ports use the address to select one register's data to output

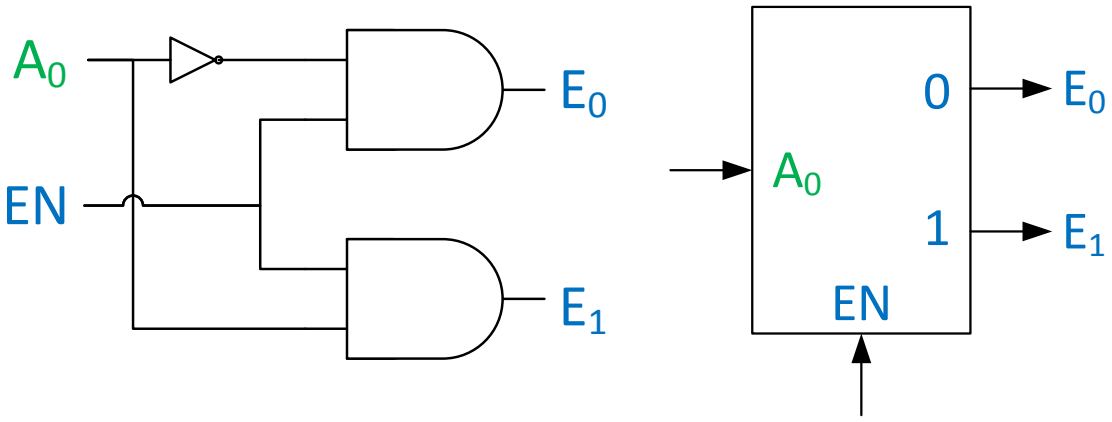


Step 3: Write ports decode the address to enable writing to exactly one register



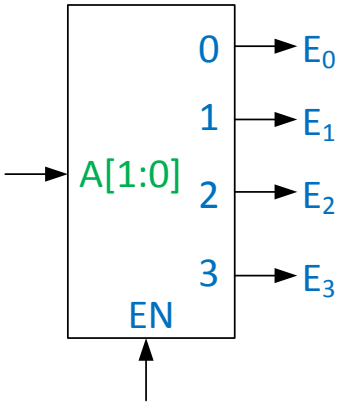
Decoders receive a binary code to generate control signals

- A 1-to-2 Binary decoder receives a 1-bit unsigned binary code and an enable signal to enable one of two devices

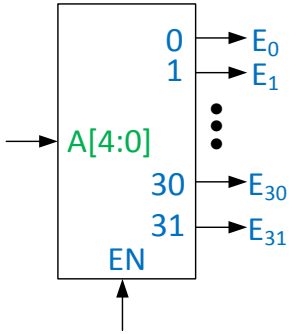


EN	Ao	(E1,Eo)
0	X	(0, 0)
1	0	(0, 1)
1	1	(1, 0)

n-to-2^n Binary decoders receive n-bit unsigned binary codes to enable one of 2^n devices

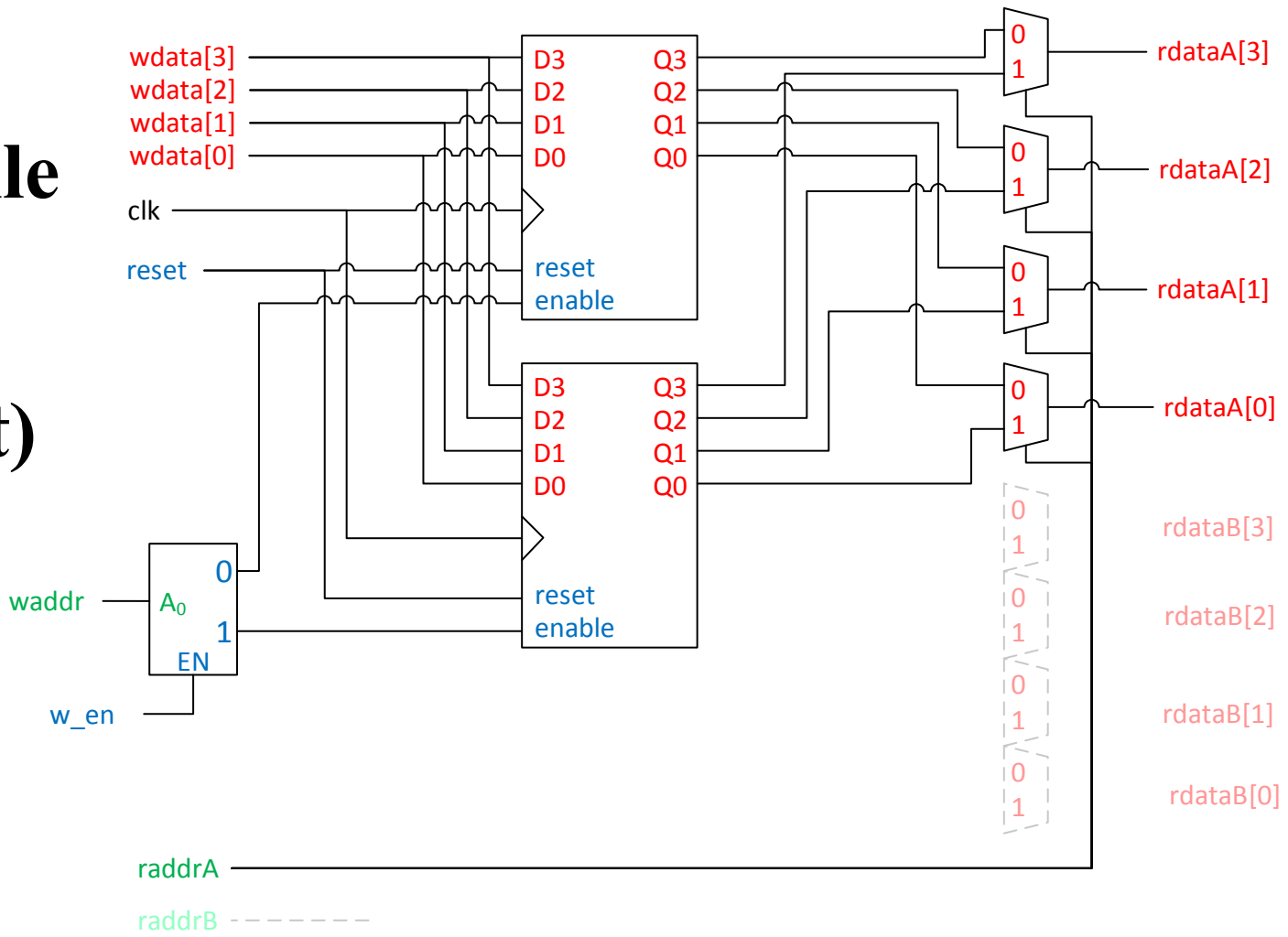


EN	A[1:0]	E[3:0]
0	X	(0,0,0,0)
1	(0,0)	(0,0,0,1)
1	(0,1)	(0,0,1,0)
1	(1,0)	(0,1,0,0)
1	(1,1)	(1,0,0,0)

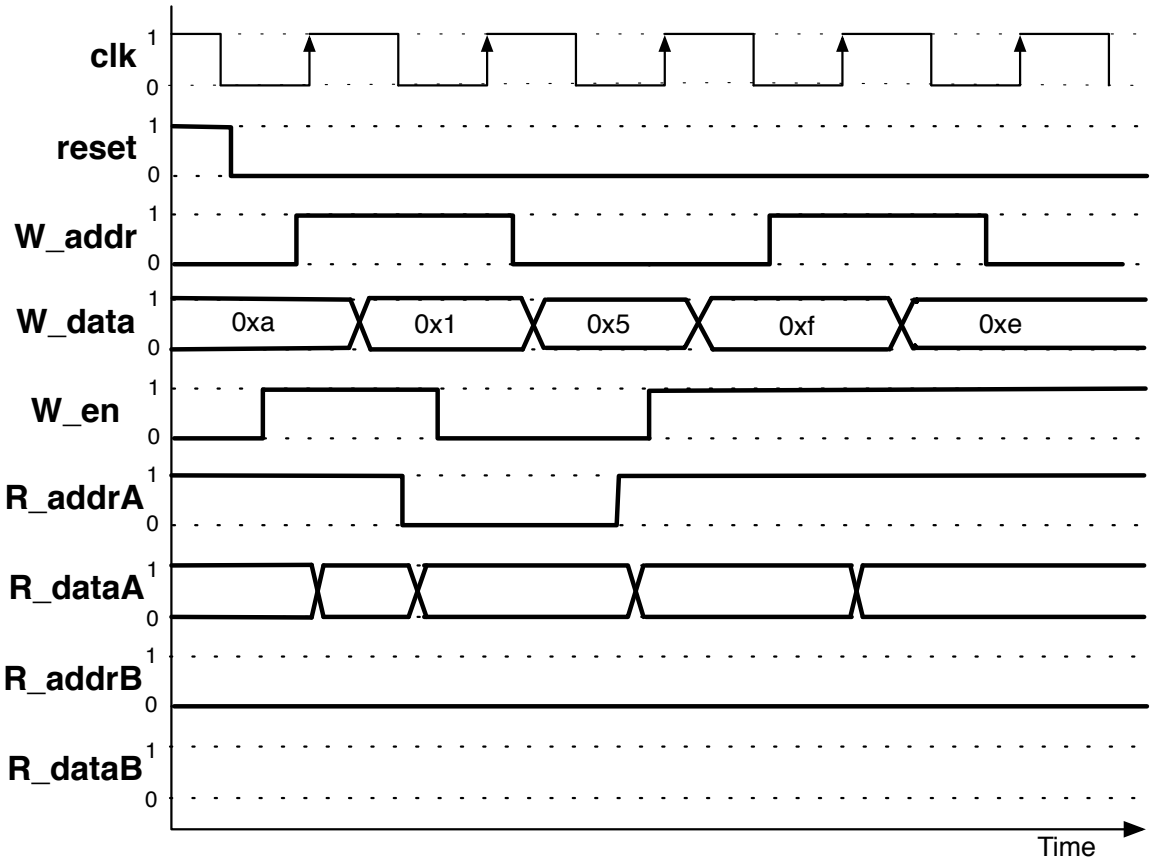


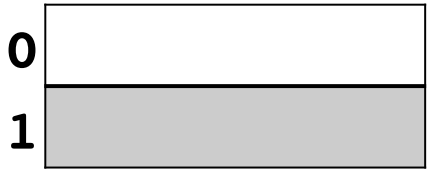
EN	A[4:0]	E[31:0]
0	X	0x0000
1	0	0x0001
1	1	0x0002
•••	•••	•••
1	30	0x4000
1	31	0x8000

2^1 x 4-bit register file (only 1 read port fully built)



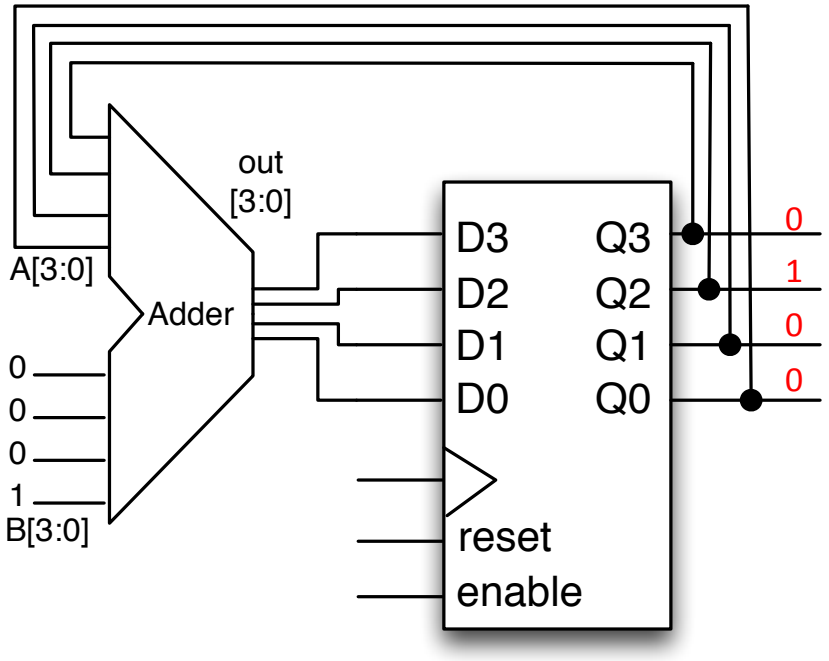
What does it do?



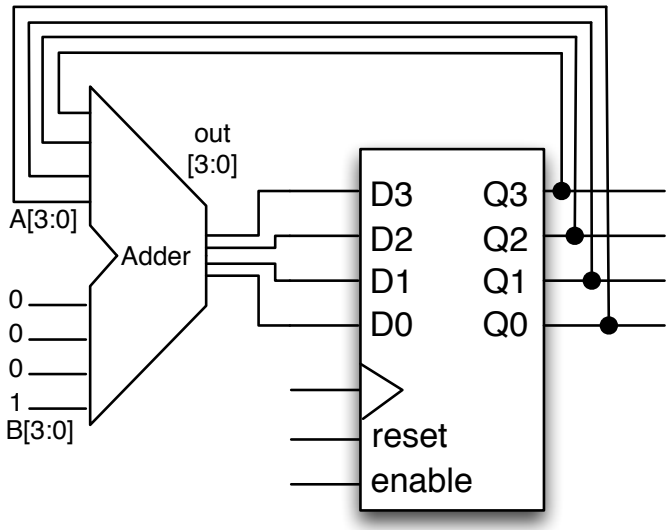


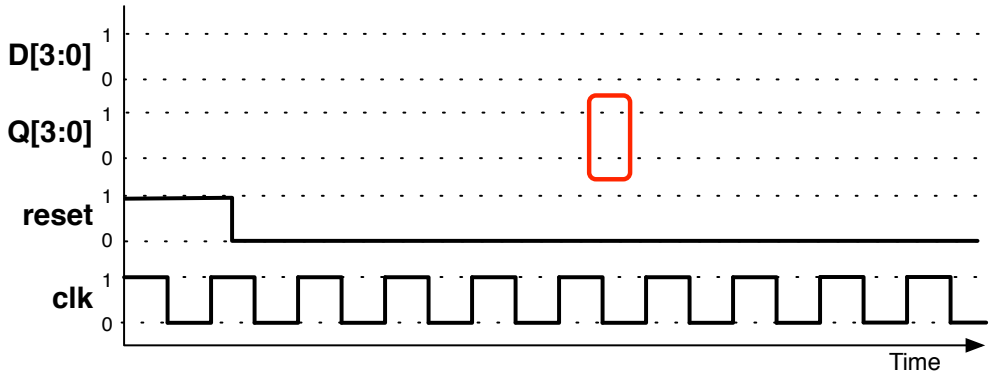
a) 0x0
b) 0xa
c) 0x1
d) 0x5
e) 0xf

What will Q[3:0] be during the next clock cycle?



a) 0
b) 1
c) 3
d) 4
e) 5





a) 0x0
b) 0x2
c) 0x4
d) 0x6
e) 0x8

Implementing counters