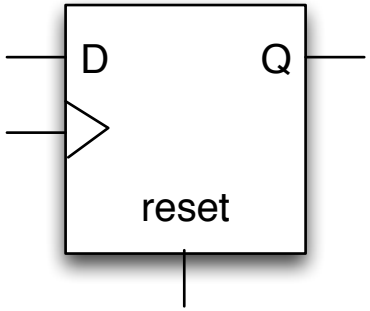
Registers and Register Files

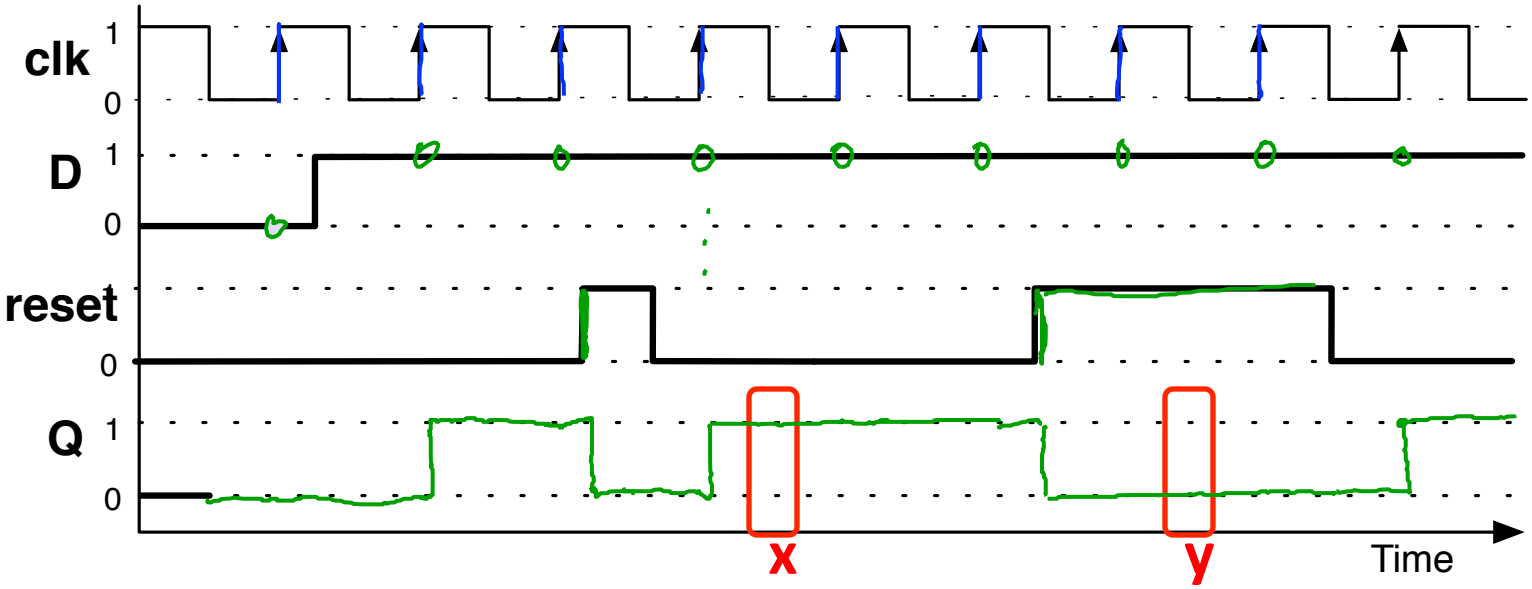
Today's lecture

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

Flip flop with asynchronous reset

- **Asynchronous = pertaining to operation without the use of fixed time intervals (opposed to synchronous).**
  - Processed immediately, ignores clock.
- **Reset = set the value to zero**

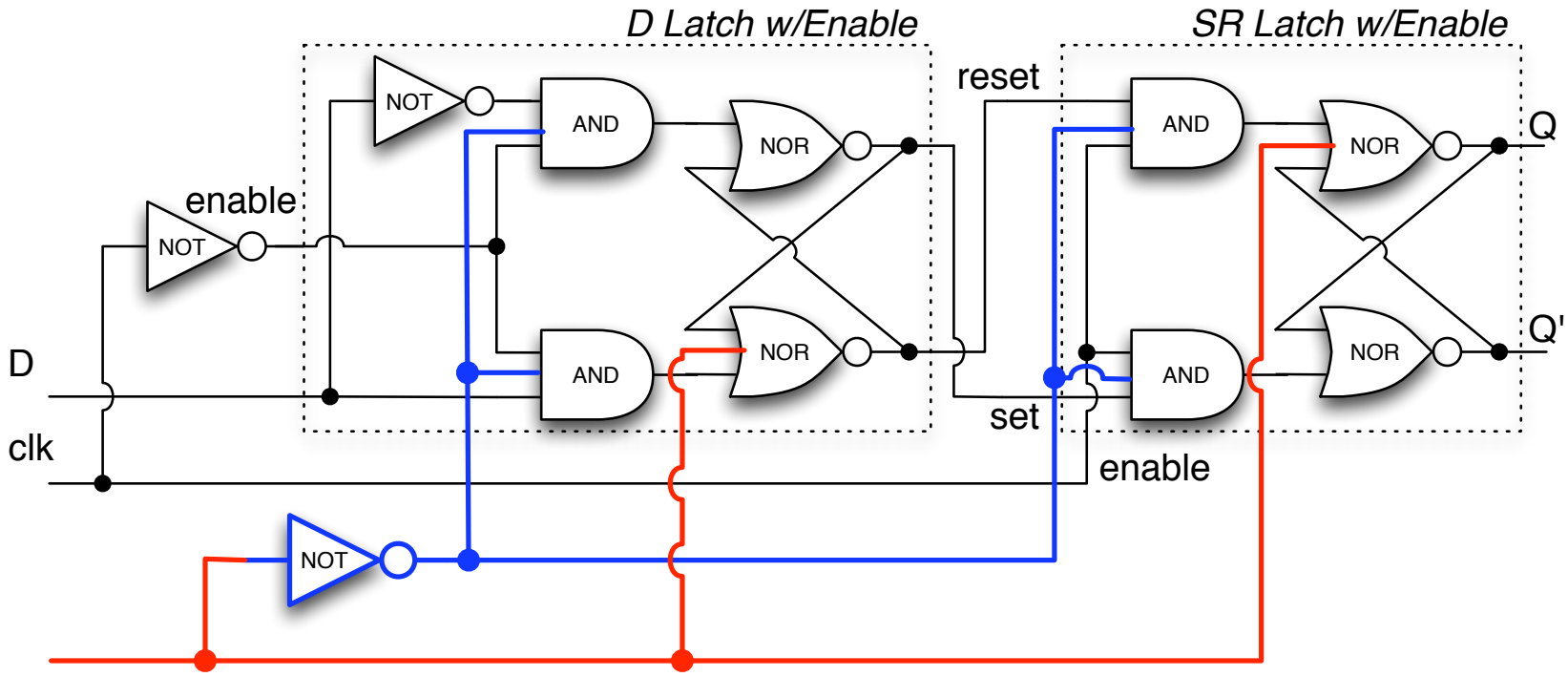




x,y =
a) 0,0
b) 0,1
c) 1,0
d) 1,1

Asynchronous Reset implementation

One example possible implementation



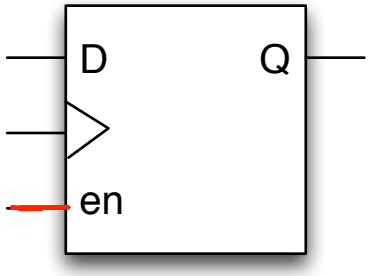
Ignores inputs and current state.

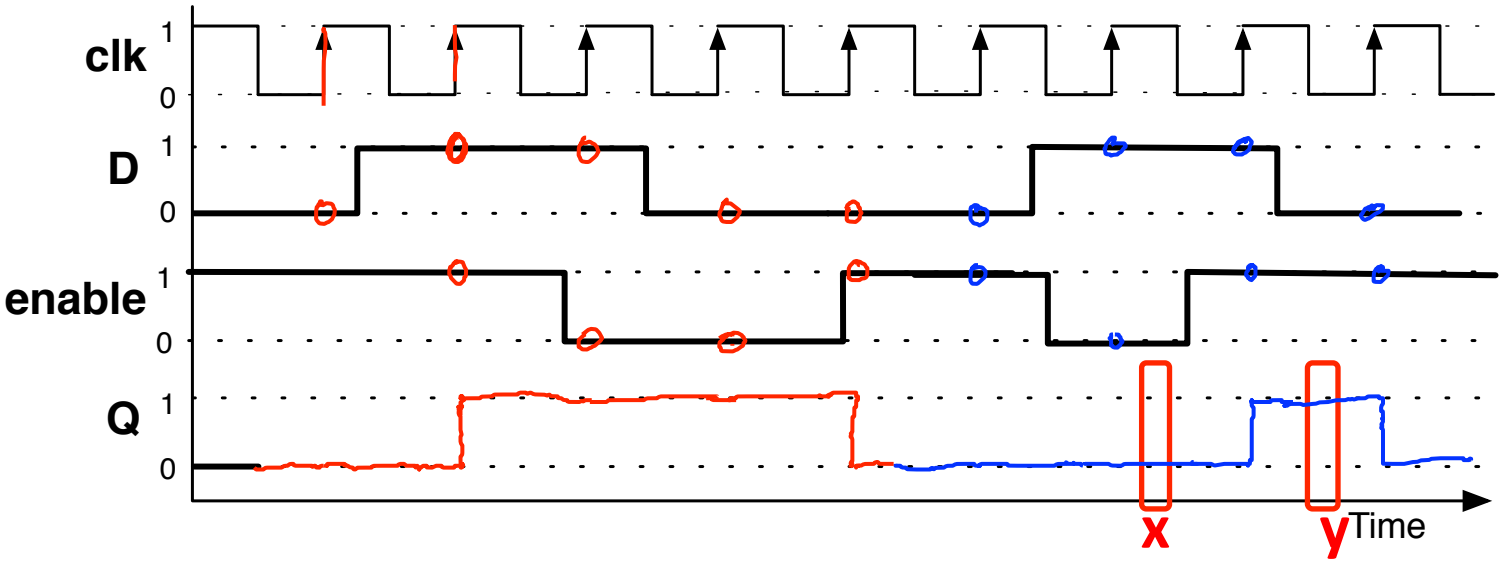
Forces Q output to zero.

(Not required material)

Flip flop with enable

- **When enable=0, Q output doesn't change on rising edge**
  - Behaves normally when enable=1





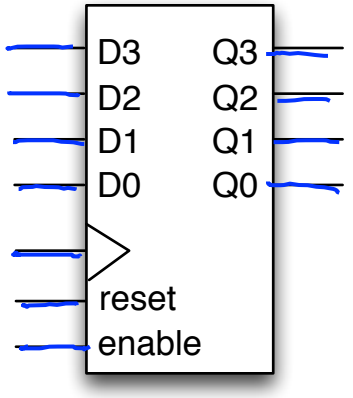
x,y =

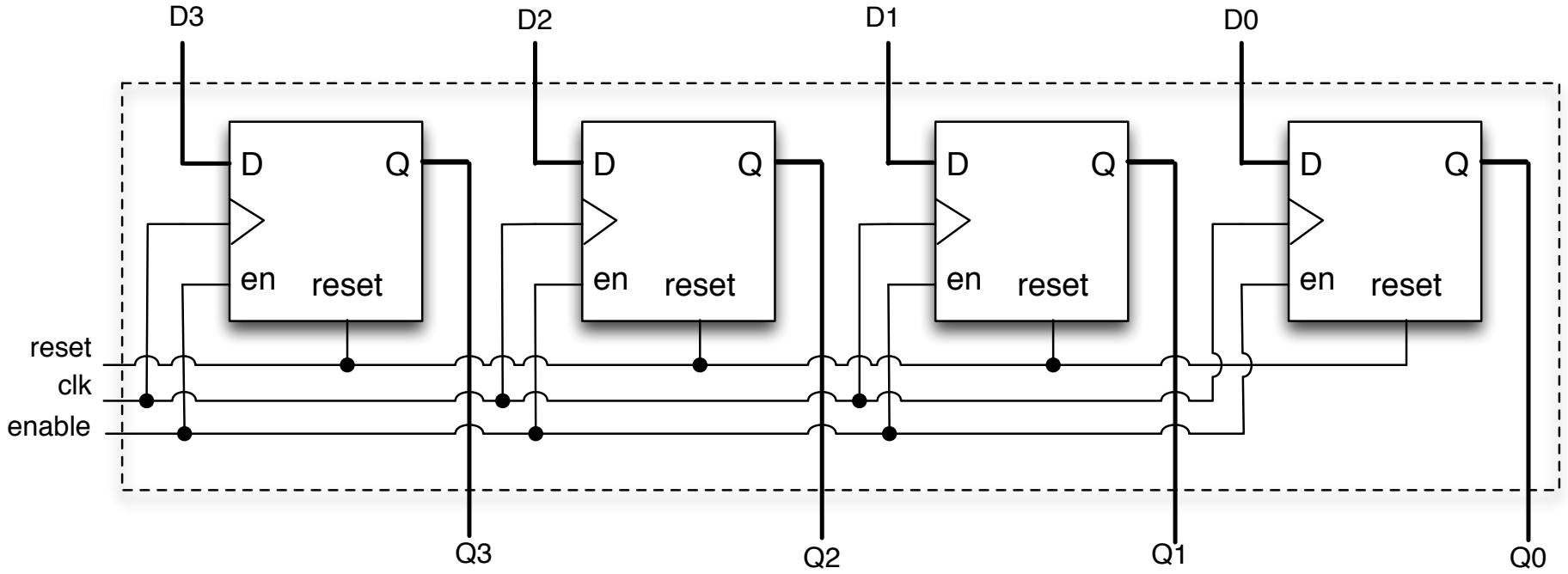
a) 0,0
b) 0,1
c) 1,0
d) 1,1

How can we store more than 1 bit?

4b - register

- **We build registers out of flip flops.**
  - Example 4-bit register made of four D flip flops
    - 1 data input, 1 data output per flip flop
    - All control signals use the same input





Introduction to RAM

- **To provide lots of storage we use Random-access memory, or RAM.**
- **Remember the basic capabilities of a memory:**
  - It should be able to store a value.
  - You should be able to read the value that was saved.
  - You should be able to change the stored value.
- **A RAM is similar, except that it can store *many* values.**
  - An address will specify which memory value we're interested in.
  - Each value can be a multiple-bit word (e.g., 32 bits).
- **We'll refine the memory properties as follows:**

A RAM should be able to:

- Store many words, one per address

- Read the word that was saved at a particular address

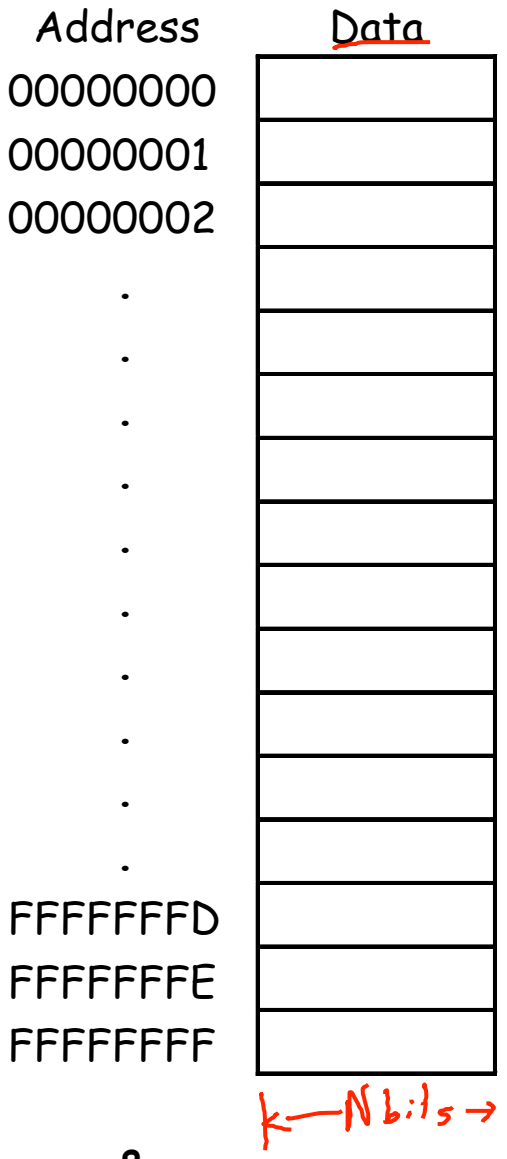
- Change the word that's saved at a particular address

A picture of memory

- **You can think of memory as being an array of data.**
  - The address serves as 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

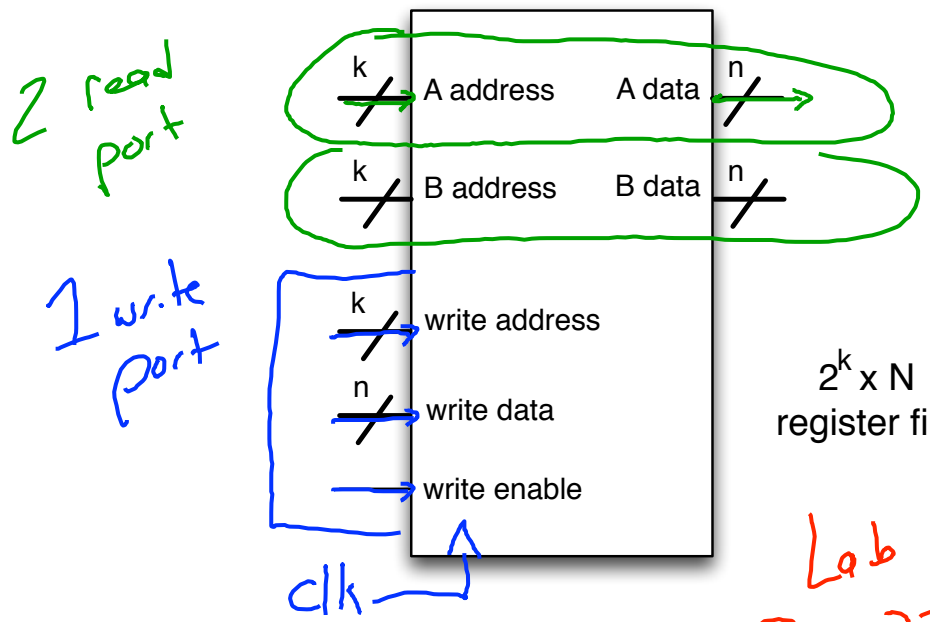
$2^k * N$ memory

- **You can read or modify the data at any given memory address, just like you can read or modify the contents of an array at any given index.**
  - This is what "random access" means.



Register File: a special kind of memory

- **We'll focus first on a special kind of RAM: a register file**
- **Our register file will have:**
  - 2 read ports, so we can read two values simultaneously
  - 1 write port



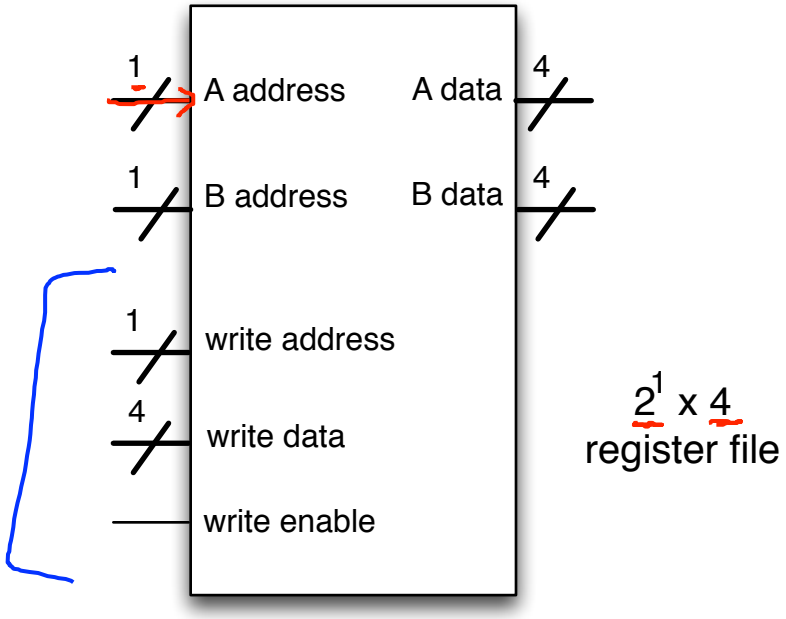
- **This will allow us to:**
  - Read 2 numbers
  - Add them together
  - Store the result back in the register file

Lab
32 x 32b
registers
big word size

Register file implementation

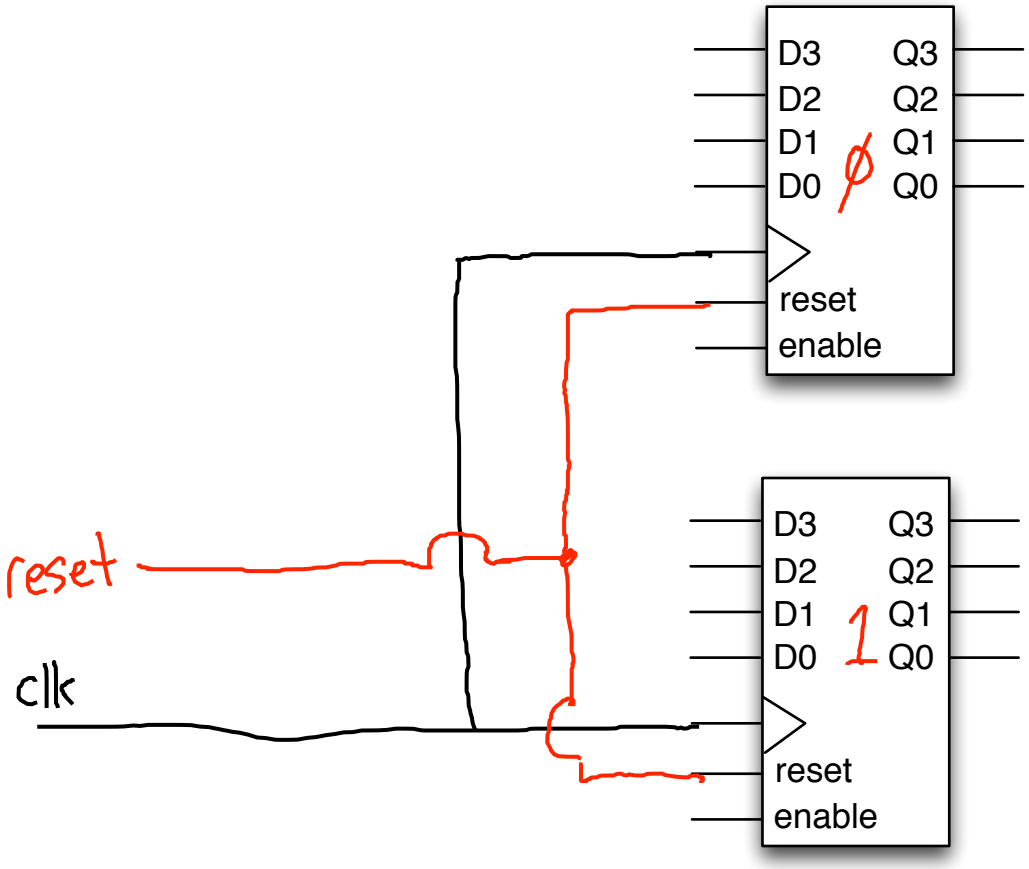
- **A register file has 3 parts**
  - **The Storage**: An array of registers
  - **The Read Ports**: Output the value of selected register
  - **The Write Port**: Selectively write one of the registers

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



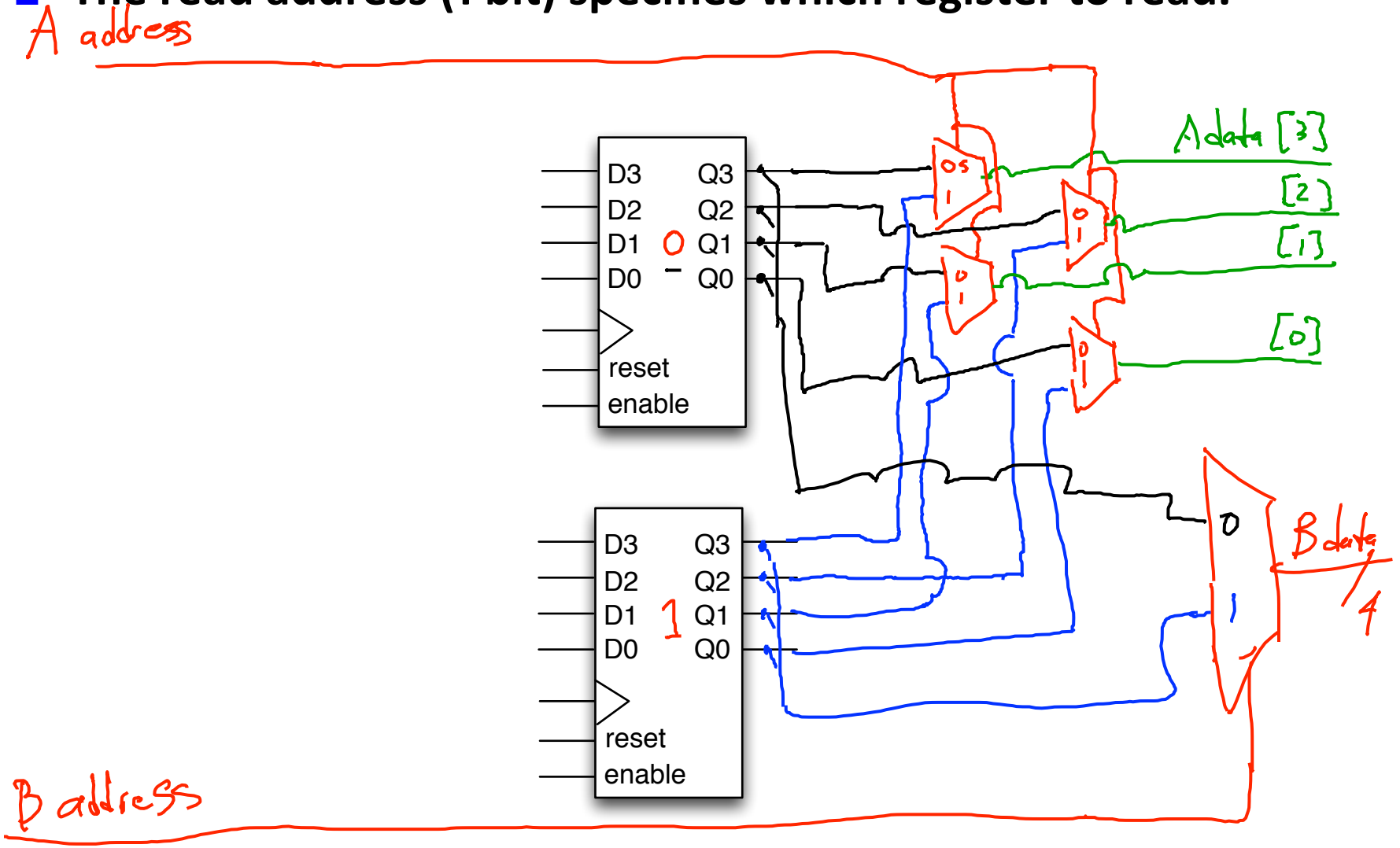
Step 1: Allocating the storage

- **We need two 4-bit registers**
  - We'll wire their clocks and resets together



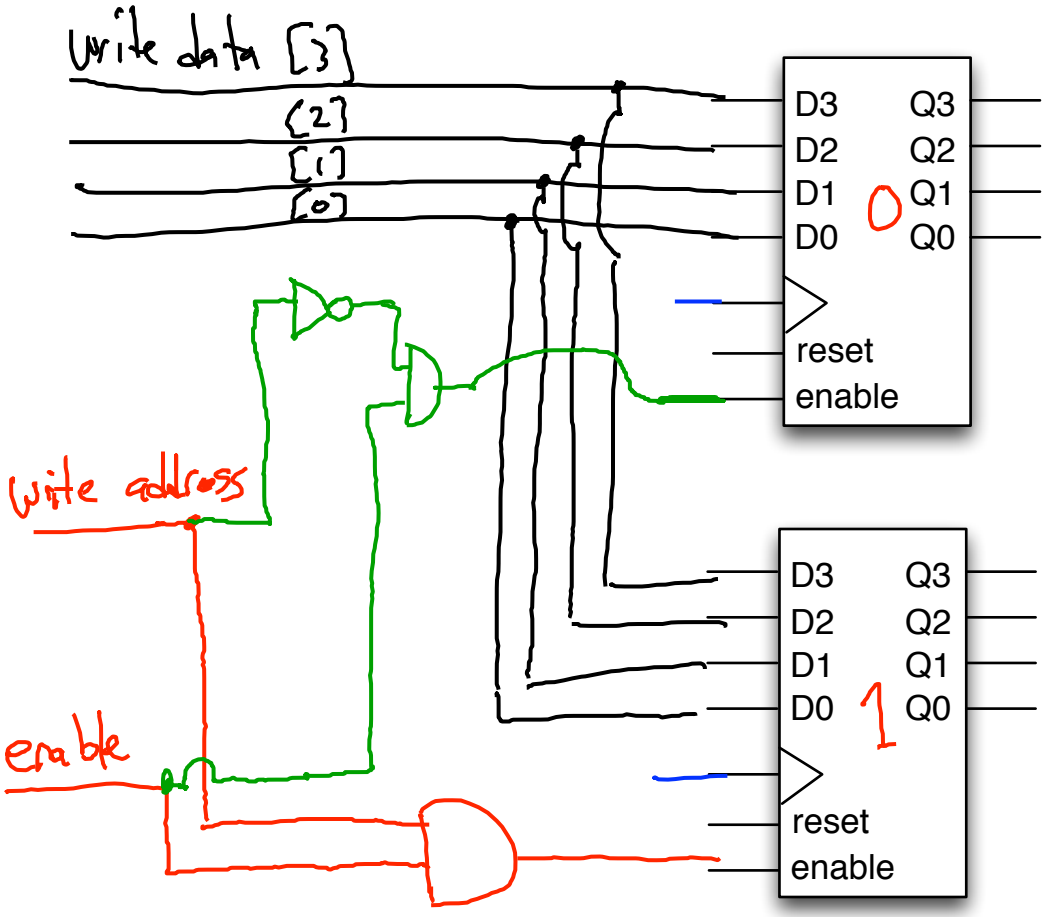
Step 2: Implementing the read ports

- The read address (1 bit) specifies which register to read.



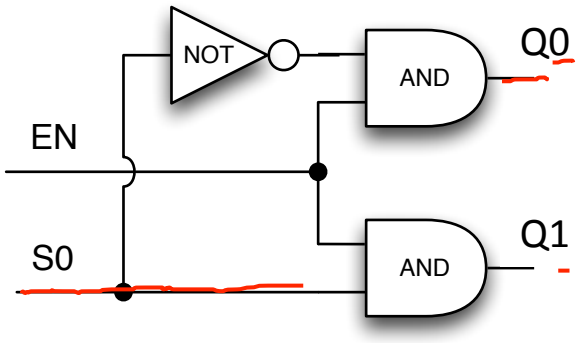
Step 3: Implementing the write ports

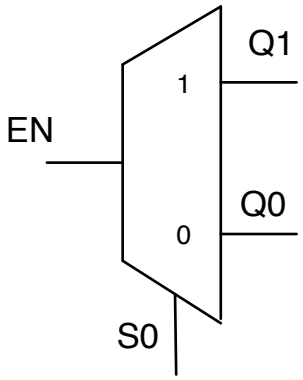
- The write address (1 bit) specifies which register to write.



Decoders

- **This circuit is a 1-to-2 decoder**
  - It decodes a 1-bit address, setting the specified output to 1
    - Assuming the circuit is enabled

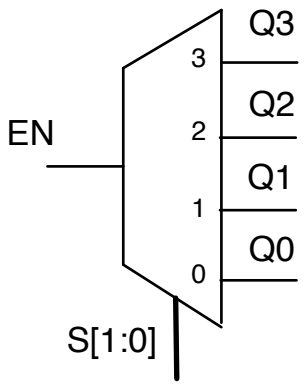




EN	S0	(Q1,Q0)
0	X	(0, 0)
1	0	(0, 1)
1	1	(1, 0)

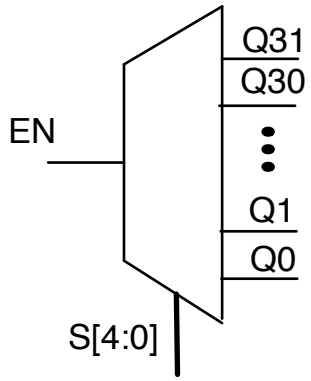
Scaling Decoders

- **Decoders can be generalized as follows**
- **A n-to-2^n decoder:**
  - Has a 1-bit enable input, and an n-bit select input
  - Has 2^n outputs
  - All the outputs are zero, except the selectth if enable = 1
    - If enable = 0, all outputs are zero.
- **A 2-to-4 decoder:**



EN	S[1:0]	Q[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)

5-to-32

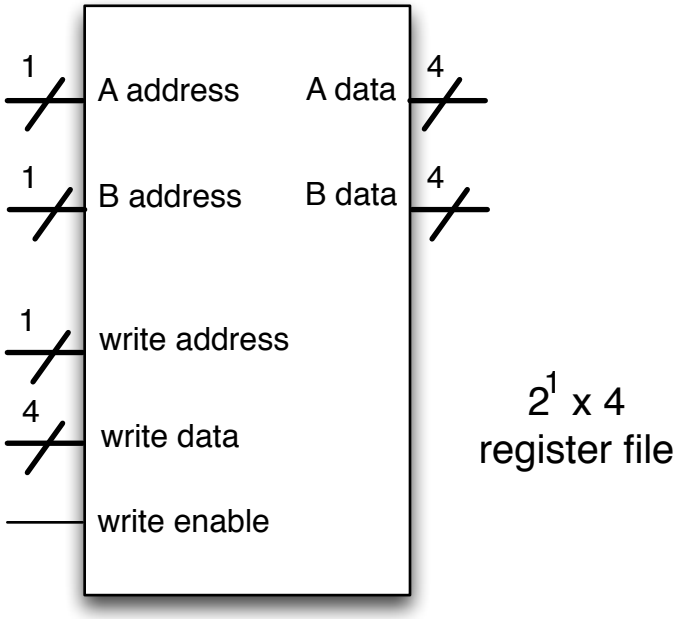


EN	S[4:0]	Q[31:0]
0	X	0x0000
1	0	0x0001
1	1	0x0002
1	…	…
1	30	0x4000
1	31	0x8000

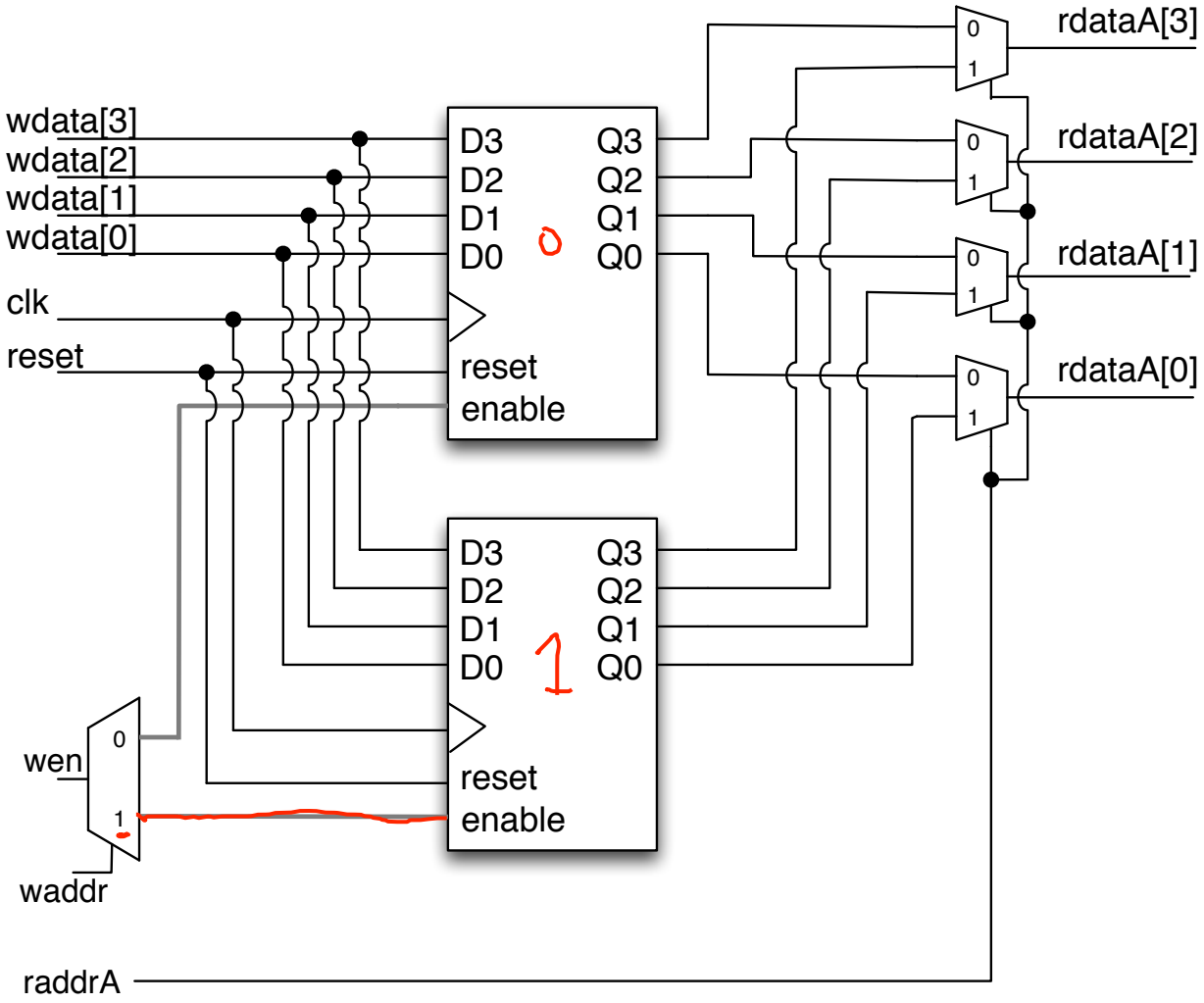
Register file implementation

- **A register file has 3 parts**
  - **The Storage**: An array of registers
  - **The Read Ports**: Output the value of selected register
  - **The Write Port**: Selectively write one of the registers

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



2 x 4-bit register file (only 1 read port shown)



What does it do?