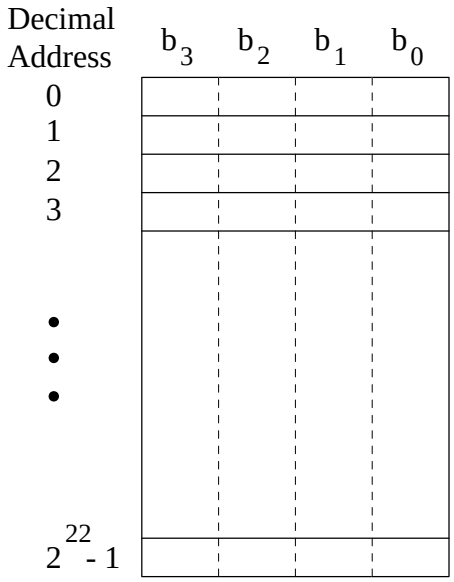
Implementation of Large Memory Chips

Consider a 4M x 4-bit chip which has a 22-bit addresses since $4M = 2^{22}$.

Logically/Externally, we view a 4M x 4-bit memory as pictured below with:

- each memory word made up of four bits: $b_3\ b_2\ b_1\ b_0$



When we want to read a word, we supply a 22-bit address, and receive the corresponding 4-bit word.

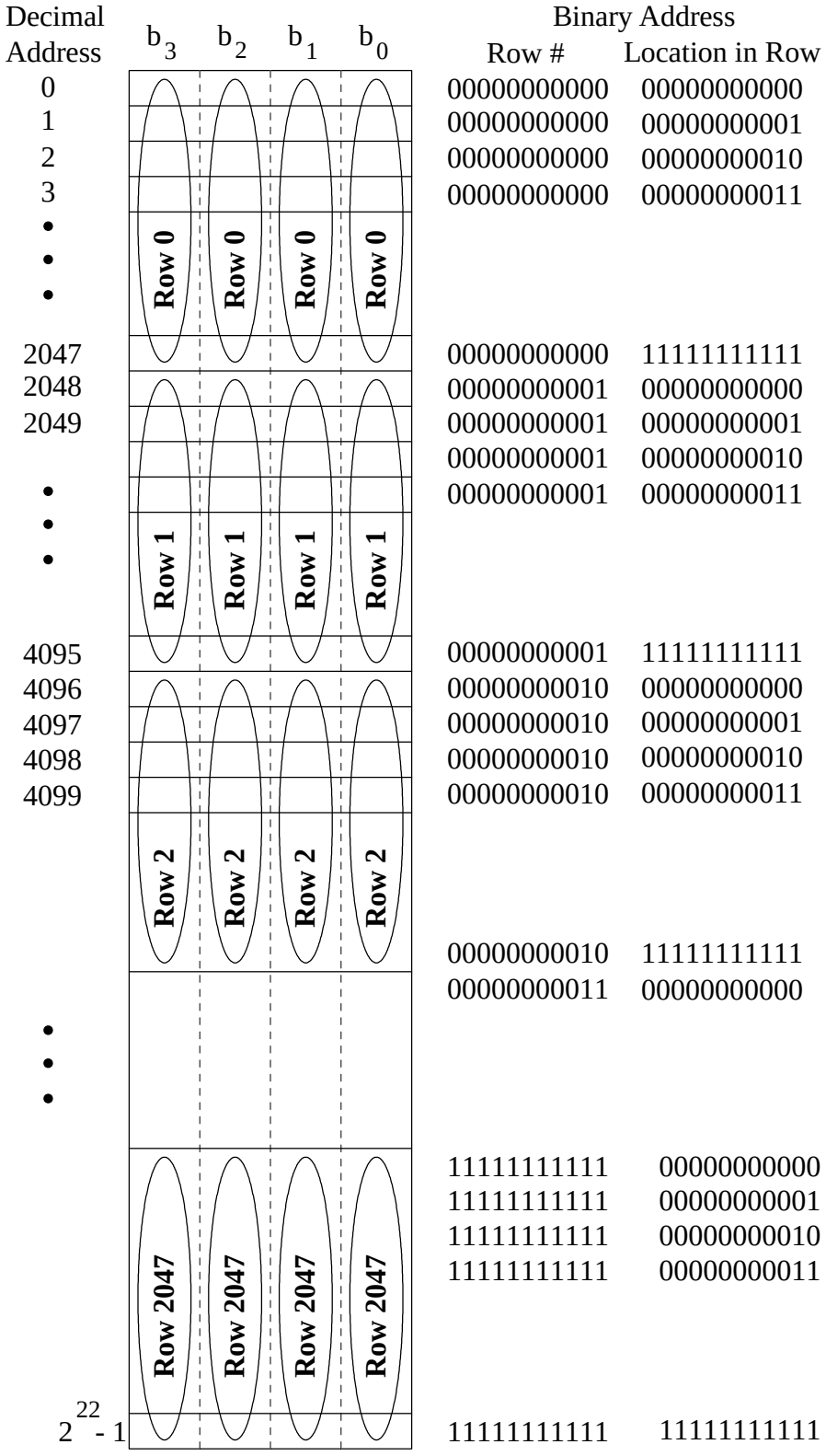
The register-file implementation (see handout), does not scale well for large memories for several reasons:

- the number of gates in the address decoder (and MUXs) grows exponentially with the number of bits in the address.
- lots of wires into/out of the memory chip for address, data, and control

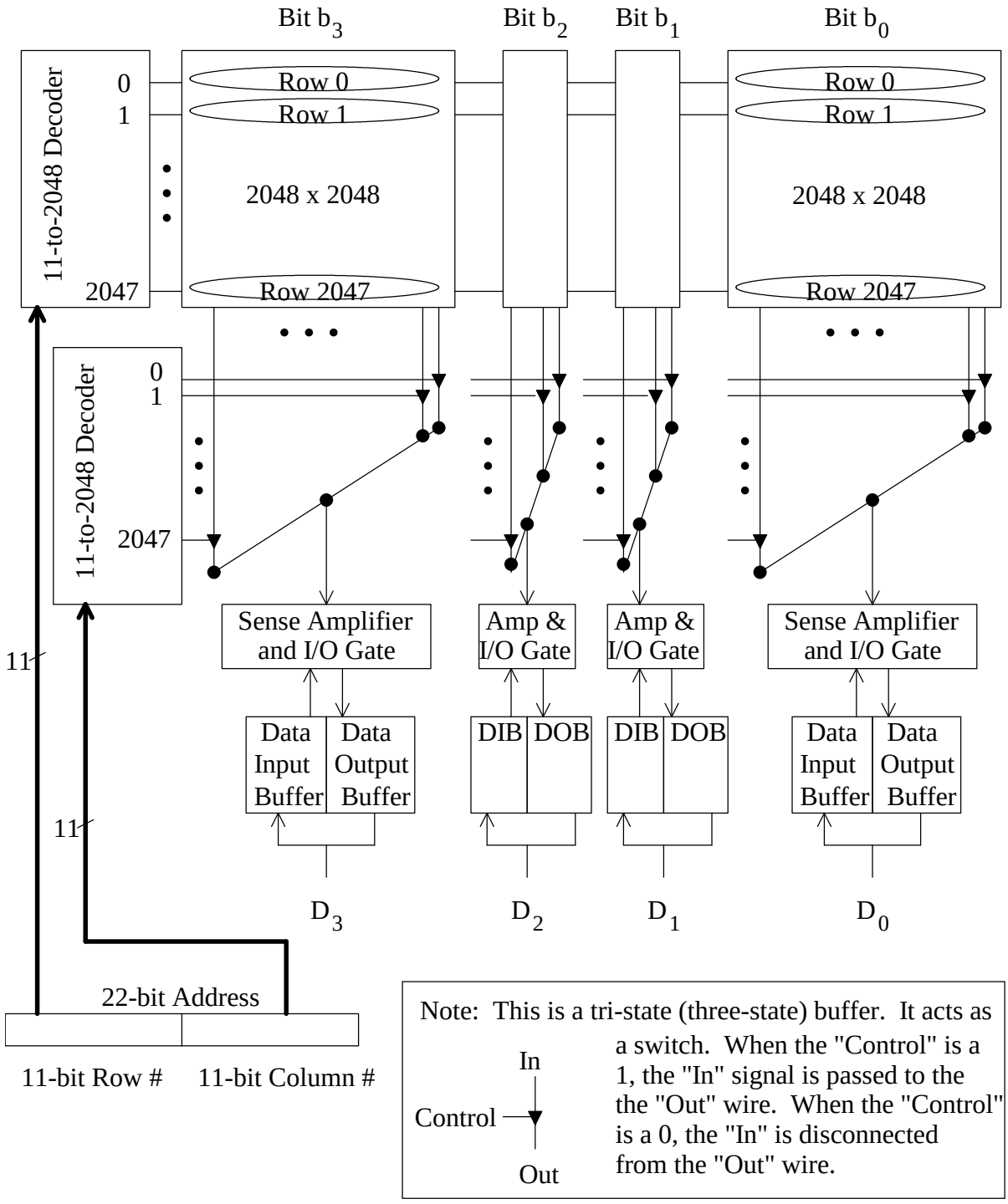
These problems are solved by

- using square-array of bits and decoding the address in two parts (row then column number)
- eliminate MUX's by using tri-state buffers
- single-port RAM memory - data wires shared for reading and writing

To help us see how the 4M x 4-bit memory gets mapped to the 2048 x 2048 x 4 memory array on the next page consider splitting memory into 2048 word blocks as shown below.



Each bit of a word is split into a separate 2048 x 2048 square memory “array”. Each of these square memory arrays is 2048 x 2048 = 2^{11} x 2^{11} = 2^{22} = 4M bits.
The 22-bit address of the 4M memory is split into two 11-bit parts. The upper 11-bits is first used to activate the correct row within the square memory arrays. Of the 2048-bit row that is read from each memory array, we are interested in only one bit. The lower 11-bits of the address specifies the location of the desired bit within the 2048-bit row.



Note: This is a tri-state (three-state) buffer. It acts as a switch. When the "Control" is a 1, the "In" signal is passed to the the "Out" wire. When the "Control" is a 0, the "In" is disconnected from the "Out" wire.

Implementing Large Memory with Smaller Chips

Consider for example, implementing 4M x 32 bits memory with 256KB x 1 bit chips.
The 256KB x 1 chips are implemented as square arrays of 512 x 512.

We will use an two-dimensional array of the 256KB x 1 chips to implement the larger memory.

The number of chips per row would be 32 bits / 1 bit = 32 chips.
The number of chips per column would be 4M / 256K = $2^{22} / 2^{18} = 2^4$ = 16 chips per column.

The 22-bit address of the 4M x 32 bit memory would be split up as follows: