

Lecture 16

Randon Access Memory and the Fetch Cycle

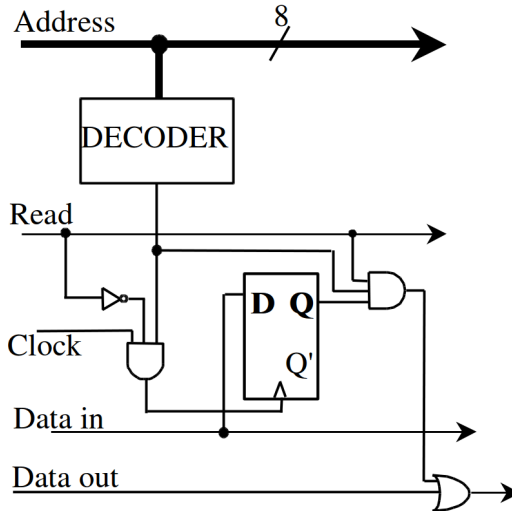
Random Access Memory

- We continue to put together a computer system, and in this lecture add the memory.
- We will consider first the implementation of random access memory, and then turn to how it is used by the central processor.

One Bit Memory

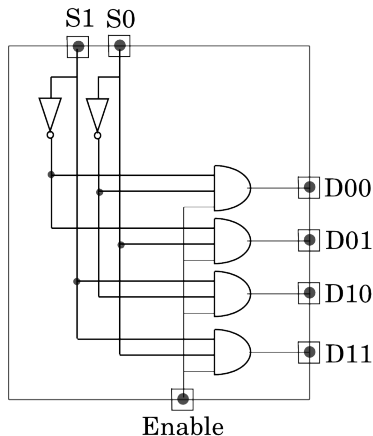
- A single D-type flip flop is a one bit memory
- To use it as such we need to give it a unique address
- An address is simply a binary number
- A binary number can be uniquely identified by a decoder.

One Bit Static RAM



Reminder

A decoder or demultiplexer is an old friend. It might have helped you to do the coursework in 30 seconds! It converts a binary address to a unary representation.



One Bit Static RAM

The one bit static RAM is an asymmetric circuit:

- for reading it is a combinatorial circuit;
- for writing it is a sequential circuit, the address and data must be present and correct when the clock pulse sets the flip flop.

RAM circuits conforming to this pattern are called static RAMs, and are only used in special applications because of their large size.

Buses

Buses are data highways which are common to many circuits.

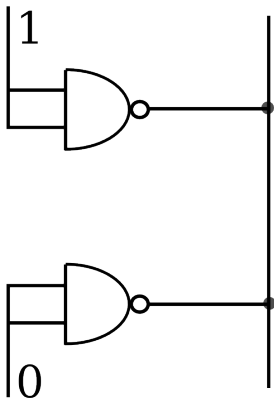
- The address lines that go to many one bit static RAM circuits are referred to as the address bus.
- The data-in and data-out lines from several one bit static RAM circuits are grouped together in bytes or words as data buses.
- The control lines (read and clock) also form a bus

Bi-directional Data Buses

- In simple processors the data-in and data-out lines are never used at the same time.
- In these cases it is convenient to use just one bus as this reduces the size and complexity of the memory circuit.
- However, to make the data line bi-directional we need to feed it from more than one place and we cannot do this with ordinary AND or NAND gates.

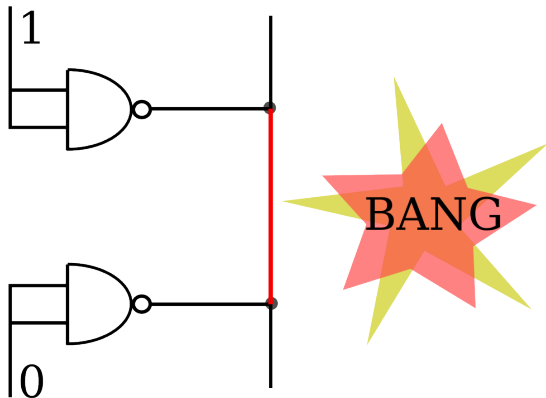
Bi-directional Data Buses

What would happen if we try to feed a bus from two different places using NAND gates?



Bi-directional Data Buses

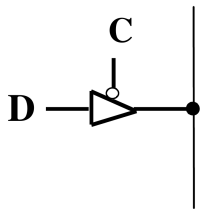
What would happen if we try to feed a bus from two different places using NAND gates?



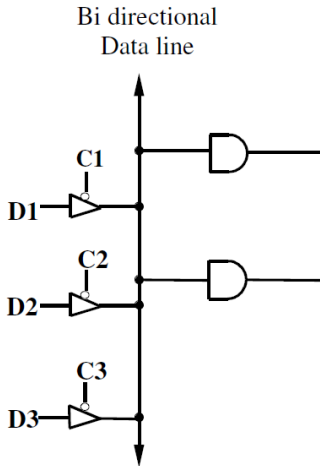
Bi-directional Data Buses

There is a special gate called a **Tri-State Buffer** which can overcome this problem:

- If the control line, C, is set to 0 the output follows the input exactly
- If C is set to 1 the output is neither zero nor one, but is effectively disconnected from the data line.



Bi-directional Data Buses



Many different data sources can be fed to a single line providing only one of them has $C=0$.

Another job for the very versatile demultiplexer!

Physical Layout of RAM

- Each one bit memory cell is only enabled when both its row and the column lines are one.
- In the case of a 256 bit RAM each decoder transforms a four bit binary number into a sixteen bit unary number.
- In the square array of one bit memory cells, there will only ever be one cell for which both the row and the column lines are one.

Physical Layout of RAM

- Each cell is connected to the same read/write line and data line.
- The data line is connected to the outside through a two way tri state buffer, so that unless the chip is enabled no data can pass either in or out.
- This enables us to build external decoders for larger capacity RAMs made up of several banks of single chips.

Problem Break

Given a 32 bit address bus:

1. How many bits of RAM could we address?
2. How many bytes of RAM could we address?
3. What size decoder do we need for the row and column decoders?

Problem Break - Answer

1. With a 32 bit address bus we have 2^{32} possible addresses which is in total 4GBytes.
2. Each bit of a byte will have the same address so again we can address 4GBytes.
3. The row and column decoders will both be 16 to 64K decoders. $(64K)^2 = 4 \text{ GBytes}$

Connecting RAM to a processor

To use RAM with a processor we need to use two new registers:

Memory Address Register (MAR)

Stores the address in memory to be stored or read.

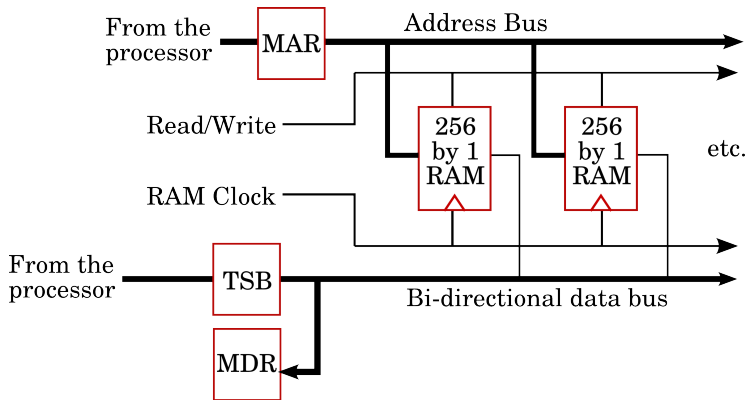
Memory Data Register (MDR) or Memory Buffer Register (MBR).

Stores data read from memory,
and sometimes data to be written to memory.

These registers are controlled by the processor.

Connecting RAM to a processor

The memory side of the connection:



TSB = Tri state buffer

Connecting RAM to a processor

The processor has some additional registers to manage the transfer of data to and from the RAM:

Program Counter (PC)

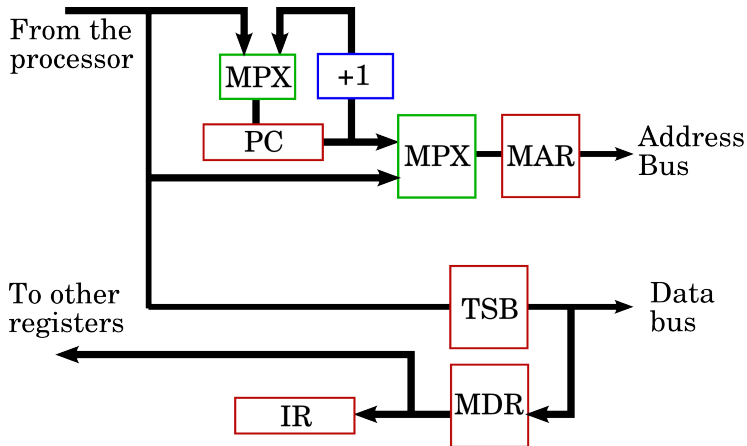
stores the address of the next program instruction to be executed.

Instruction Register (IR)

stores the program instruction being executed.

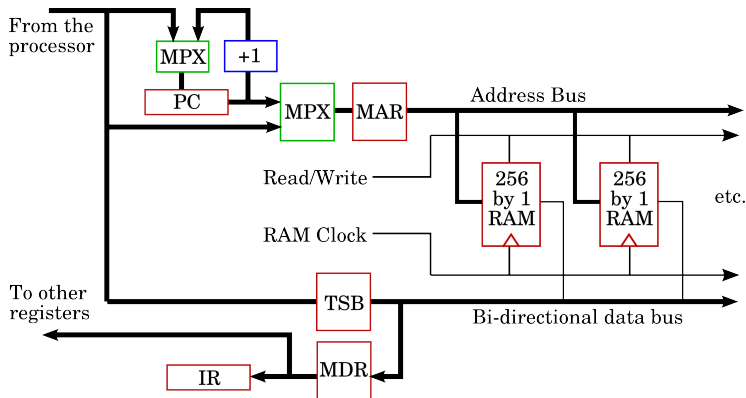
Connecting RAM to a processor

The processor side of the connection:



Connecting RAM to a processor

To complete the connection we need some control lines:



The Fetch Cycle

In order to retrieve data from memory we need to go through a number of steps. Each step transfers data from one register to another. The process is called the fetch cycle.

For example if we want to get the next program instruction and load it into the instruction register we need three steps:

$$\text{MAR} \leftarrow \text{PC}$$
$$\text{MDR} \leftarrow \text{RAM}[\text{MAR}], \text{PC} \leftarrow \text{PC}+1$$
$$\text{IR} \leftarrow \text{MDR}$$

Notice that in the second step we do two register transfers in parallel.

Register Transfers

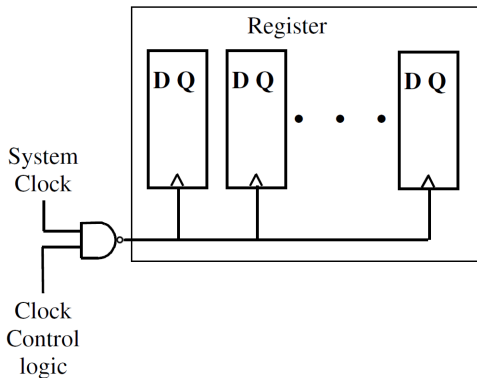
There are two things that need to be done in register transfer operations:

1. The multiplexers must be set to establish the required connection paths.
2. The registers which are to be loaded must be given a signal (falling edge) on their clock inputs.

These tasks are carried out by a controller which is a synchronous sequential circuit.

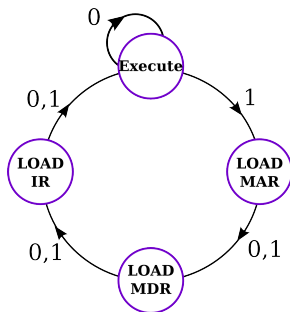
Clock Control Logic

Controlling when individual registers are loaded can be done simply by gating the system clock.



The Fetch Cycle

The controller for the fetch cycle is a synchronous sequential circuit.



Note that after fetching an instruction from memory the processor will “Execute” it. In practice this will require more than the one state shown here.

The Fetch Cycle

The output logic for the fetch cycle is summarized by the following table:

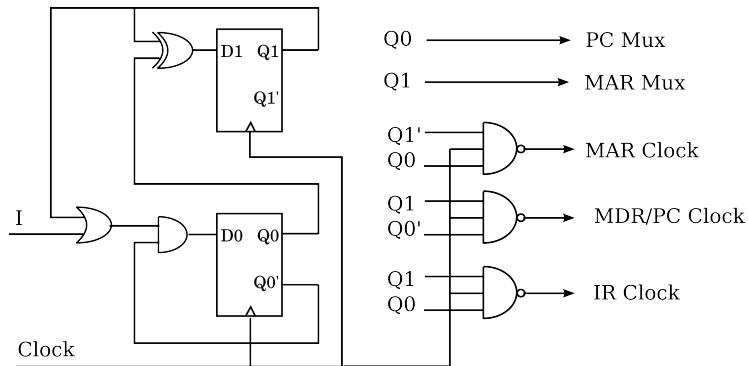
State	Clock Control				Multiplexer Control	
	MAR	MDR	IR	PC	PC Inp.	MAR Inp.
0. Executing	0	0	0	0	×	×
1. Load MAR	1	0	0	0	×	0
2. Load MDR/PC	0	1	0	1	0	×
3. Load IR	0	0	1	0	×	×

NB 0 sets the MAR input multiplexer to the PC output, and 0 sets the PC input multiplexer to the incrementer output.

Note the use of don't cares in selecting the register inputs.

The Fetch Cycle

The fetch cycle controller is:



Problem - Why are NAND gates used in the clock control lines in the output logic?

Dynamic RAM (DRAM)

- For large RAMs (eg $> 1\text{Mbit}$ chips) D-Q flip flops are not used since they are too big.
- Instead large RAMs use only one transistor and one capacitor for each bit.

Capacitor charged = 1

Capacitor uncharged = 0

Dynamic RAM (DRAM)

- The store is not permanent and all the cells storing ones drift to zero in a fraction of a second.
- A method is employed to restore the capacitor charge regularly.
- For this reason circuits of this kind are called dynamic RAM.
- Refreshing the capacitor's charges is done when the the computer is not accessing the memory.
- A memory controller must tell the DRAM to refresh itself periodically ... This overhead is very low.