Imperial College
London

Lecture 13

Memory Interface

Peter Cheung
Department of Electrical & Electronic Engineering
Imperial College London

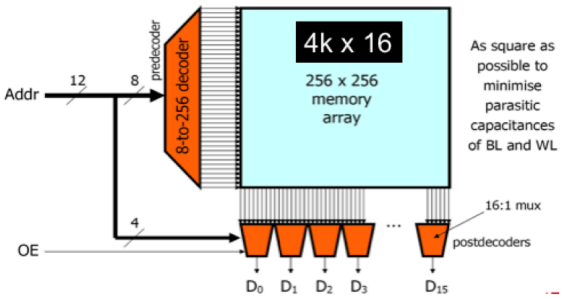
URL: www.ee.imperial.ac.uk/pcheung/teaching/ee2_digital/
E-mail: p.cheung@imperial.ac.uk

Lecture Objectives

- Explain how memory is connected to a microprocessor
- Describe the sequence of events in reading from and writing to a static RAM
- Describe the structure and input/output signals of a static RAM
- Investigate the timing constraints for a microprocessor when reading from or writing to memory

Memory interfacing is an essential topic for digital system design. In fact the among silicon area devoted to memory in a typical digital embedded system or a computer system is substantial. For example, in a mobile phone, the number of transistors devoted to memory is many times more than those used for computation. For the second year course, I will only focus on interfacing to static memory, known as RAM (Random Access Memory) or ROM (Read-Only Memory). There are other types of memory such as dynamic memory (DRAM), Synchronous DRAM (SDRAM) and flash memory (Flash RAM) which will not be covered on this course.

Simplified RAM Organization



A typical memory has:

- N-bit address bus
  - This defines 2^N memory locations
- M-bit data bus
  - This defines the size of data word
- Control signals to define read/write cycles,
  - Asynchronous memory – no clock in control
  - Synchronous memory – every action synchronised to a clock

A typical 8-bit microprocessor typically has

- A 16-bit address bus, A15:0
  - Can have up to 2^{16}=65536 memory locations
- An 8-bit data bus, D7:0 - Each data word in memory has 2^8 = 256 possible values
- In the RAM shown above uses 12-bit address and 16-bit data, i.e. 4096 locations of 16-bits each
- These are arranged as 256 x 256 rows of memory cells. 4096 = 256 rows x 16 columns as shown
- The address bus is therefore split into two components: 8-bit to specify which row, and 4-bit to select the correct column.

This slide shows a typical organisation inside a RAM chip. Memory cells are usually organised in the form of a 2-D array of RAM cells. These are accessed first in a row, then in a column. The address bus is divided into two components, the row address (8-bit in the example here) and the column address (4-bit in this example). There is a decode to translate the 8-bit row address into one-hot outputs in order to specify which row is being accessed. Only ONE ROW will be enable at any one time (hence one-hot).

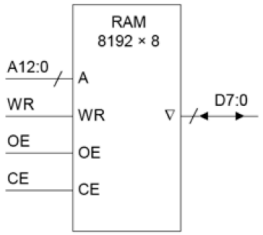
The second part of the address (normally the less significant bits) is used as select signal into the output mux. This is because when memory is accessed, they are normally read or written in a sequence. Using LSB for column decoding means that one stays on the same row of memory as much as possible. Staying in the same row uses significantly lower energy than switching between rows in memory accesses.

In the example here, the 4-bit column address is used to select from a 16-to-1 mux to provide the correct location in memory to access. There are 16 identical blocks, each providing one-bit of the data output.

The output enable signal OE allows the selected data value be driven on the data bus.

RAM: Read/Write Memory

8k × 8 Static RAM



CE	OE	WR	D0:7	Action
0	?	?	Hi Z	Disabled
1	0	0	Hi Z	Idle
1	1	0	Out	Read
1	?	1	In	Write

Hi Z = High impedance

Static RAM:	Data stored in bistable latches
Dynamic RAM:	Data stored in charged capacitors: retained for only 2ms. Less circuitry ⇒ denser ⇒ cheaper.

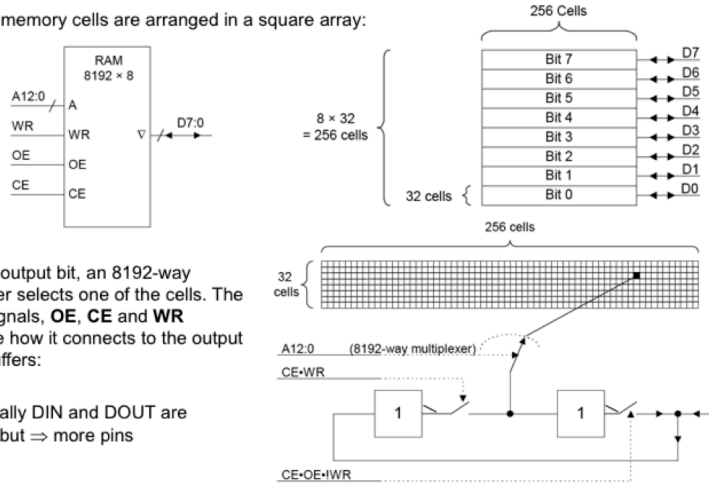
∇	Tri-state output: Low, High or Off (High Impedance). Allows outputs from several chips to be connected; Designer must ensure only one is enabled at a time.
CE	Chip Enable: disabling chip cuts power by 80%.
OE	Output Enable: Turns the tri-state outputs on/off.
A12:0	Address: selects one of the 2^{13} 8-bit locations.
WR	Write: stores new data in selected location
D7:0	Data in for write cycles or out for read cycles.

Here is a 8K x 8 static RAM chip and its associated digital signals. The 13-bit address bus A12:0, the 8-bit data bus D7:0 are mandatory. There are three more control signals: Output Enable OE which we have seen before, Chip Enable CE which is used to address or select this particular memory chip (hence the name), and finally the WRITE ENABLE signal WE, which, when set high, indicates that you are writing to the RAM chip, and is normally low (i.e. reading).

Note that the data bus has an inverted triangle sign, indicating that this is a **tri-state bus**. This means that the pin could be an input pin, output pin, or an open-circuit pin (i.e. not connected to anything – we call the signal *floating*). The truth table shown here specifies the behaviour of the data bus in one of the three possible states.

8k x 8 Static RAM

- The 64k memory cells are arranged in a square array:



- For each output bit, an 8192-way multiplexer selects one of the cells. The control signals, **OE**, **CE** and **WR** determine how it connects to the output pin via buffers:
- Occasionally DIN and DOUT are separate but ⇒ more pins

For a 8k x 8 RAM, there are 8 data bits, and therefore 8 separate 1-bit arrays. Let us assume that each data bit array is organised as a 256 rows x 32 column (=8192) of memory cells. Eight such array are placed next to each other to form the 8 data bits required. This makes the memory chip roughly square (which is generally desirable).

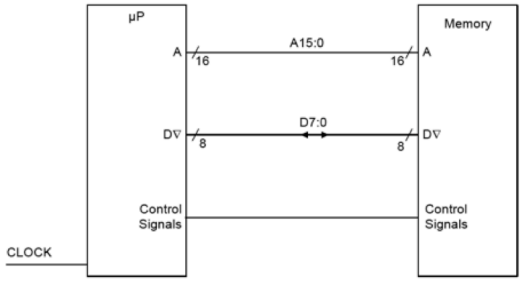
You can think of the row decoder and the column selector driven by the 13-bit address as a 8192 way multiplexer, selecting one of 8192 cells organised as 256 x 32, to be accessed.

The simplified circuit of each memory cell shown here consists of two inverters and two switches is a schematic of the read-write circuit. When reading from the cell, A12:0 select one of 8192 cells to route its signal via the right inverter to Dn. Now Dn is an output pin. This only happens if CE*OE* !WR = 1 (i.e. asserting CE and OE, but not asserting WR).

When writing to the memory cell, the right switch is open, Dn is an input pin driving the left hand inverter and the output switch from that inverter is closed because both CE and WR are asserted.

Some memory chips have separate Din and Dout pins, but that's expensive on pins and is not particularly common nowadays.

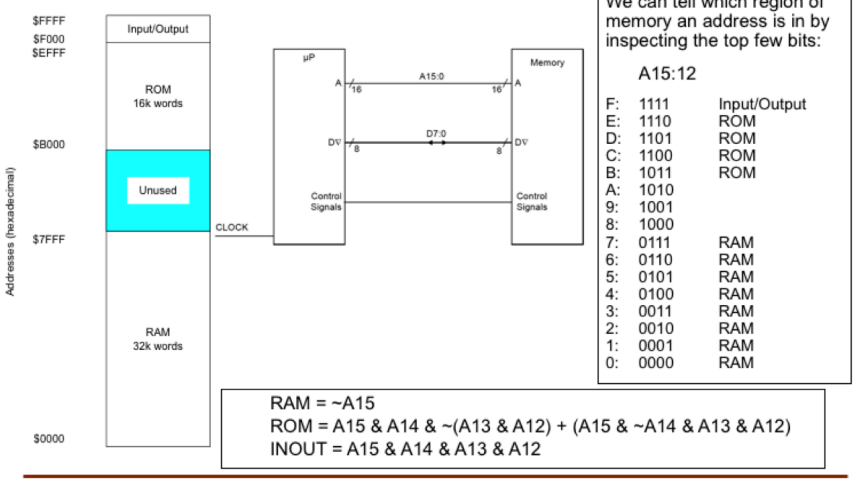
Microprocessor ⟷ Memory Interface



- During each memory cycle:
- A15:0 selects one of 2^{16} possible memory locations
- D7:0 transfer one word (8 bits) of information either to the memory (write) or to the microprocessor (read).
- D7:0 connections to the microprocessor are tri-state (∇): they can be:
  - "logic 0", "logic 1" or "high impedance" (inputs)
- The control signals tell the memory what to do and when to do it.

Here is a slide showing a generic interfacing between a microprocessor and a memory sub-system. We assume that we use a 16-bit address bus and an 8-bit data bus. The control signals go between the two to control the transfer of information, and is in general governed by the microprocessor which acts as the "**master**".

Microprocessor Memory Map



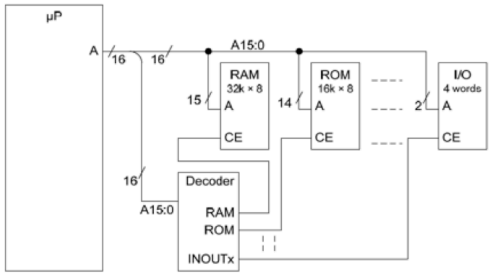
While we show memory as a block, in a real system, the memory address space is divided into many different partitions. Here we use '$' (instead of 16'hxxxx) to indicate that the addresses are hexadecimal numbers. The left hand diagram shows the memory being partitioned into 32k of RAM, 16k of ROM and 4k space for input/output devices.

A design needs to take **the upper bits of the address bus** and decode these bits into **enable signals** for the three different partitions. In this case, we can see that we only need to decode A15:12 according to the Boolean equations shown here. What about A11:0? These are the address bits used inside the RAM, ROM and input/output modules to select particular locations.

Memory Chip Selection

- Each memory circuit has a "chip enable" input (CE)
- The "Decoder" uses the top few address bits to decide which memory circuit should be enabled. Each one is enabled only for the correct address range:
  - RAM = ~A15
  - ROM = A15 & A14 & ~(A13 & A12) | (A15 & ~A14 & A13 & A12)
  - INOUTx = **A15 & A14 & A13 & A12** & ~A11 & A10 & ~A9 & A8 & ~A7 & A6 & A5 & A4 &~A3 & A2
- INOUTx responds to addresses: $F574 to $F577
  other I/O circuits will have different addresses
- Low *n* address bits select one of 2^n locations within each memory circuit (value of n depends on memory size)

Addr Range	Usage
$F578 - $FFFF	Not used
$F574 - $F577	**INOUTx**
$F000 - $F573	Not used
$B000 - $EFFF	16k ROM
$8000 - $AFFF	Not used
$0000 - $7FFF	32k RAM



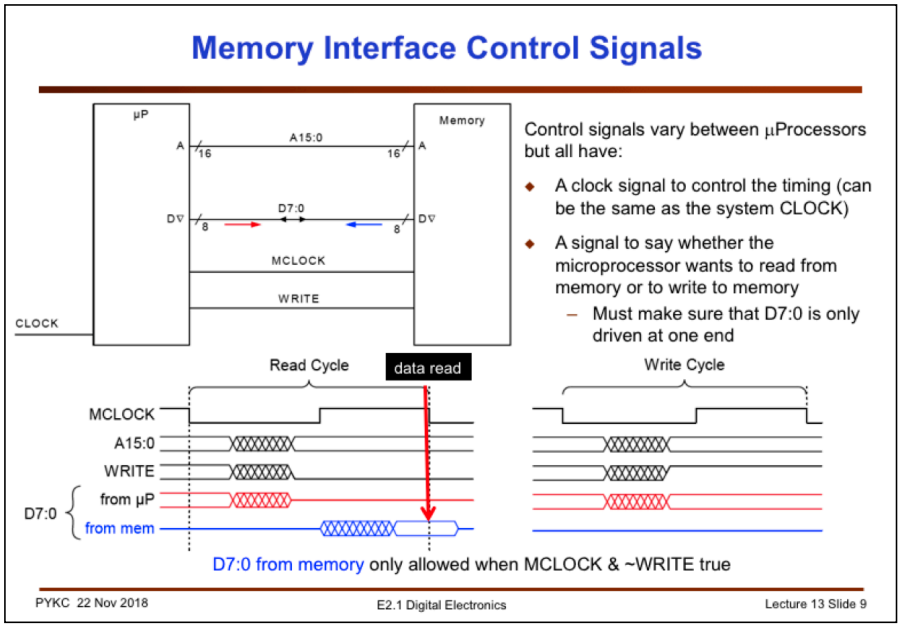
Selecting which memory sub-system and therefore which memory chip to enable is the job of the address decoder circuit. This circuit takes the upper bits of the address bus, and produce enable signals for RAM, ROM and INOUTx for a particular I/O device.

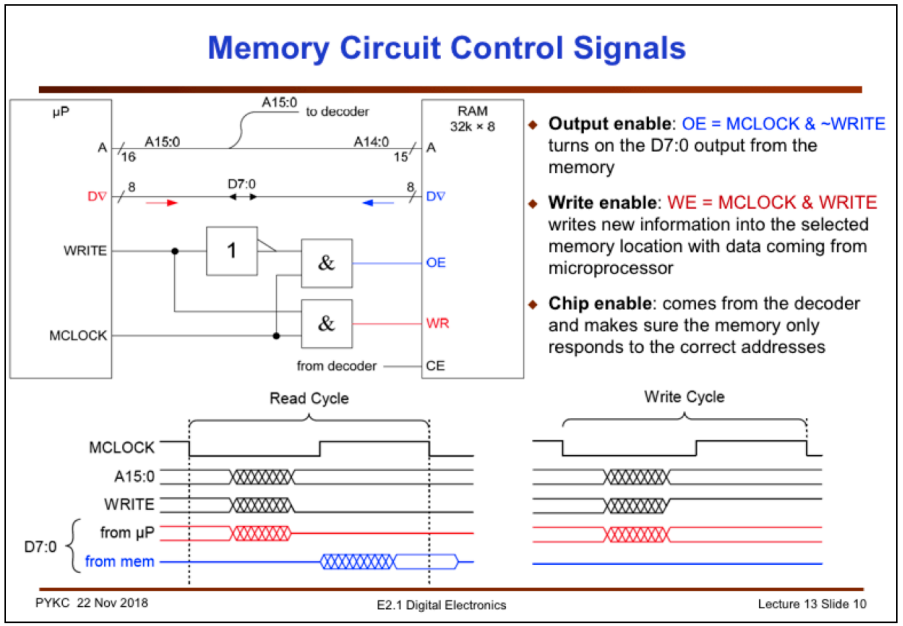
In the previous slide, we showed that the input/output occupies 4k of memory space. This is uncommon. Typically an I/O device may take up, say, 4 memory locations.

In this example, INOUTx occupies only the address space $F574 - $F577, i.e. 4 locations. Therefore we need to decode lots of address signals: A15:2.

Can you work out the Boolean equations for the address decoder shown here?

The ROM CE signal is another challenge. The ROM is enable if the address A15:A12 falls between the range 4'b1011 and 4'b1110. You should prove for yourself that the Boolean equation to decode the address for the ROM is as shown here.





In addition to the address decoder circuit, we need to provide the control signals from the microprocessor to the memory chips. Here we assume there exists at least two control signals from the microprocessor: MCLOCK which is memory clock signal (which may be different from the system clock signal CLOCK), and a WRITE signal, which is high when writing to memory, but low otherwise.

The interaction between the microprocessor and memory can be separated into two types of transactions: a **Read Cycle** and a **Write Cycle**.

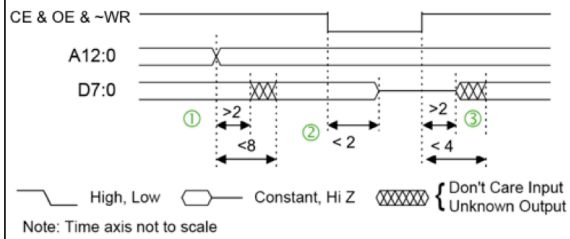
During Read Cycle, the microprocessor asserts the address A15:0 and the control signals MCLOCK and WRITE. Shortly after the beginning of the Read Cycle, the microprocessor must STOP driving the data bus D7:0, and on the second half of the cycle, we assume that memory will then provide the data for the microprocessor to read. Reading is actually performed at the end of the Read Cycle, on the falling edge of MCLOCK. Note that I use red colour to indicate the action of the microprocessor on the data bus, and blue colour for the action by the memory chip on the data bus.

During a Write Cycle, the microprocessor drives everything. Writing also occurs on the falling edge of MCLOCK in our case. (Note that other system may have a different protocol than the one shown here.)

This slide shows the control circuit used to interface the microprocessor to the 32k x 8 RAM chip.

Chip Enable (CE) is driven by the output from the address decoder, which we have considered in an earlier slide. Remember the colour code I am using: RED driven by the microprocessor, BLUE driven by memory.

Memory Read Cycle



Note: Time axis not to scale

A *read cycle* happens when CE&OE&~WR is true.

① If A12:0 changes, D7:0 remains for at least 2 ns and goes to new value within 8 ns. Rubbish in between even if new and old locations contain the same value.

② If a read cycle ends due to OE going low, the outputs go Hi-Z within 2 ns

③ If a read cycle starts due to OE going high, D7:0 stays Hi-Z for at least another 2 ns and the selected word appears within 4ns

- This diagram shows the timing constraints on memory during a read cycle.
- All timing shown here are only indicative. Another memory device would have totally different timing specifications.
- You can use CE instead of OE but it is slower: 10 ns to turn off and 15 ns to turn on (in parentheses on timing diagram).
- When reading data, the propagation delay to the D7:0 outputs is called the RAM's *access time*: 8 ns from A12:0 and 4 ns from OE.

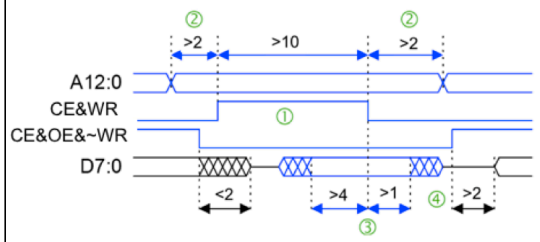
Let us now consider the timing constraints imposed by memory during a Read Cycle. First thing that happens would be a valid address A12:0 being presented at (1). As a typical example for memory timing, it is assume that data D7:0 holds for at least 2ns before changing, but it is guaranteed to provide the correct D7:0 at the new address in 8ns or earlier. This is address to data ACCESS TIME for this RAM. Note that even if new and old location have the same data value, there will be period when D7:0 contain rubbish – beware. Also note that memory is providing data to be read by the microprocessor, CE, OE and ~WR must all be asserted (i.e. '1').

At (2), memory is deselected or output not enabled, or we are no longer reading from memory. D7:0 again is guaranteed to go high-impedance after 2ns.

Some time later, if member is selected again at (3), it takes 2ns before memory start to drive D7:0, but guaranteed to provide correct data after 4ns.

The most important delay here is that from address or OE to data. They are called address access time and output enable access time. Usually address access timing is longer (here it is 8ns) than OE access time (4ns) because output enable simply enable the output multiplex stage, which is close to the data output pin. Address access involves decoding the address values to produce the one-hot row select signal (known as the WORD line), and then the row of memory cells needs to present its data to the column multiplexer. Selecting which row to access is generally a much slower process than the column multiplexer.

Memory Write Cycle



A *write cycle* happens whenever CE&WR is true.

① CE•WR must go high for at least 10 ns.

② To avoid writing to the wrong locations, the address, A12:0 must remain constant for at least 2 ns at both ends of the write pulse.

③ Input data D0:7 only matters at the *end* of the write pulse. Setup & hold times of 4 ns & 1 ns define a window within which data must not change.

④ When CE&OE&~WR goes high, the memory reverts to read mode. The input data must be removed from D7:0 before this happens.

- Timing specifications that end on an output are guarantees from the chip manufacturer (shown in **black**).
- Timing specifications that end on an input are requirements that the designer must meet (shown **blue**).

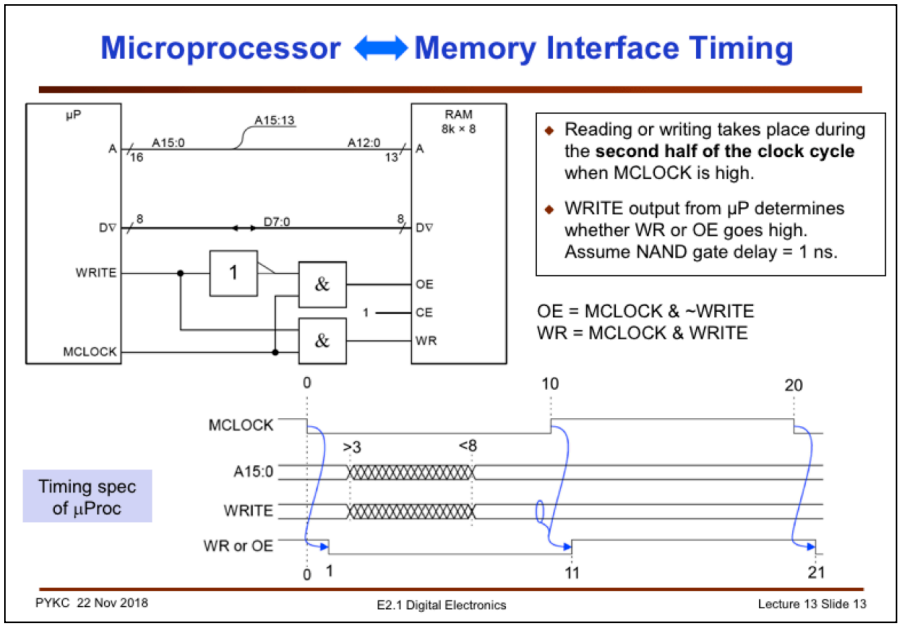
Here is the timing for the **Write Cycle**. Remember that Write Cycle timing is particularly important – any timing error here could result in corrupting the contents store in RAM.

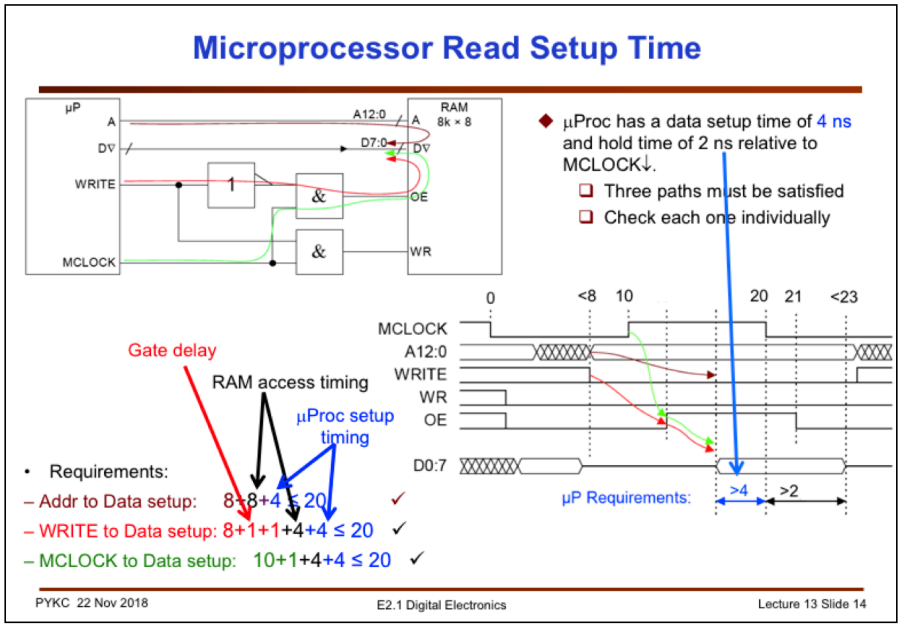
(1)The write pulse is signified by CE and WR both being asserted (i.e. TRUE). There is usually a minimum period specified – here 10ns. Also as soon as the WR is asserted, WR = 1 and D7:0 must go high-impedance within 2ns (i.e. memory no longer driving the data bus).

(2)The address A12:0 must be stable at least 2ns before the write pulse, and it must hold for another 2ns after the write pulse.

(3)The data is written to memory on the falling edge of the write pulse. The setup and hold time is 4ns and 1ns respectively.

(4)This is when the Write Cycle finishes, and we go back to Read Cycle. Expect D7:0 stays high impedance for at least 2ns.





The previous two slides only look at the timing constraints from the perspective of memory. Here we must consider what the microprocessor expects about timing.

We assume that falling edge of MCLOCK starts a memory transaction cycle. Gates are assumed to have a 1ns delay. Therefore WR and OE signals are delayed by 1ns. A15:0 is assumed to hold its previous value for at least 3ns, but will go to the new vale by 8ns. The same is true with the WRITE signal.

Let us consider the timing constraints when the microprocessor is reading from RAM, using all the timing specifications we have defined in previous slides. Here the microprocessor is assumed to have a data setup time of 4ns, and hold time of 2ns relative to the following edge of MCLOCK (when reading occurs).

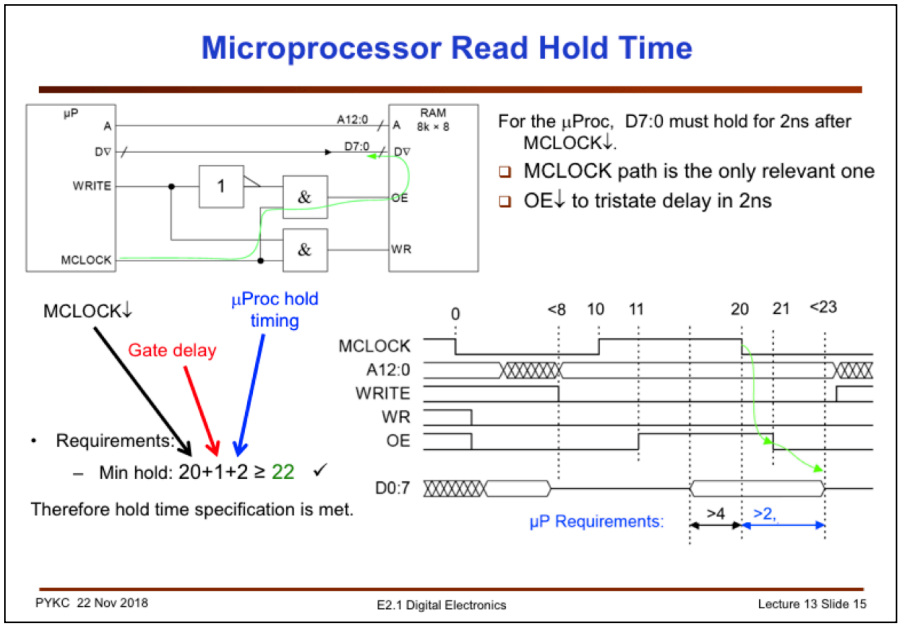
The three timing inequality are:

(1)Address to Data setup time – RAM must provide the data to the micro early enough for the micro to read it properly. The setup time of the data input to micro is 4ns, the address access time of the RAM is 8ns, and the address is only stable 8ns after the falling edge of MCLOCK. Therefore we obtain the left side of the inequality.

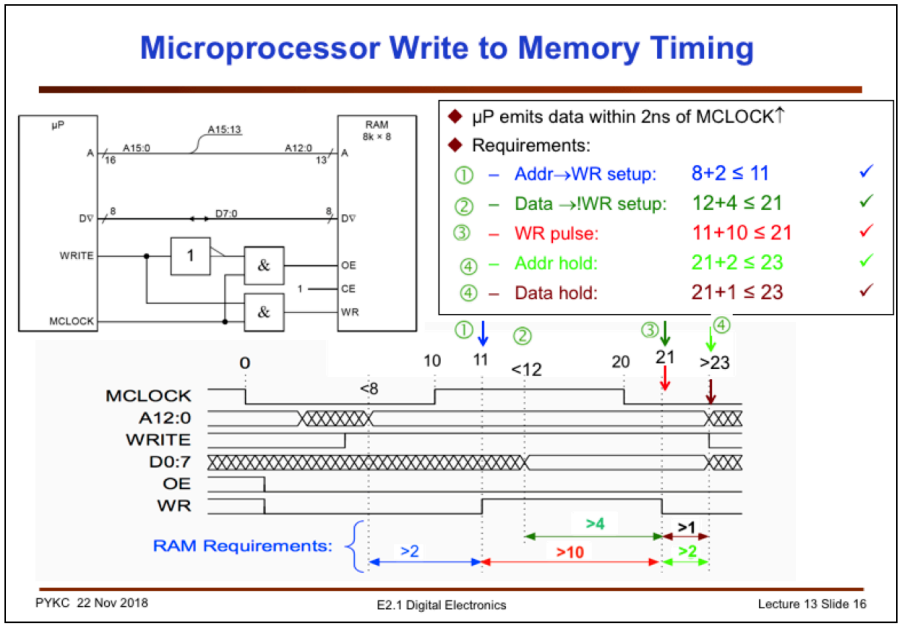
(2)WRITE to Data setup time – We need to check the WRITE signal to data setup. The WRITE signal goes low (indicating a read cycle) at 8ns. The worst case path is to OE is through two gates, adding another 2ns delay. RAM access time from OE is 4ns and the data setup time is 4ns. The total of this must be less than or equal to 20, which it is.

(3)Finally, the MCLOCK to Data setup time – MCLOCK also control OE and WR, so we need to check that it does not violate any timing requirement. MCLOCK goes high at 10ns, and there is a 1ns delay with the AND gate. Adding 4ns OE setup time and 4ns micro data setup time, the total must be earlier than 20ns, which it is.

Therefore all the timing constraints for the Read setup time are satisfied.



Let us now consider the hold time. This is relatively simple as shown here.



Let us now combine the RAM timing with the microprocessor timing. There are 4 separate timing specifications to consider.

(1)Address to WR setup time – this timing constraint is imposed by RAM that address must be stable (set up time) 2ns before WR pulse goes high. Since the WR pulse goes high at 11ns (relative to falling edge of MCLOCK which is the reference 0ns), and that the address A12:0 are stable no later than 8ns, the constraint is satisfied.

(2)Data to !WR setup time – writing to memory occurs on the falling edge of WR. Data must be stable 4ns before that. WR goes low at 21ns. Data is stable at 12ns or earlier. Therefore this constraint is also satisfied.

(3)WR pulse width – this must be at least 10ns. Since the write pulse starts at 11ns and finishes at 21ns, the minimum WR pulse width is just met (with no margin).

(4)Address hold time – The address must be held 2ns after WR signal is deasserted. Therefore this hold time is also just met (with no margin).

(5)Data hold time – Data is held for 1ns. Therefore data hold time is also met.

As can be seen, checking all timing constraints being met is quite a tedious process. Fortunately the timing analyzer tools perform all the setup and hold time checking for us in Quartus and report any violations.

Quiz Questions

1. What is the *memory map* of a microprocessor system
2. Why do all microprocessor systems include some read-only memory (ROM)
3. What does it mean if a digital device has a *tri-state* output? When are such outputs necessary ?
4. What is the difference between the *chip enable* and the *output enable* inputs of a static RAM?
5. If a static RAM has *n* address inputs and *m* data outputs, how many bits of information does it store?
6. What is the binary value of the three most significant address bits for the hexadecimal address $BC37 ?
7. What is the *access time* of a static RAM?
8. When writing to a static RAM, why is does the state of the data inputs matter only at the end of the write pulse?
9. How do you check timing constraints if the manufacturer specifies a maximum propagation delay but no minimum ?
10. How do you check timing constraints if the validity of an output depends on several of the input signals ?

Answers are all in the notes.