Imperial College London

Lecture 13

Memories

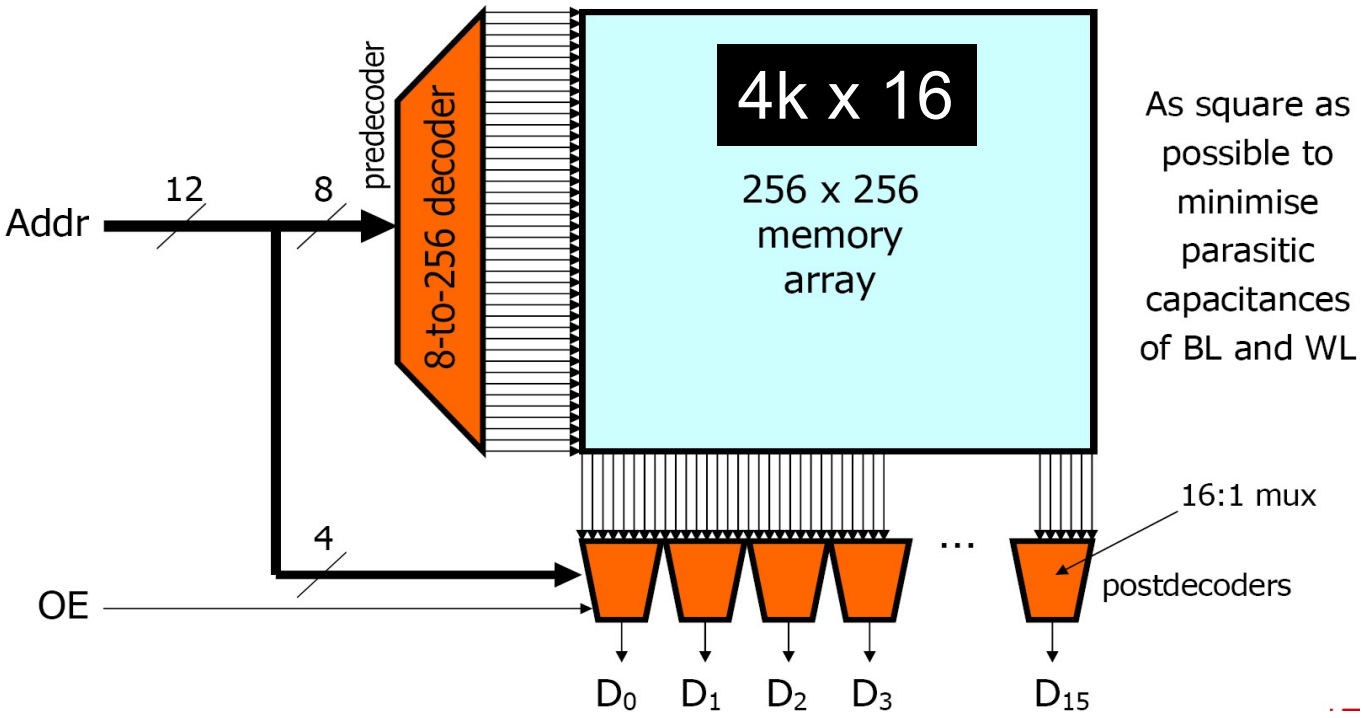
Peter Cheung
Imperial College London

URL: www.ee.imperial.ac.uk/pcheung/teaching/E2_CAS/
E-mail: p.cheung@imperial.ac.uk

Lecture Objectives

- Explain the sequence of events in reading from and writing to a static RAM
- Explain the structure and input/output signals of a static RAM
- How to design an address decoder
- Investigate the timing diagrams for a microprocessor when reading from or writing to memory
- Explain how the embedded memory in an FPGA can be used to implement memory blocks in a digital design

Simplified RAM Organization



A typical memory has:

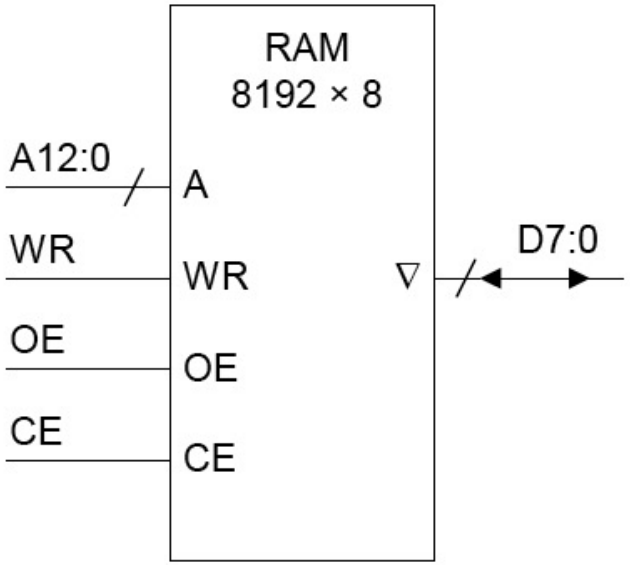
- N-bit address bus
  - This defines 2^N memory locations
- M-bit data bus
  - This defines the size of data word
- Control signals to define read/write cycles,
  - Asynchronous memory – no clock in control
  - Synchronous memory – every action synchronised to a clock

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.

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

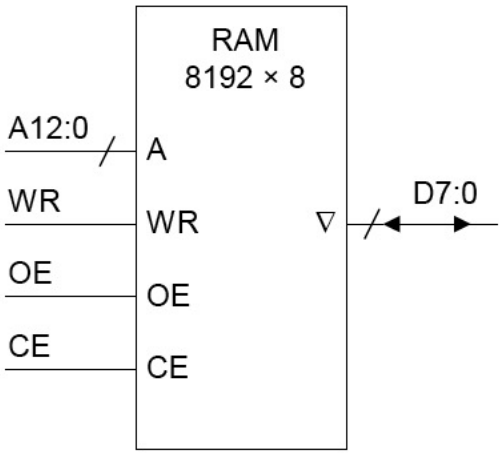
Hi Z = High impedance

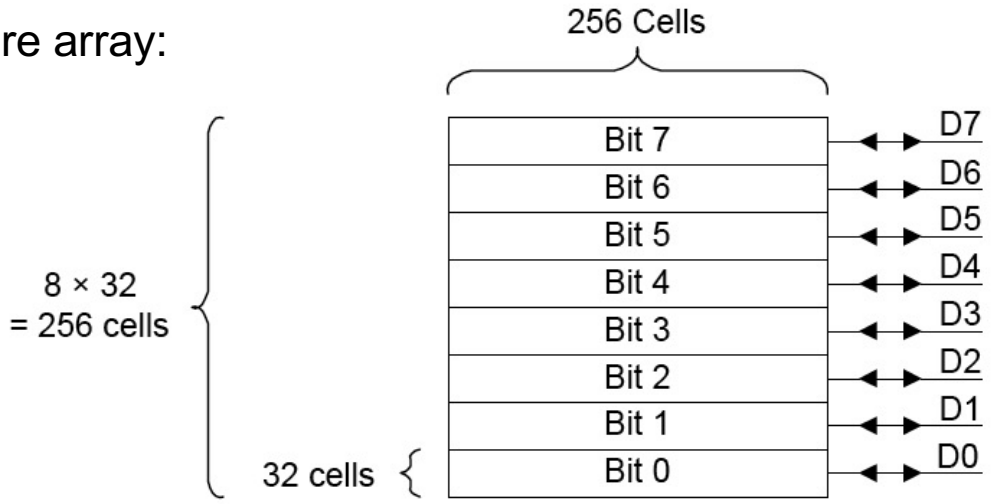
Static RAM:	Data stored in bistable latches
Dynamic RAM:	Data stored in charged capacitors: retained for only 2ms. Less circuitry ⇒ denser ⇒ cheaper.

∇	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.

8k x 8 Static RAM

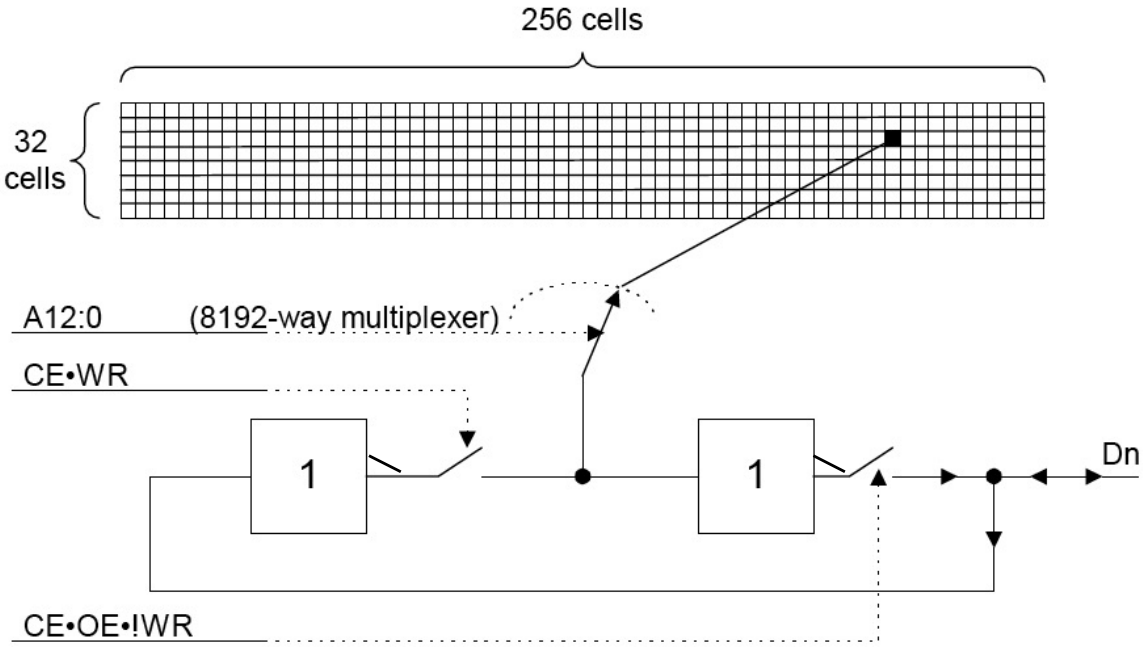
- The 64k memory cells are arranged in a square array:



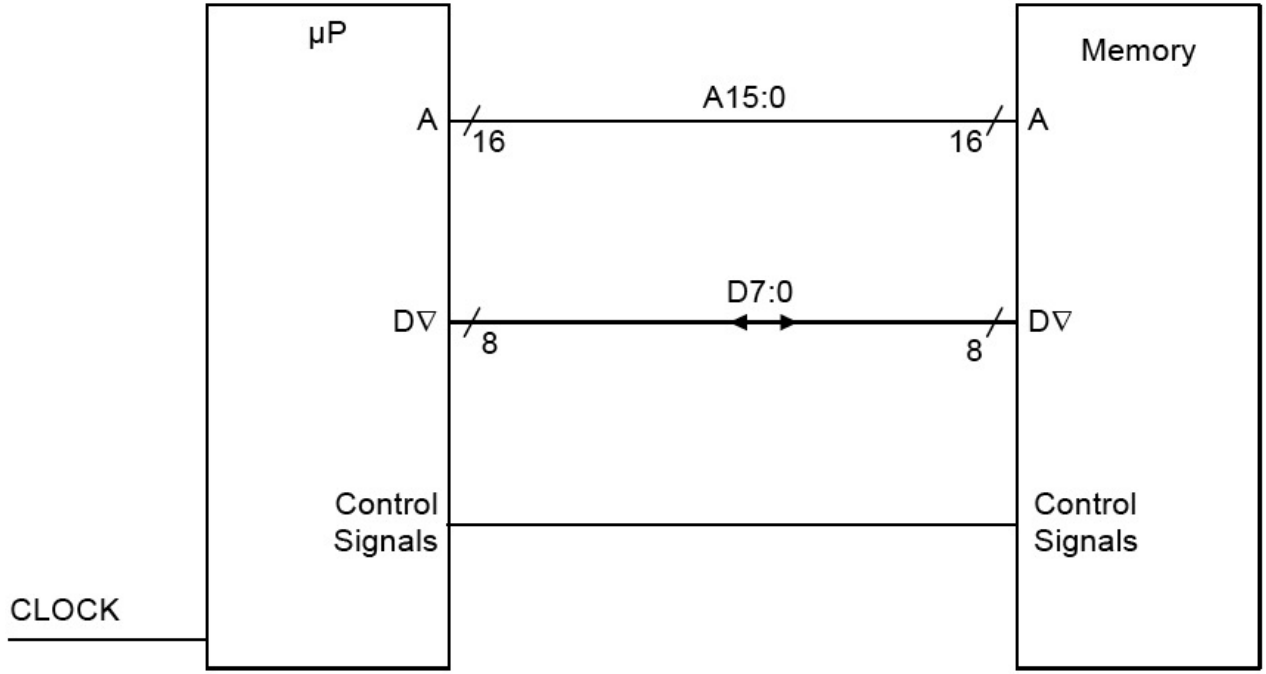


- For each output bit, an 8192-way multiplexer selects one of the cells. The control signals, **OE**, **CE** and **WR** determine how it connects to the output pin via buffers:

- Occasionally DIN and DOUT are separate but ⇒ more pins

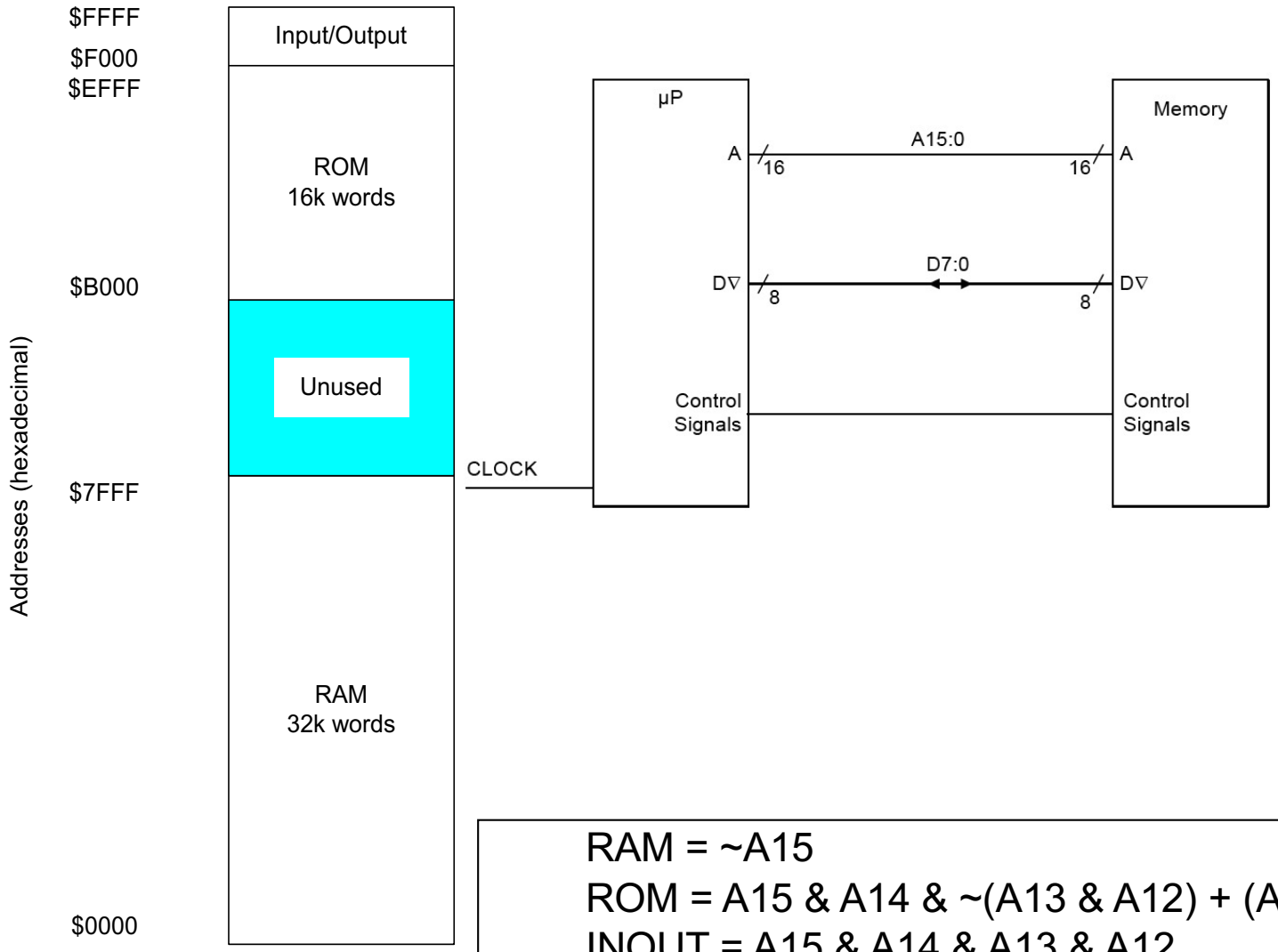


Microprocessor ⟺ Memory Interface



- 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.

Microprocessor Memory Map



We can tell which region of memory an address is in by inspecting the top few bits:

A15:12

F:	1111	Input/Output
E:	1110	ROM
D:	1101	ROM
C:	1100	ROM
B:	1011	ROM
A:	1010	
9:	1001	
8:	1000	
7:	0111	RAM
6:	0110	RAM
5:	0101	RAM
4:	0100	RAM
3:	0011	RAM
2:	0010	RAM
1:	0001	RAM
0:	0000	RAM

RAM = ~A15

ROM = A15 & A14 & ~(A13 & A12) + (A15 & ~A14 & A13 & A12)

INOUT = A15 & A14 & A13 & A12

Memory Chip Selection

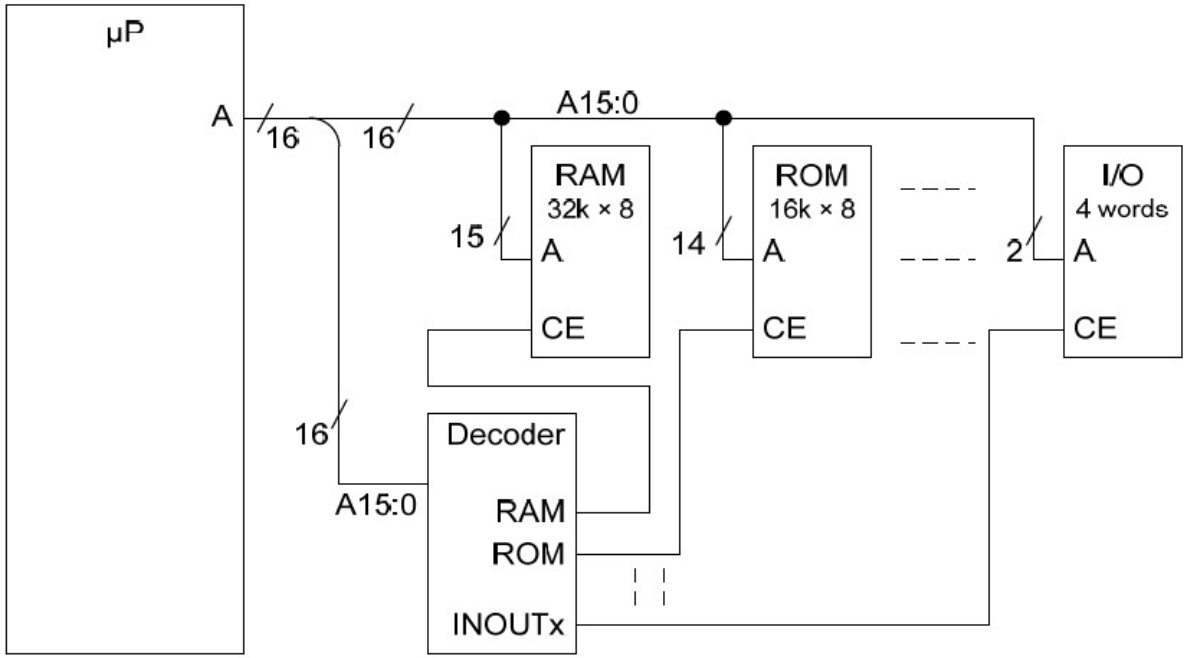
- Each memory circuit has a "chip enable" input (CE)
- The "Decoder" uses the top few address bits to decide which memory circuit should be enabled. Each one is enabled only for the correct address range:

  RAM = ~A15

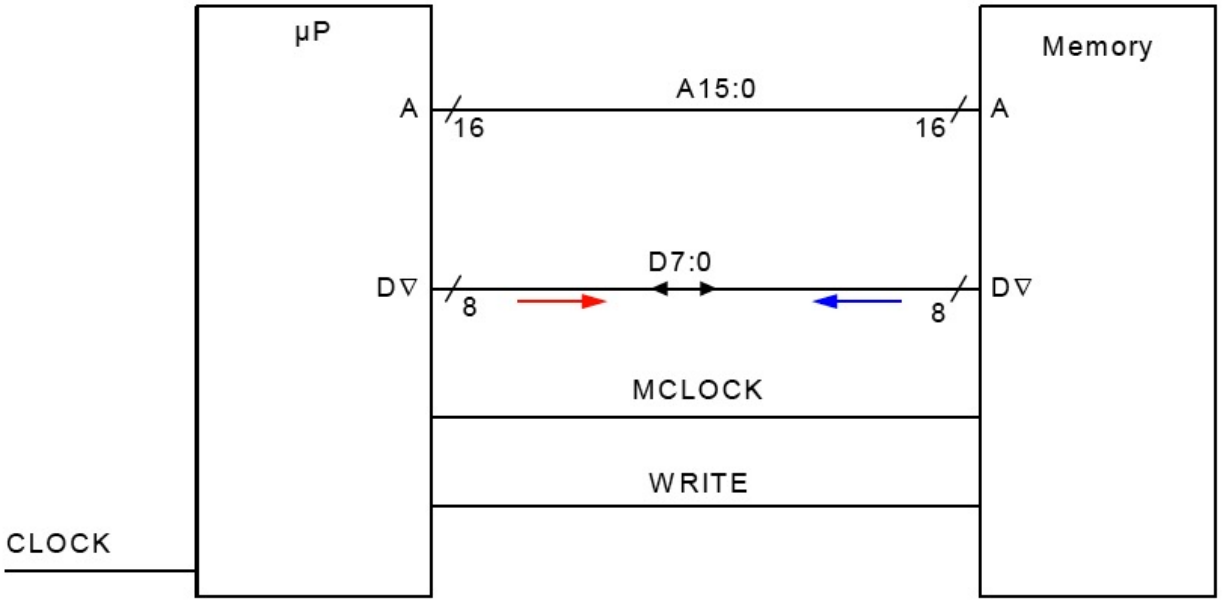
  ROM = A15 & A14 & ~(A13 & A12) + (A15 & ~A14 & A13 & A12)

  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

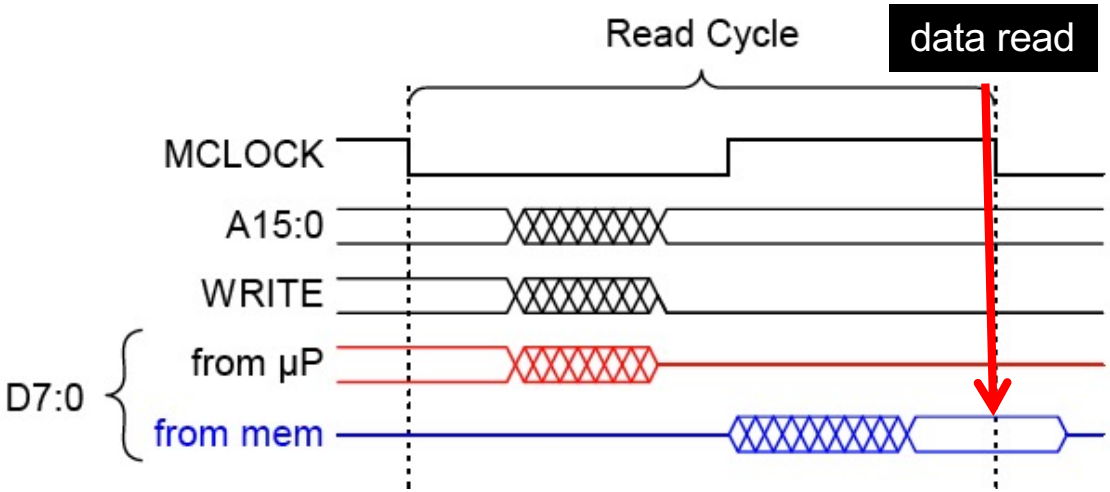


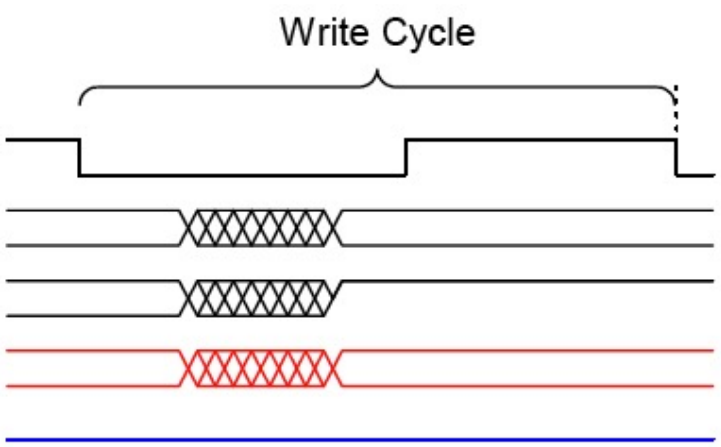
Memory Interface Control Signals



Control signals vary between µProcessors but all have:

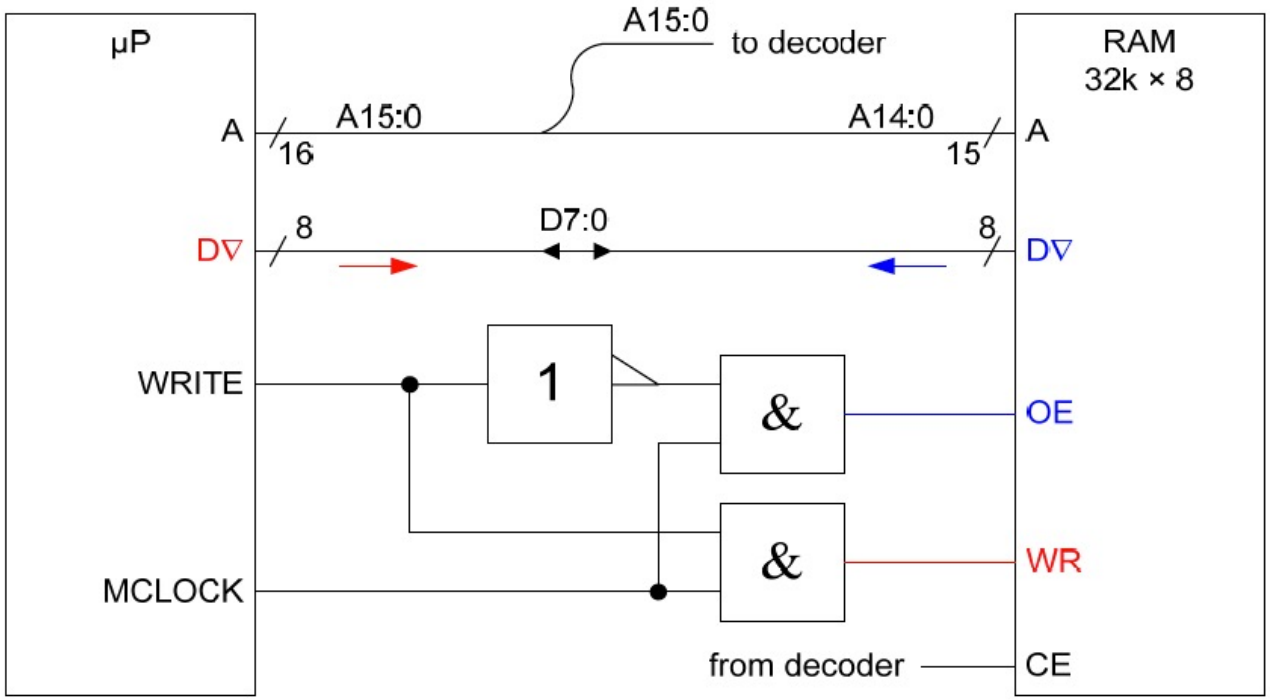
- A clock signal to control the timing (can be the same as the system CLOCK)
- A signal to say whether the microprocessor wants to read from memory or to write to memory
  - Must make sure that D7:0 is only driven at one end



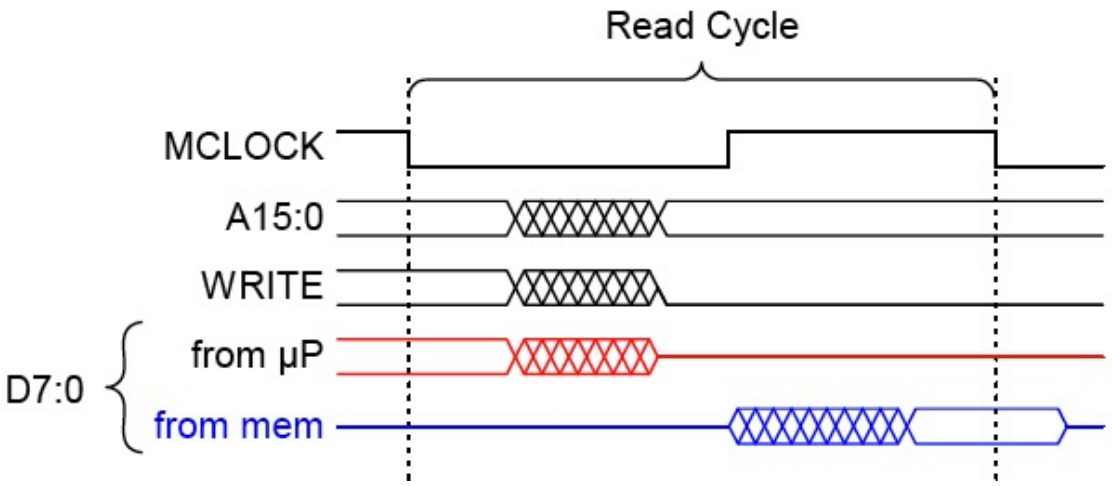


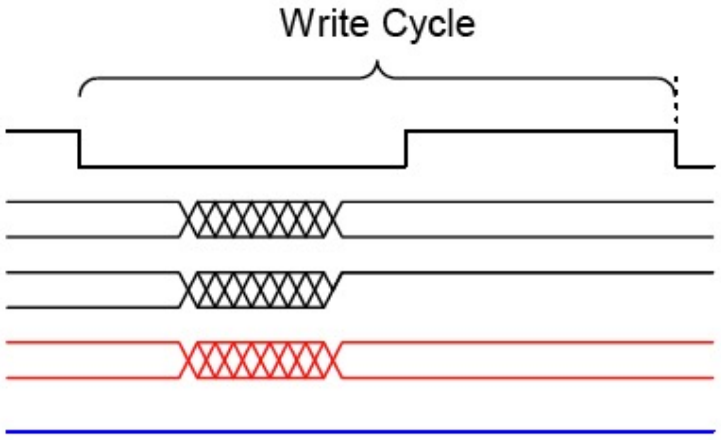
D7:0 from memory only allowed when MCLOCK & ~WRITE true

Memory Circuit Control Signals

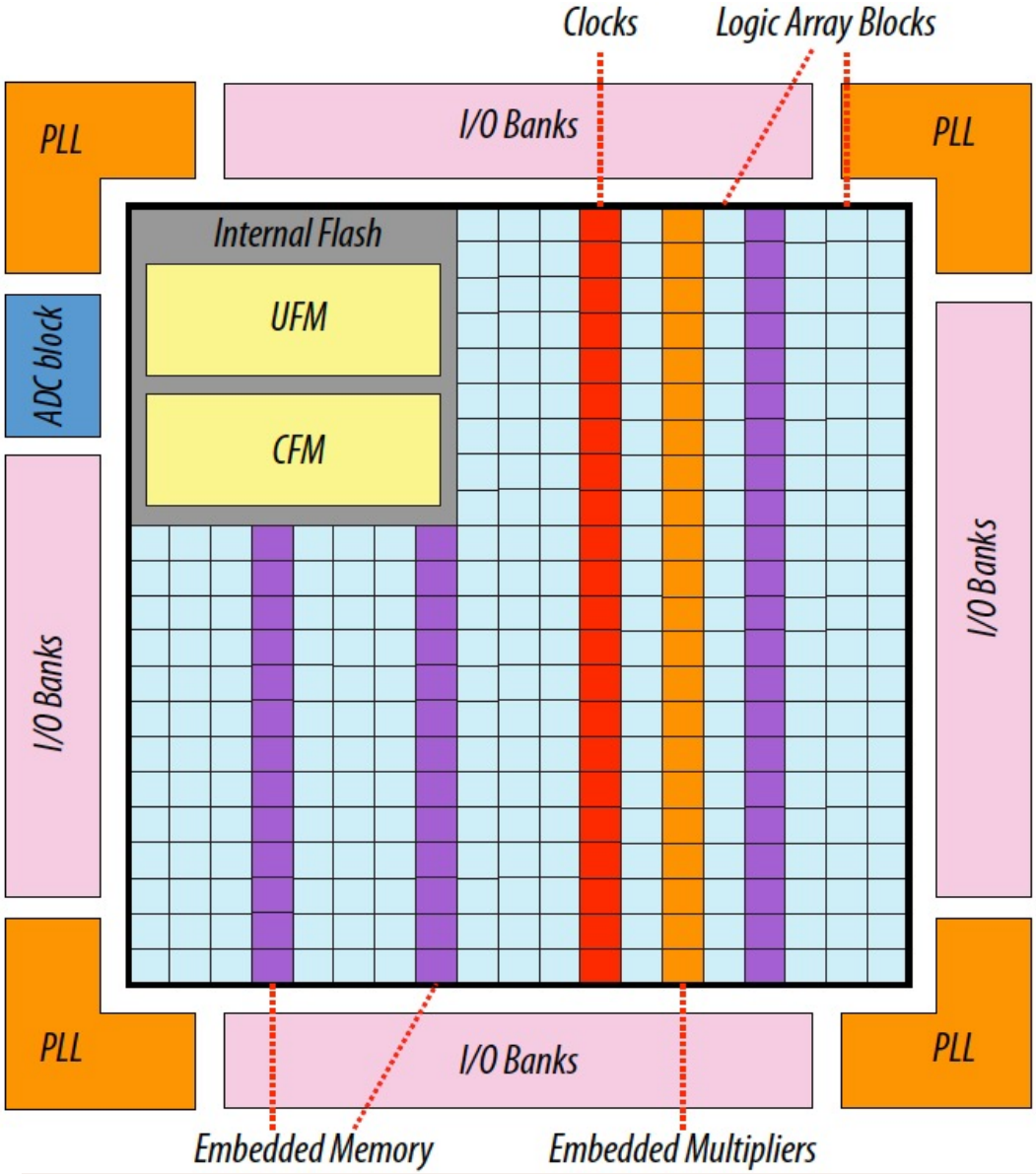


- **Output enable**: OE = MCLOCK & ~WRITE turns on the D7:0 output from the memory
- **Write enable**: WE = MCLOCK & WRITE writes new information into the selected memory location with data coming from microprocessor
- **Chip enable**: comes from the decoder and makes sure the memory only responds to the correct addresses





Max 10 FPGA – Embedded blocks



- MAX10 device: 10M50DAF484C7G
- 50,000 Logic Elements (4-LUT + FF)
- 182 M9K embedded memory blocks
- 5,888 kbits user flash memory
- 144 hard multiplier (18 x 18)
- 4 PLL (for clock generation

MAX 10 Embedded Memory

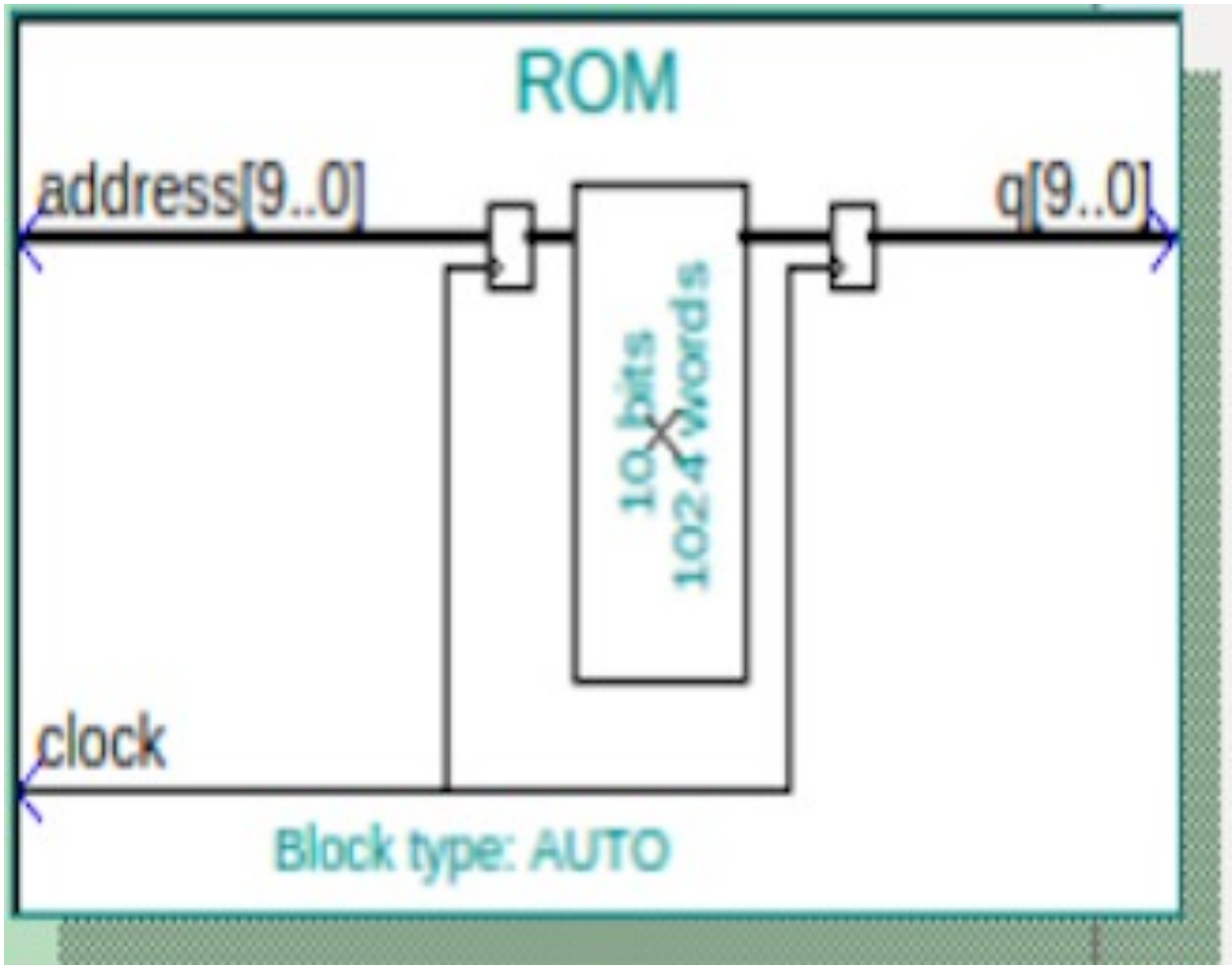
- Each 9kbit memory block (M9K) can be configured with different data width from 1 bit to 36 bit wide
- It also has multiple operating modes (which is user configurable), of which we will focus on the following only: 1-port ROM, FIFO, 2-port RAM

- Single-port
- Simple dual-port
- True dual-port
- Shift-register
- ROM
- FIFO

Configuration	M9K Block
Depth × width	8192 × 1
	4096 × 2
	2048 × 4
	1024 × 8
	1024 × 9
	512 × 16
	512 × 18
	256 × 32
	256 × 36

Intialization of ROM Contents (1k x 8)

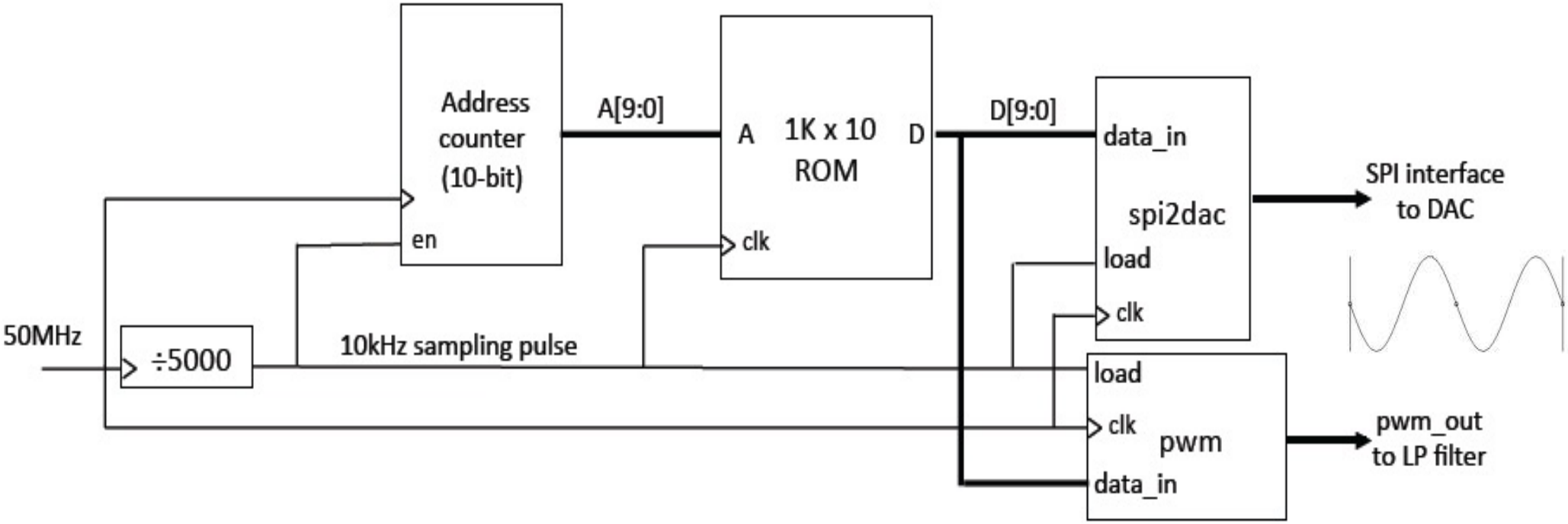
- Create ROM and initialize its content in a .mif file:



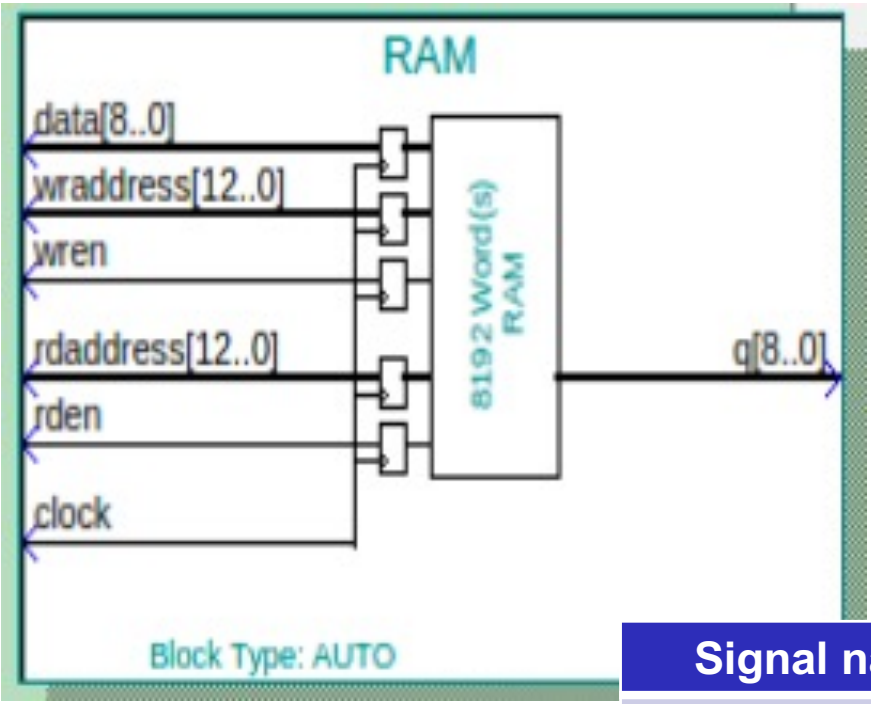

```
-- ROM Initialization file
WIDTH = 10;
DEPTH = 1024;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN
    0 :   200;
    1 :   203;
    2 :   206;
    3 :   209;
    4 :   20C;
    5 :   20F;
    6 :   212;
    7 :   215;
    8 :   219;
    9 :   21C;
    A :   21F;
```

Sinewave Generation

- Generate any waveform or function $y = F(x)$ using table lookup
- Phase counter increment phase whenever *step* goes high
- ROM stores one cycle of sinewave to produce F(x)
- Digital-to-Analogue convert and the PWM DAC generate the analogue outputs on L & R channels



Dual-port RAM (8k x 9)

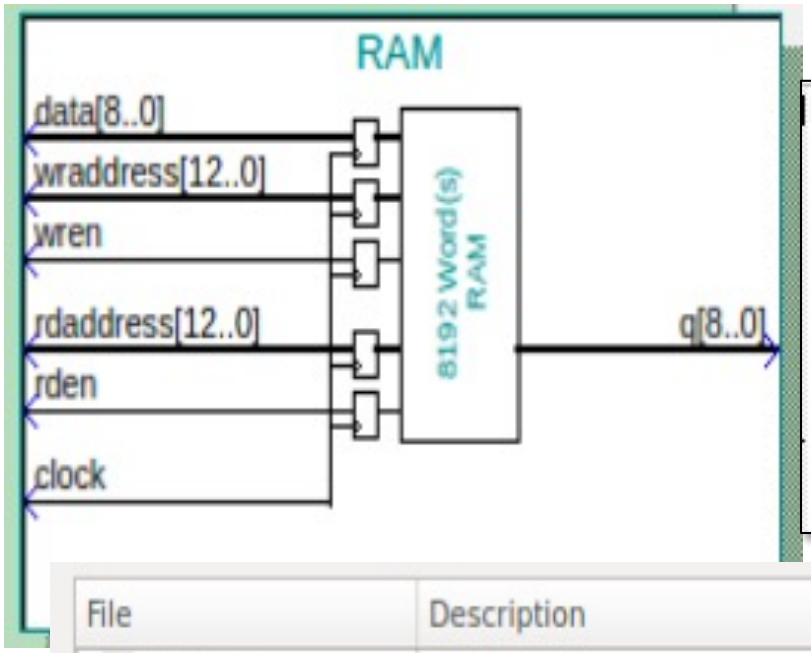


- Limited data width (1 to **9**, 16, 18, 32, 36 …)
- Depth up to 65,536 (4-bit to 16-bit address)
- **Simple dual-port: 1 read port & 1 write port**
- True dual-port: both ports can read/write

Signal name	meaning
data[ ]	Write data port
address[ ]	Read/write address port
q[ ]	Read data port
wren	Write enable
rden	Read enable
clock	Clock signal to control both read & write

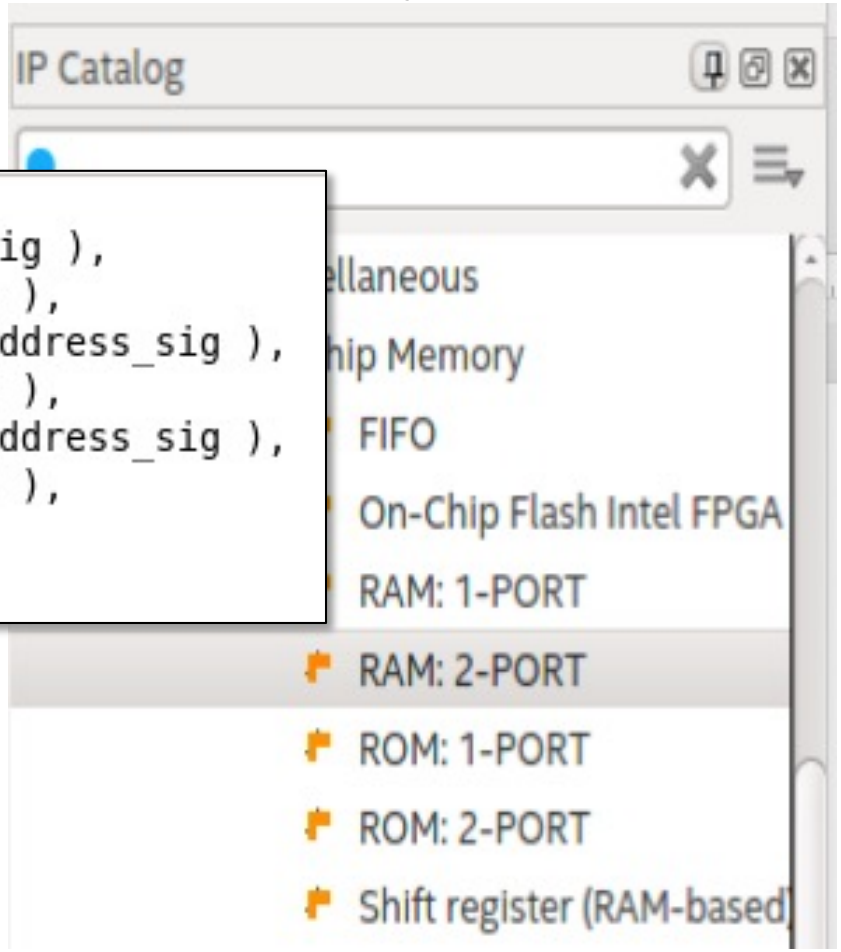
How to use M9K memory block? (8k x 9)

- Use IP Catalog manager tool in Quartus to produce memory of the correct configuration:



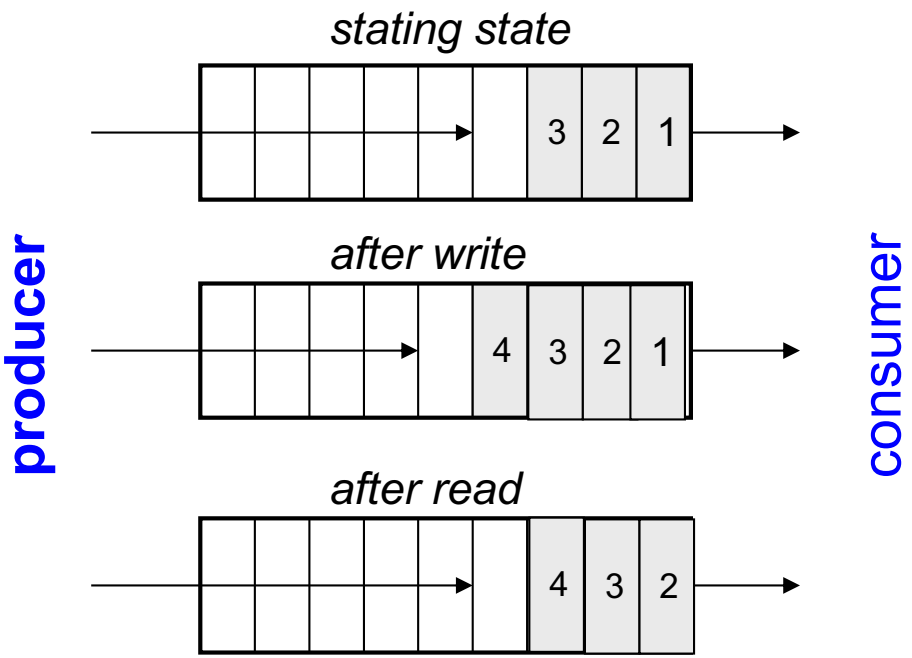

```
RAM (
    .clock ( clock_sig ),
    .data ( data_sig ),
    .rdaddress ( rdaddress_sig ),
    .rden ( rden_sig ),
    .wraddress ( wraddress_sig ),
    .wren ( wren_sig ),
    .q ( q_sig )
    );
```

File	Description
RAM.v	Variation file
RAM.inc	AHDL Include file
RAM.cmp	VHDL component declaration file
RAM.bsf	Quartus Prime symbol file
RAM_inst.v	Instantiation template file
RAM_bb.v	Verilog HDL black-box file



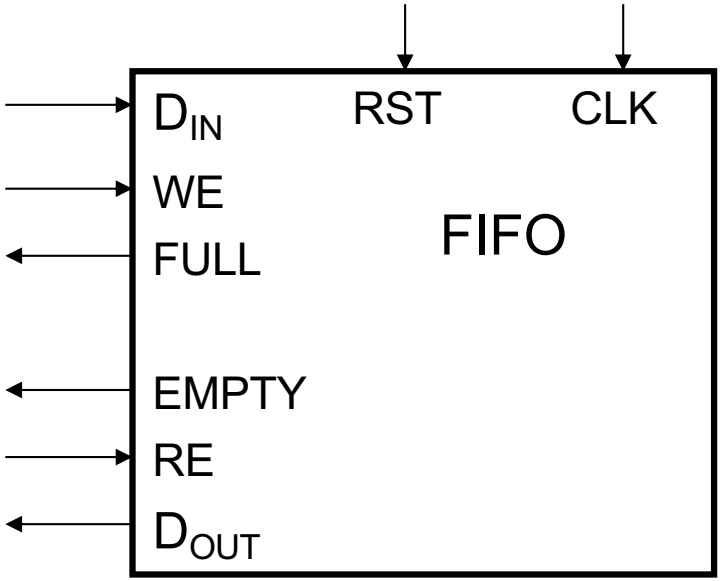
First-in-first-out (FIFO) Memory

- Used to implement *queues*.
- These find common use in computers and communication circuits.
- Generally, used for rate matching data producer and consumer:



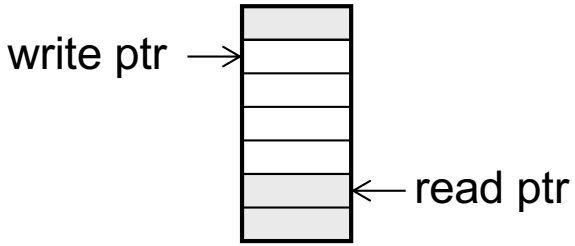
- Producer can perform many writes without consumer performing any reads (or vice versa). However, because of finite buffer size, on average, need equal number of reads and writes.
- Typical uses:
  - interfacing I/O devices. Example network interface. Data bursts from network, then processor bursts to memory buffer (or reads one word at a time from interface). Operations not synchronized.
  - Example: Audio output. Processor produces output samples in bursts (during process swap-in time). Audio DAC clocks it out at constant sample rate.

FIFO Interfaces

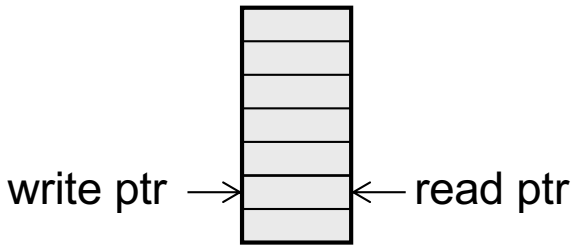


- After write or read operation, FULL and EMPTY indicate status of buffer.
- Used by external logic to control own reading from or writing to the buffer.
- FIFO resets to EMPTY state.

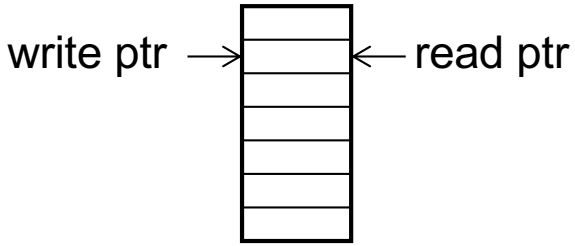
- Address pointers are used internally to keep next write position and next read position into a dual-port memory.



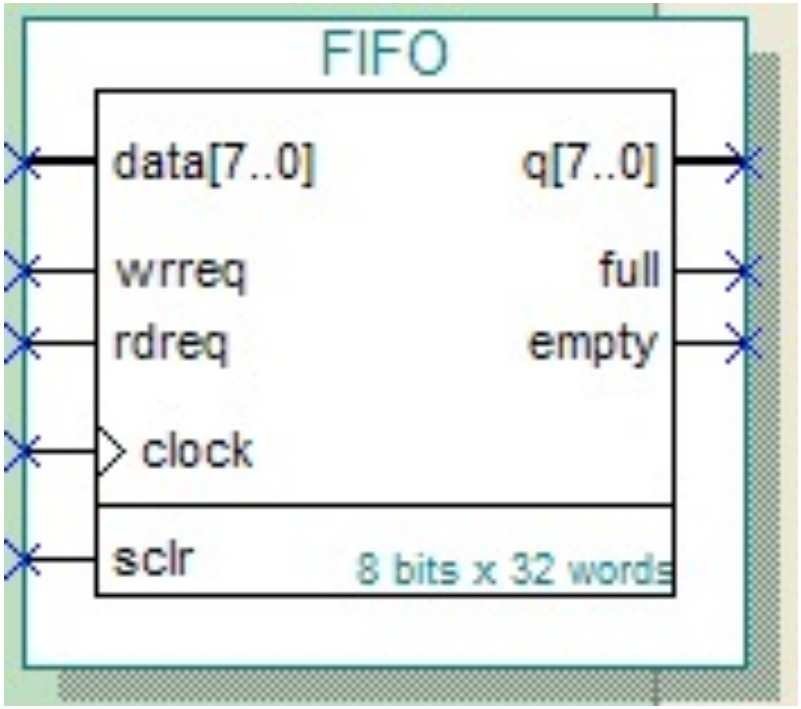
- If pointers equal after write ⇒ FULL:



- If pointers equal after read ⇒ EMPTY:



M9K Memory as FIFO (8-bit x 32 word)




```
module FIFO (
    clock,
    data,
    rdreq,
    sclr,
    wrreq,
    empty,
    full,
    q);

    input     clock;
    input   [7:0]  data;
    input     rdreq;
    input     sclr;
    input     wrreq;
    output    empty;
    output    full;
    output  [7:0]  q;

endmodule
```