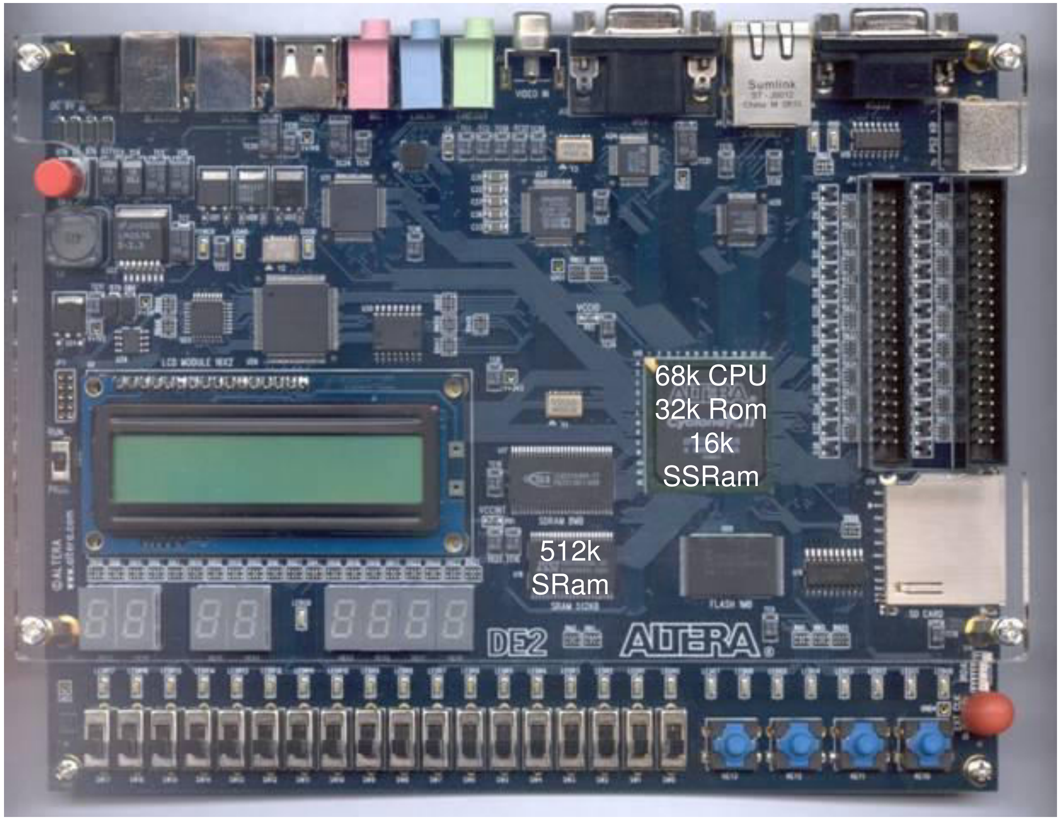
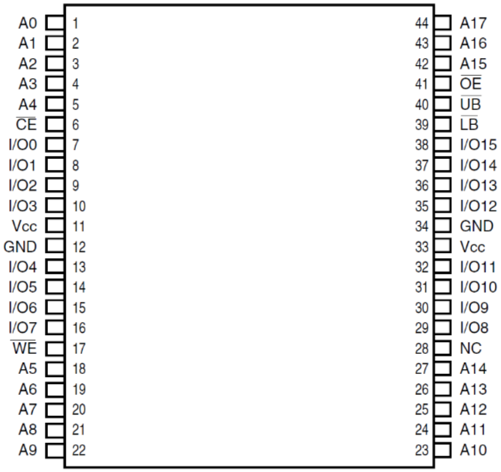
Static Memory Systems Design

- **Async Static (SRAM) Memory Technology**
  - Internal Architecture of the Async SRAM Chip.
  - Operation of a Single bit Sram Cell – read/write operation.
  - Important Asynchronous Sram Timing Parameters
- **Synchronous and Dual Port SRAM**

Asynchronous Static Memory on the DE2

- The *vast majority* of stand-along Sram chips are asynchronous by nature, i.e. they require no clock and operation is defined by the asynchronous timing of control signals, i.e. the levels of the signals rather than clocks/edges.
- The DE2 board has an external 256k x 16 asynchronous SRAM chip made by ISSI, as well as some synchronous Sram on the FPGA.

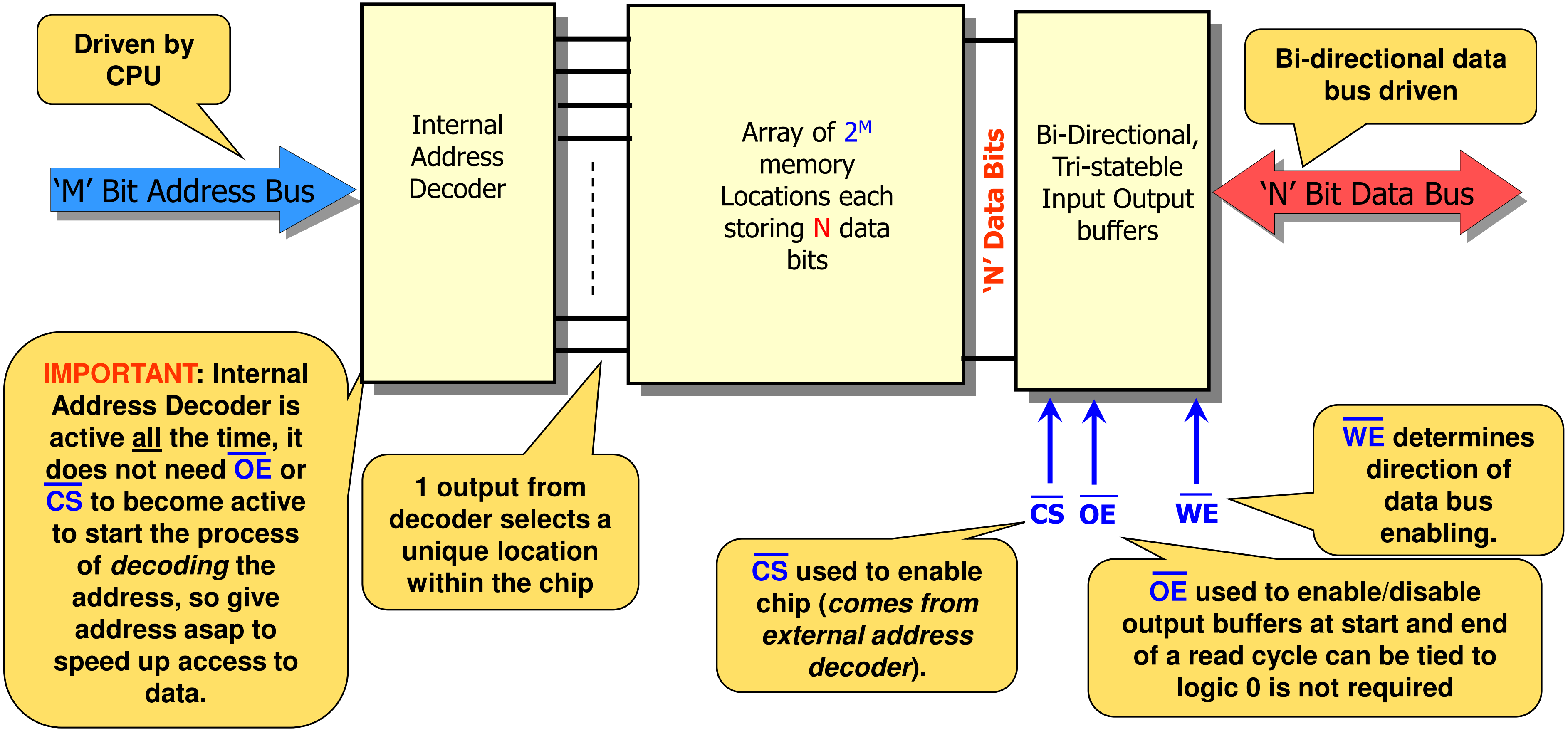




IS61LV25616

General Architecture of an Asynchronous Sram Chip

- Internally an SRAM chip is organised as an array of memory locations.
- Each location is able to store 1 or more bits. Defined by the width of the chips Data Bus.
- ‘M’ address lines are internally decoded to select 1 unique location from 2^M locations.



Driven by CPU

Bi-directional data bus driven

IMPORTANT: Internal Address Decoder is active all the time, it does not need $\overline{OE}$ or $\overline{CS}$ to become active to start the process of *decoding* the address, so give address asap to speed up access to data.

1 output from decoder selects a unique location within the chip

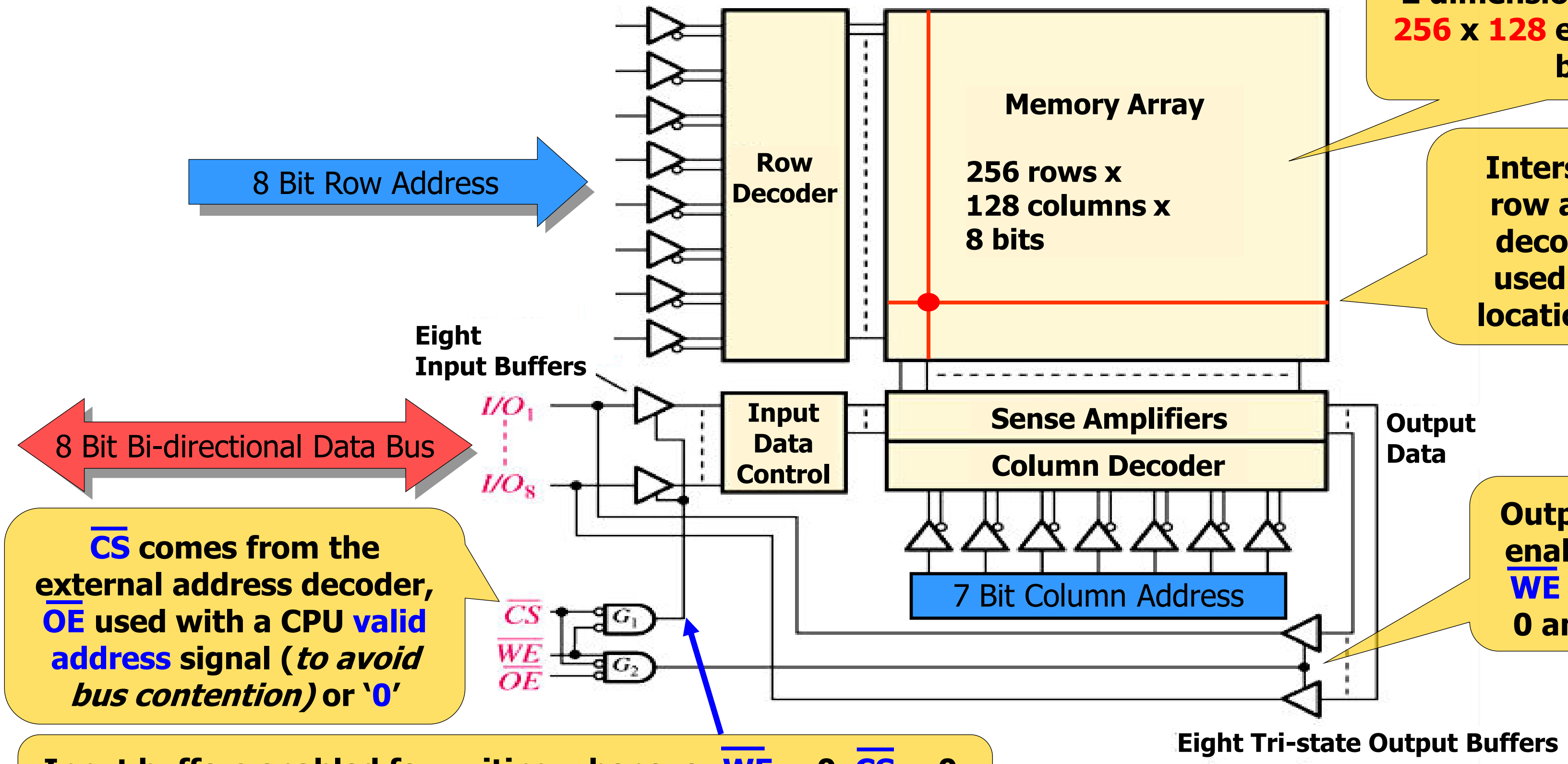
$\overline{CS}$ used to enable chip (*comes from external address decoder*).

$\overline{WE}$ determines direction of data bus enabling.

$\overline{OE}$ used to enable/disable output buffers at start and end of a read cycle can be tied to logic 0 is not required

Architecture of a 32k x 8 Asynchronous Sram Chip

- To simplify the internal address decoder, a 32k device (with 15 address lines) is physically organised into two smaller row and column decoders.
- The Row decoder is fed with 8 (of the 15) address lines selecting 1 of 256 rows.
- The Column decoder is fed with the remaining 7 address lines selecting 1 out of a possible 128 columns. The intersection of a row and column output is used to select a location within a 2 dimensional array within the chip.



2 dimensional matrix of 256 x 128 each storing 8 bits

Intersection of a row and column decoder output used to select a location of '8' bits

Output buffers enabled when $\overline{WE}$ = 1, $\overline{CS}$ = 0 and $\overline{OE}$ = 0

$\overline{CS}$ comes from the external address decoder, $\overline{OE}$ used with a CPU valid address signal (*to avoid bus contention*) or '0'

Input buffers enabled for writing whenever $\overline{WE}$ = 0, $\overline{CS}$ = 0. Can lead to false writes if address is not valid ($\overline{WE}$ should be gated with 68k's $\overline{AS}$ signal to avoid false writes.)

Static Memory Technology

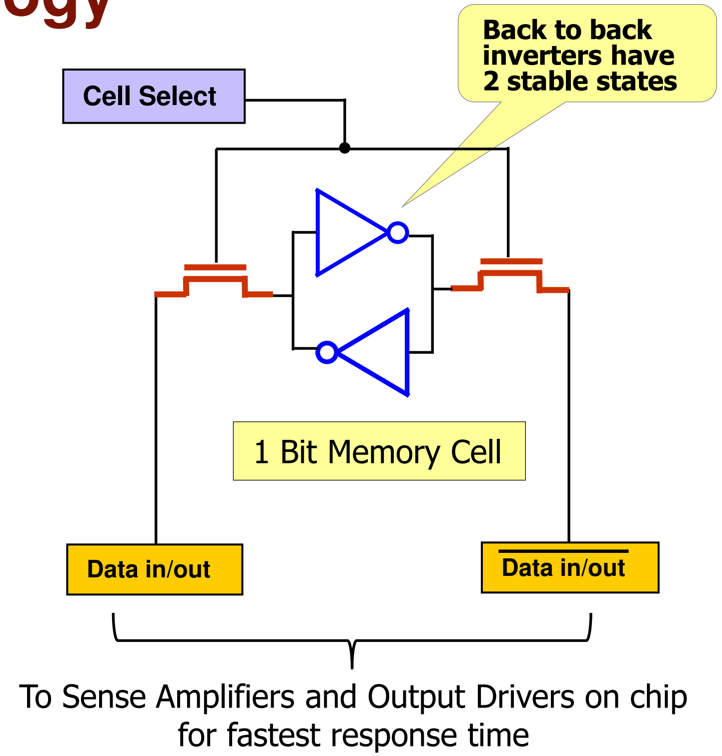
Static (SRAM) Memory Cell

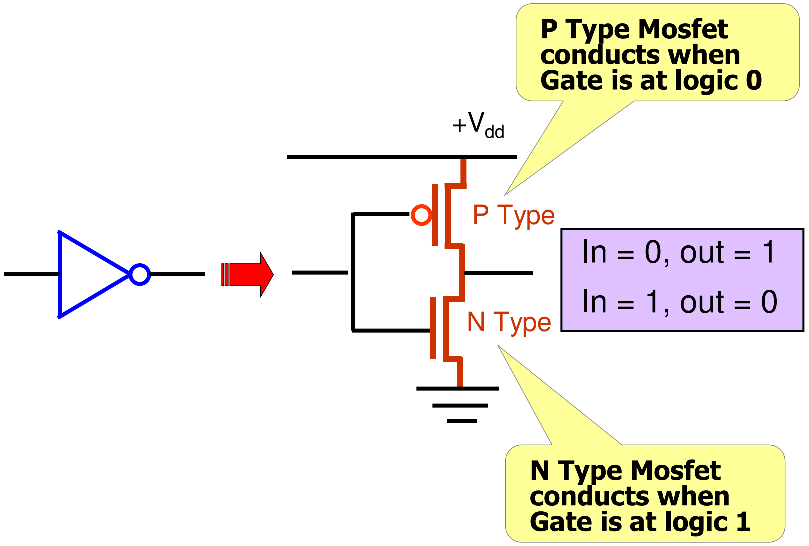
- The illustration opposite is that of an SRAM cell able to store a single bit of data, requiring six Mosfets (6T cell).

- Four low power Mosfets are used to make a pair of inverters – i.e. NOT gates.

- The two inverters are connected in a back-to-back loop, with the output of one connected as the input of the other, reinforcing each other, providing two **stable states** at the output of each inverter: ‘0’ or ‘1’. It is these inverters that perform the data storage.

- The other two Mosfets control activation of the cell for reading and writing purposes and are activated by memory chip’s internal address decoder, selecting a cell within the chip.

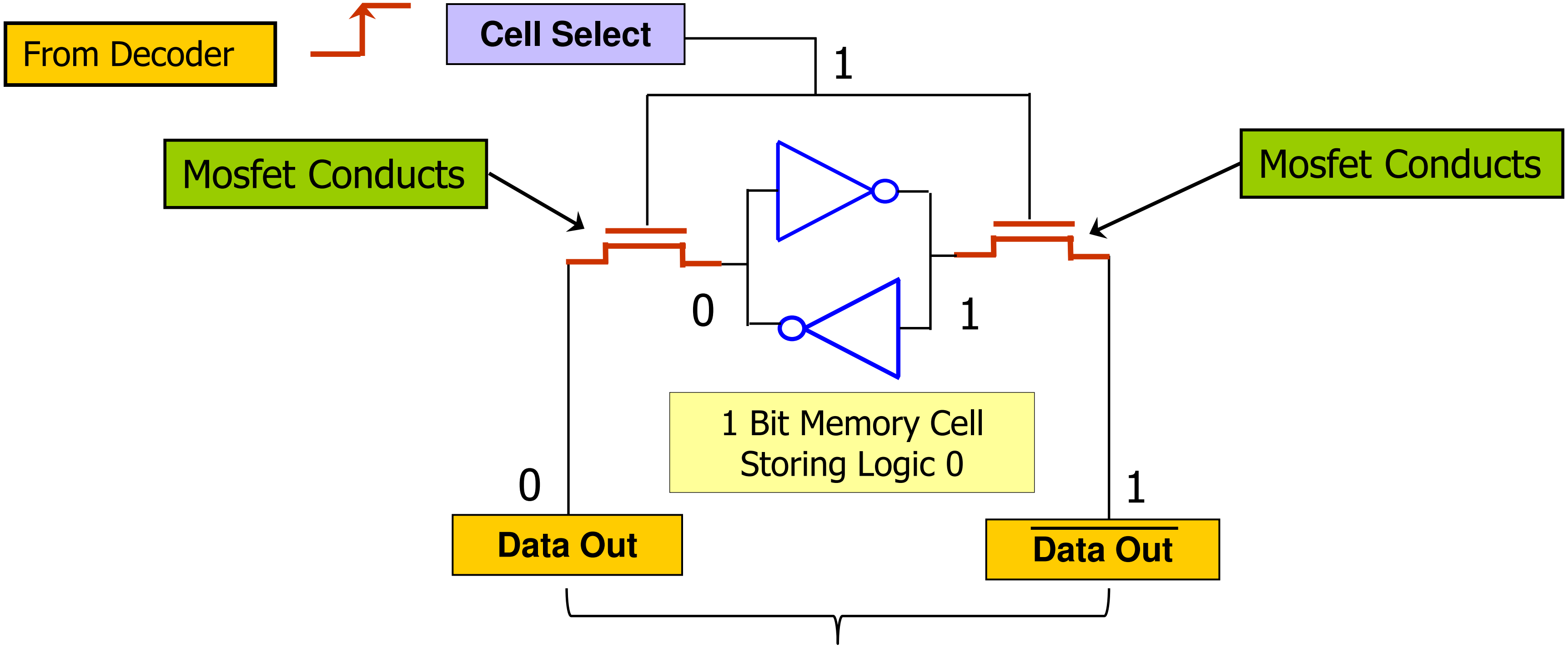

Cell Select
Back to back inverters have 2 stable states
1 Bit Memory Cell
Data in/out
$\overline{\text{Data in/out}}$
To Sense Amplifiers and Output Drivers on chip for fastest response time


P Type Mosfet conducts when Gate is at logic 0
$+V_{dd}$
P Type
N Type
In = 0, out = 1
In = 1, out = 0
N Type Mosfet conducts when Gate is at logic 1

Static Memory Technology

Reading

- The *cell select* line is activated (*logic 1*) by the internal address decoder in response to a CPU address. The signal is formed from the 'AND' of the internal row and column decoders. This turns on the two control Mosfets and the differential (*opposite*) value of the two inverter outputs is fed to a sense amplifier which drives the Logic 0 or 1 at the chip output buffers.

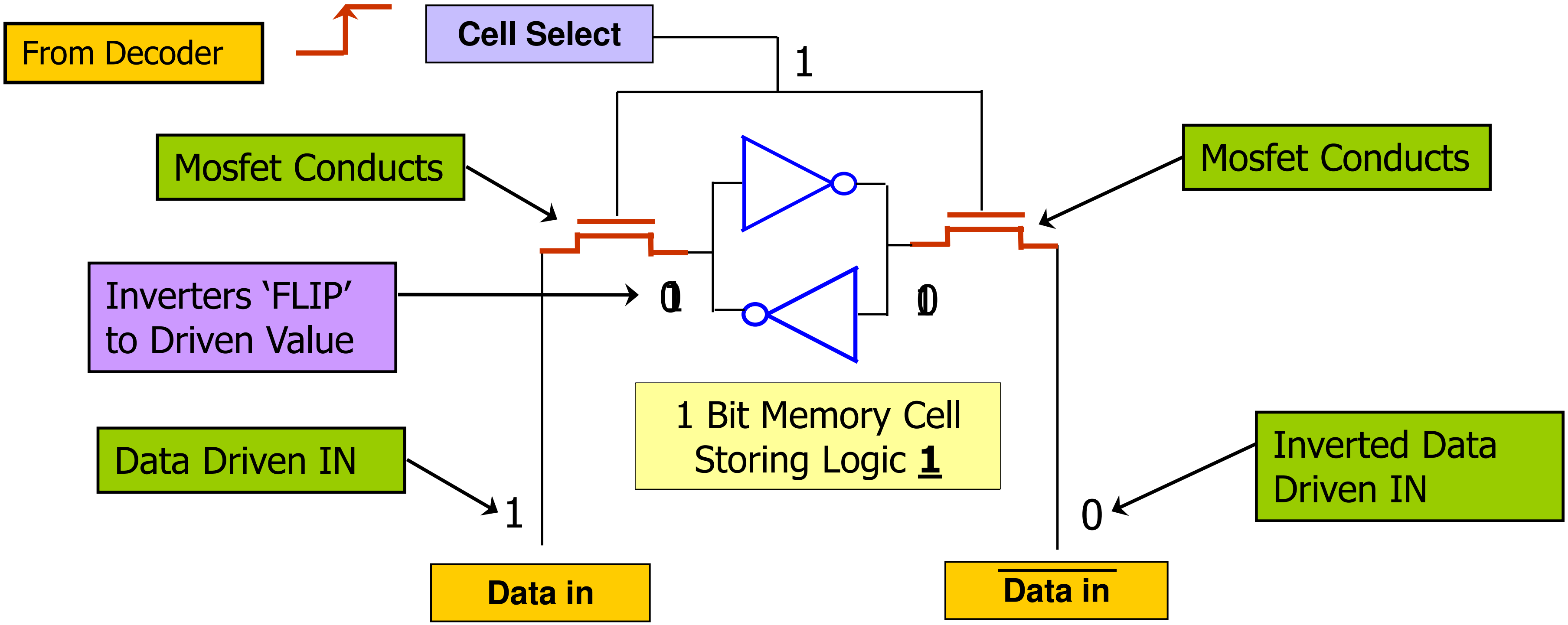


To "on chip" **Sense Amplifiers** and
Output Drivers for fastest response time

Static Memory Technology

Writing

- To write to the cell, the *cell select* line is again taken high in response to a particular address, but this time the CPUs data is driven into the *Data in/out* line (*and its inverse*). The higher power control mosfets literally overpower the inverters to adopt the appropriate stable state.



From Decoder

Cell Select

Mosfet Conducts

Mosfet Conducts

Inverters 'FLIP' to Driven Value

1 Bit Memory Cell Storing Logic **1**

Data Driven IN

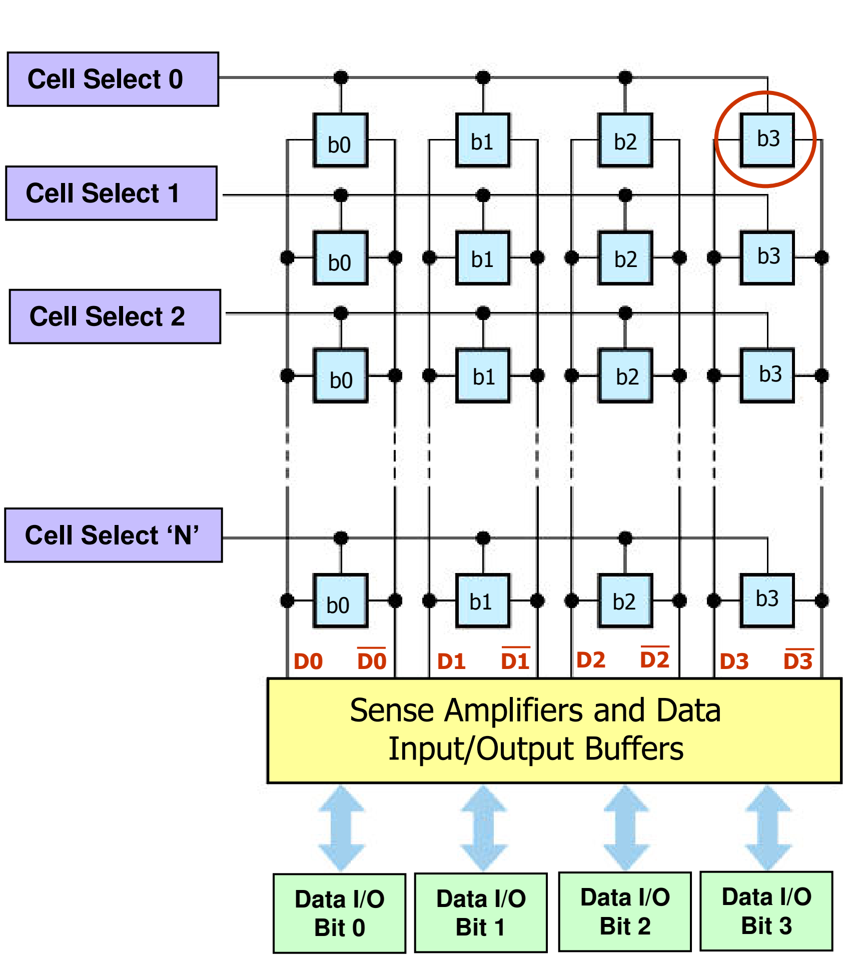
Inverted Data Driven IN

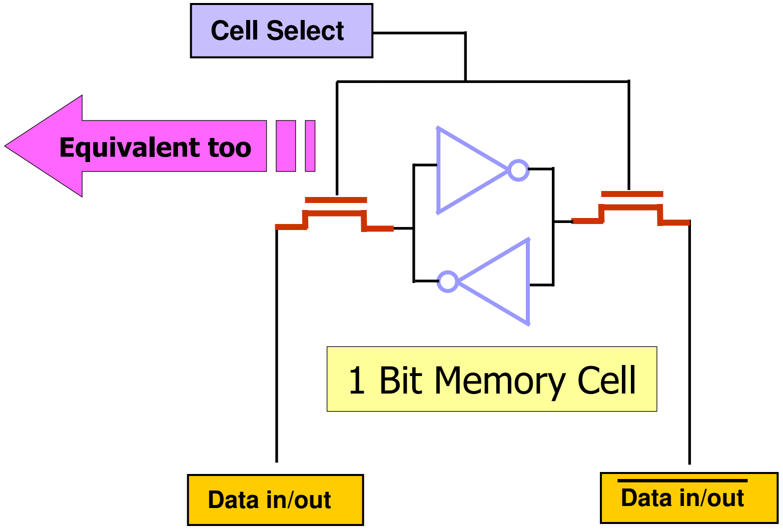
Data in

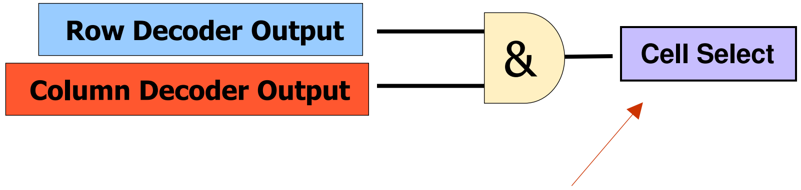
$\overline{\text{Data in}}$

Static Memory Technology

- The illustration below shows how a 4 bit wide memory chip might be organised internally.
- Each blue cell corresponds to the 6 transistor implementation seen previously, capable of storing a single bit of data. Thus in this case, 4 cells are activated by each cell select line.
- The cell select lines correspond to Row and Column decoder outputs And'ed together.







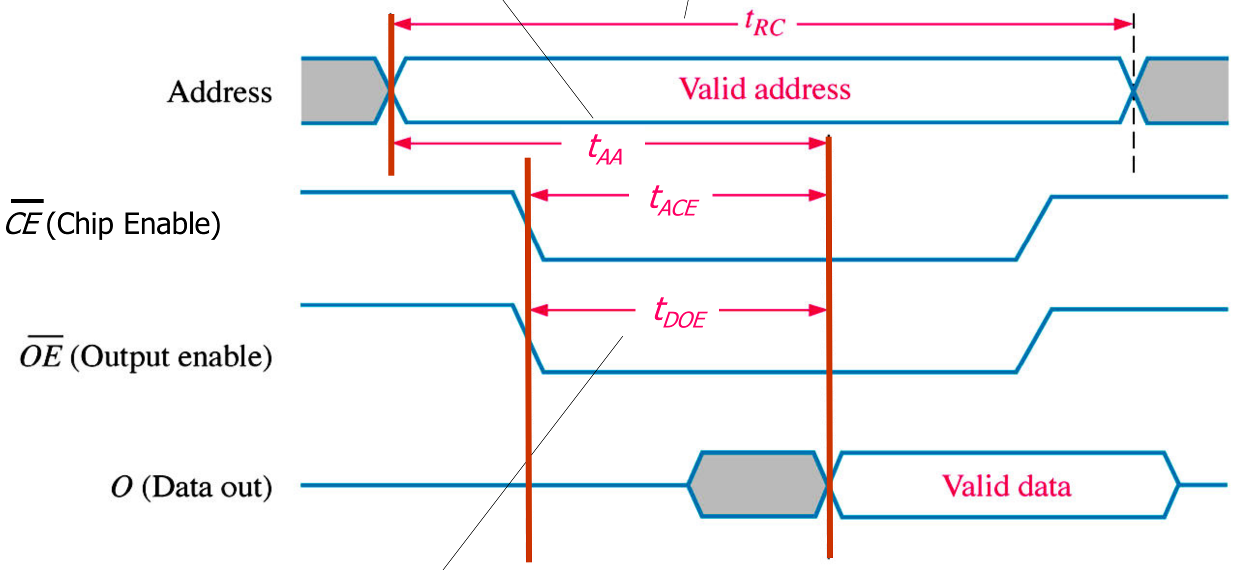
Cell select signal comes from the bitwise '**AND**' of the **Row** and **Column** Address Decoder line outputs

Asynchronous Static Memory Timing Diagrams – Read Cycle

- An appreciation of the timing characteristics of an Async Sram chip is essential for determining if a specific device will interface to the CPU correctly. For example, is it **fast enough** to supply the data, during a CPU read operation.

t_{AA} – The *worst case* **Read Access Time** of the chip after acccess becomes stable. A **key** parameter

t_{RC} – The **Read Cycle Time** of the chip, the **MINIMUM** time for a valid read operation. Chip is not guaranteed to read correctly if address is not stable for this length of time.



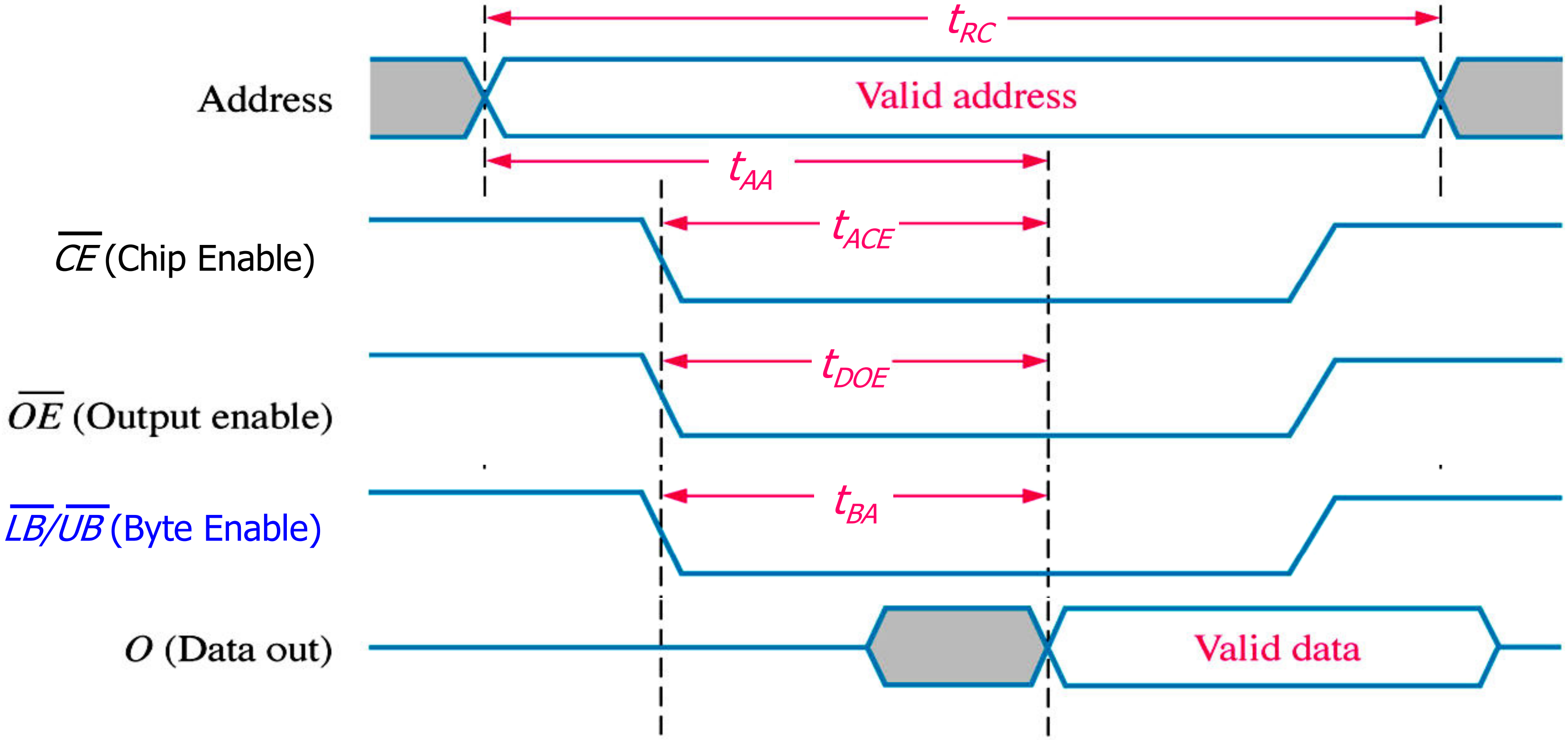
t_{ACE} and t_{DOE} – The *worst case* time taken to **turn on** the output buffers of the chip after presentation of **$\overline{CE}$** and/or **$\overline{OE}$** which **can** delay the data out if either is late or delayed.

Key design decisions for READ operation:

1) Make sure **$\overline{CE}$** and **$\overline{OE}$** arrive early so as **not** to delay turning on output buffers inside the memory chip (usually easy)
2) Get the CPU **address** to the chip **ASAP** don't wait for $\overline{AS}$. This means internal decoders can go to work on the address quickly.
3) Make sure CPU waits T_{AA} seconds before "reading" Data.

Example Static Ram Memory Chip

The ISSI IS61LV25616-10 256K x 16 Async Static Ram (DE2) – Read Cycle Parameters



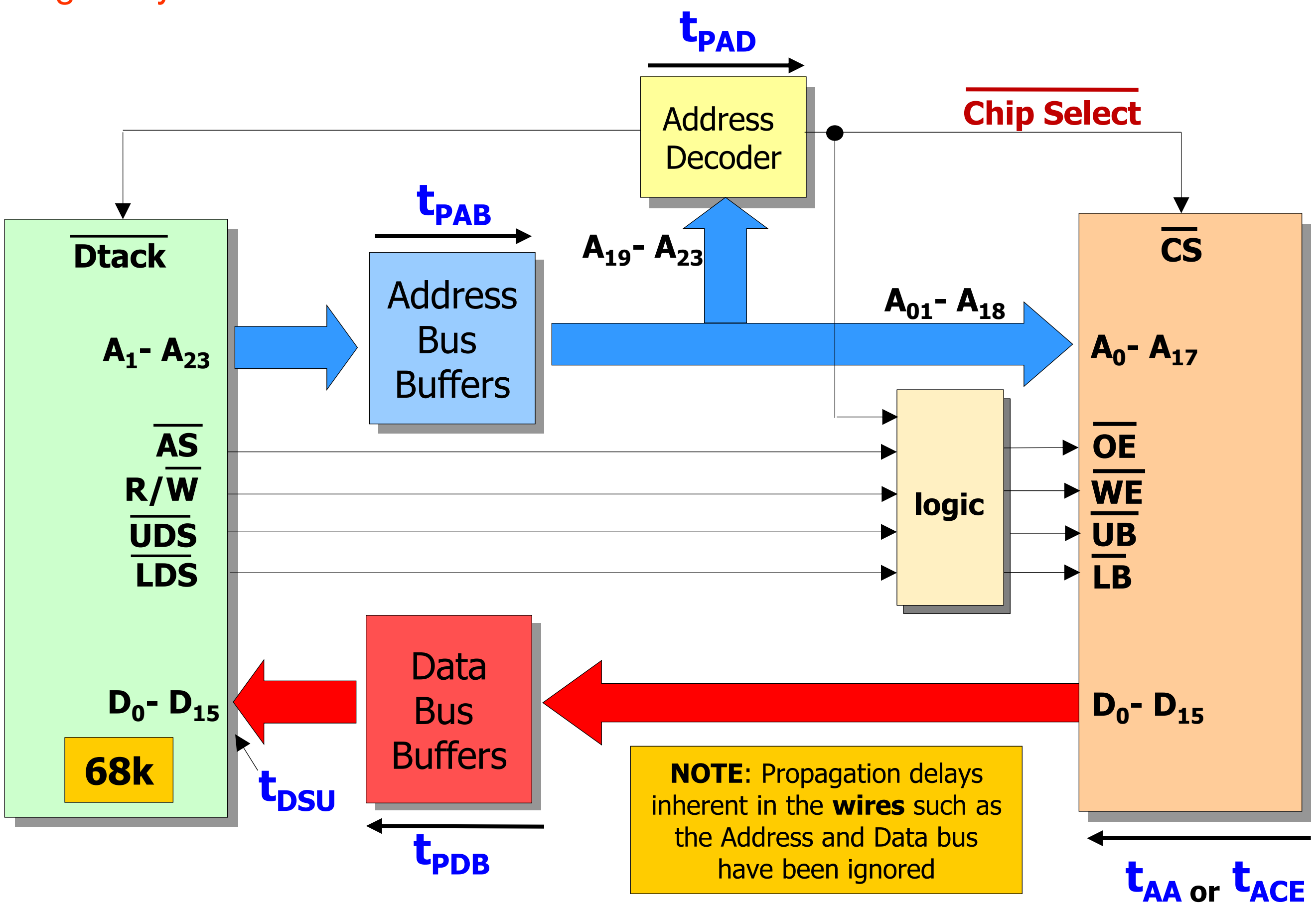
t_{RC}
Address
Valid address
t_{AA}
$\overline{CE}$ (Chip Enable)
t_{ACE}
t_{DOE}
$\overline{OE}$ (Output enable)
t_{BA}
$\overline{LB}/\overline{UB}$ (Byte Enable)
O (Data out)
Valid data

Parameter	Symbol	Min	Max
Read Cycle Time	t_{RC}	10ns	
Address Access Time	t_{AA}		10ns
Chip Enable Access Time	t_{ACE}		10ns
Output Enable Access Time	t_{DOE}		4ns
Byte Enable Access Time	t_{BA}		4ns

Propagation Delays Inherent in Memory Systems

Other Timing Considerations

- The block diagram below show a simple 68k CPU interfacing to a 256K word memory chip. The Parameters, t_{PAB} , t_{PAD} , t_{PDB} and t_{AA} indicate the various signal propagation delays and access times that may be encountered (*depending on inclusion of buffers etc.*) along the way, during a CPU **read** cycle. Time t_{DSU} is the Data **setup** time back at data pins of the CPU. Why are these important?
- When the CPU outputs an address, it will expect to sample the data some time later. Our job is to do the timing analysis and make sure that our data is back at the CPU in time to be read.



t_{PAD}

Address Decoder

$\overline{\text{Chip Select}}$

t_{PAB}

A_{19}- A_{23}

A_{01}- A_{18}

$\overline{\text{Dtack}}$

A_1- A_{23}

$\overline{\text{AS}}$

R/$\overline{\text{W}}$

$\overline{\text{UDS}}$

$\overline{\text{LDS}}$

D_0- D_{15}

68k

Address Bus Buffers

logic

$\overline{\text{CS}}$

A_0- A_{17}

$\overline{\text{OE}}$

$\overline{\text{WE}}$

$\overline{\text{UB}}$

$\overline{\text{LB}}$

D_0- D_{15}

Data Bus Buffers

t_{DSU}

t_{PDB}

NOTE: Propagation delays inherent in the **wires** such as the Address and Data bus have been ignored

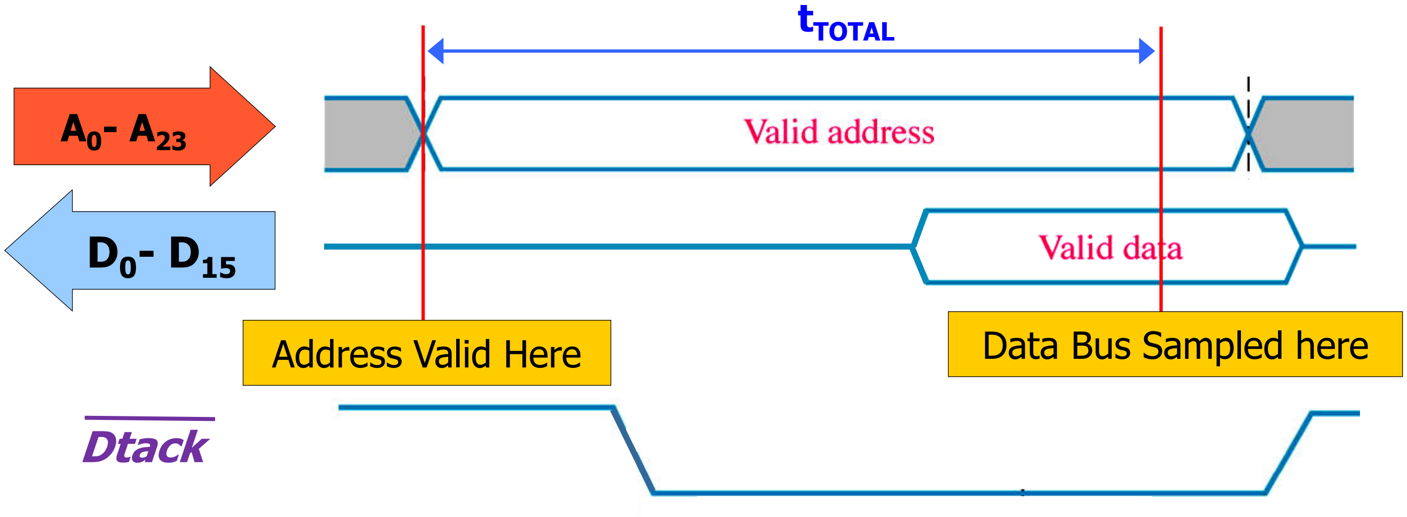
t_{AA} or t_{ACE}

256K x 16 SRAM Chip

Access time from stable address (t_{AA}) or from CS* (t_{ACE})

Typical CPU Read Cycle Timing

- The timing diagram below shows a simple but typical 68k read cycle in action.
- An address is issued, and some time t_{TOTAL} later, after Dtack received, the CPU will expect the memory system to have decoded the address and supplied the data back at the CPU data inputs.



- We can see that in order for the memory system to work correctly for a CPU **read**

$$t_{TOTAL} > t_{PAB} + t_{AA} + t_{PDB} + t_{DSU}$$

$$t_{TOTAL} > t_{PAB} + t_{PAD} + t_{ACE} + t_{DSU}$$

Synchronous Sram

- Synchronous Sram (SSRAM) is a development of Asynchronous Sram for very high speed systems and uses a clock synchronised to the CPU. Often used as cache memory.

- Same internal Architecture and Sram Cell (6 Mosfets per bit) etc.

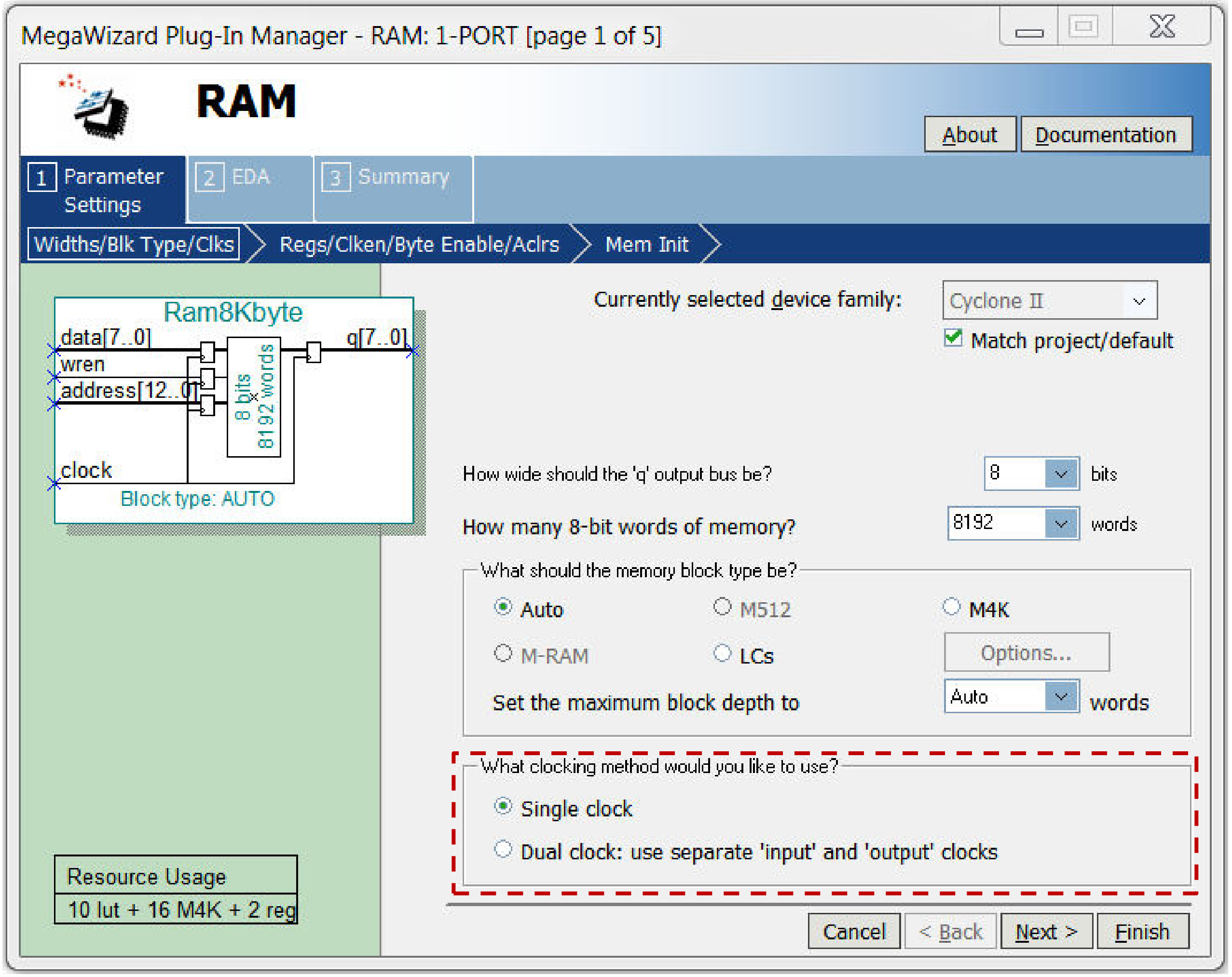
- Asynchronous control signals such as CS*, Address Bus, Data Bus, R/W* etc. latched internally (*see below right for latches on input/output signals*) on a clock edge, meaning address and data-in etc. do not need to be held for duration of read/write, i.e. they can change while read/write is in progress making successive access's faster.

- Data out may also be latched as an option but this delay access to memory by 1 more clock cycle.

- Some SSRams may have internal counters to auto increment the last address, allowing burst reading or writing of data from/to chip, one data item per clock without need for a new address from CPU.

Synchronous Sram

- Altera's FPGAs only support SSRam. An example illustration of the Megawizard in Quartus being used to construct 8k bytes of SSRam is shown below.



MegaWizard Plug-In Manager - RAM: 1-PORT [page 1 of 5]

Currently selected device family: Cyclone II

☑ Match project/default

How wide should the 'q' output bus be? 8 bits

How many 8-bit words of memory? 8192 words

What should the memory block type be?

● Auto ○ M512 ○ M4K

○ M-RAM ○ LCs

Set the maximum block depth to Auto words

What clocking method would you like to use?

● Single clock

○ Dual clock: use separate 'input' and 'output' clocks

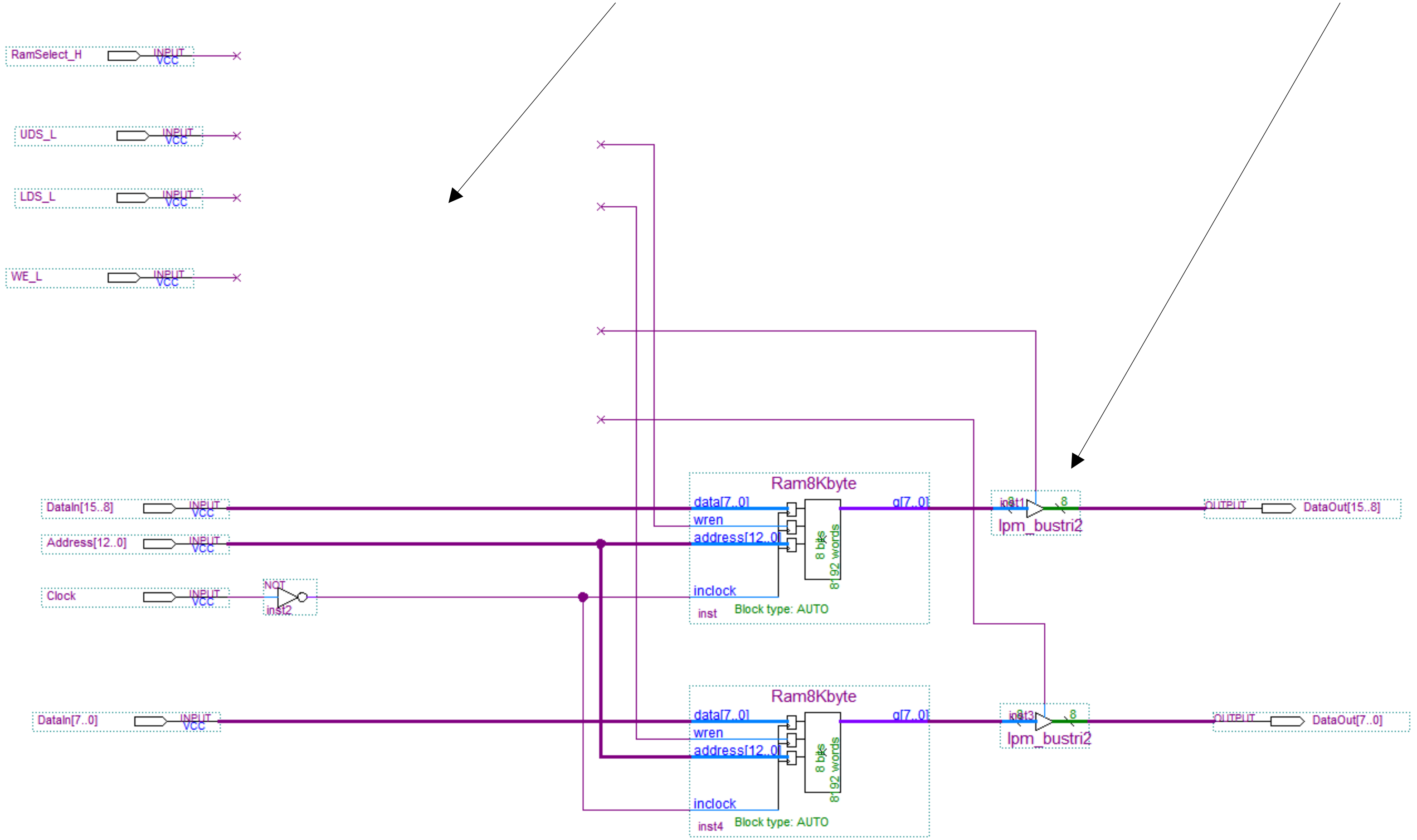
Resource Usage

10 lut + 16 M4K + 2 reg

Synchronous Sram

Class Exercise 1:

- Think about the logic to drive the Sync SRam and the output buffers below.



- Don’t worry about timing at the moment, just functionality

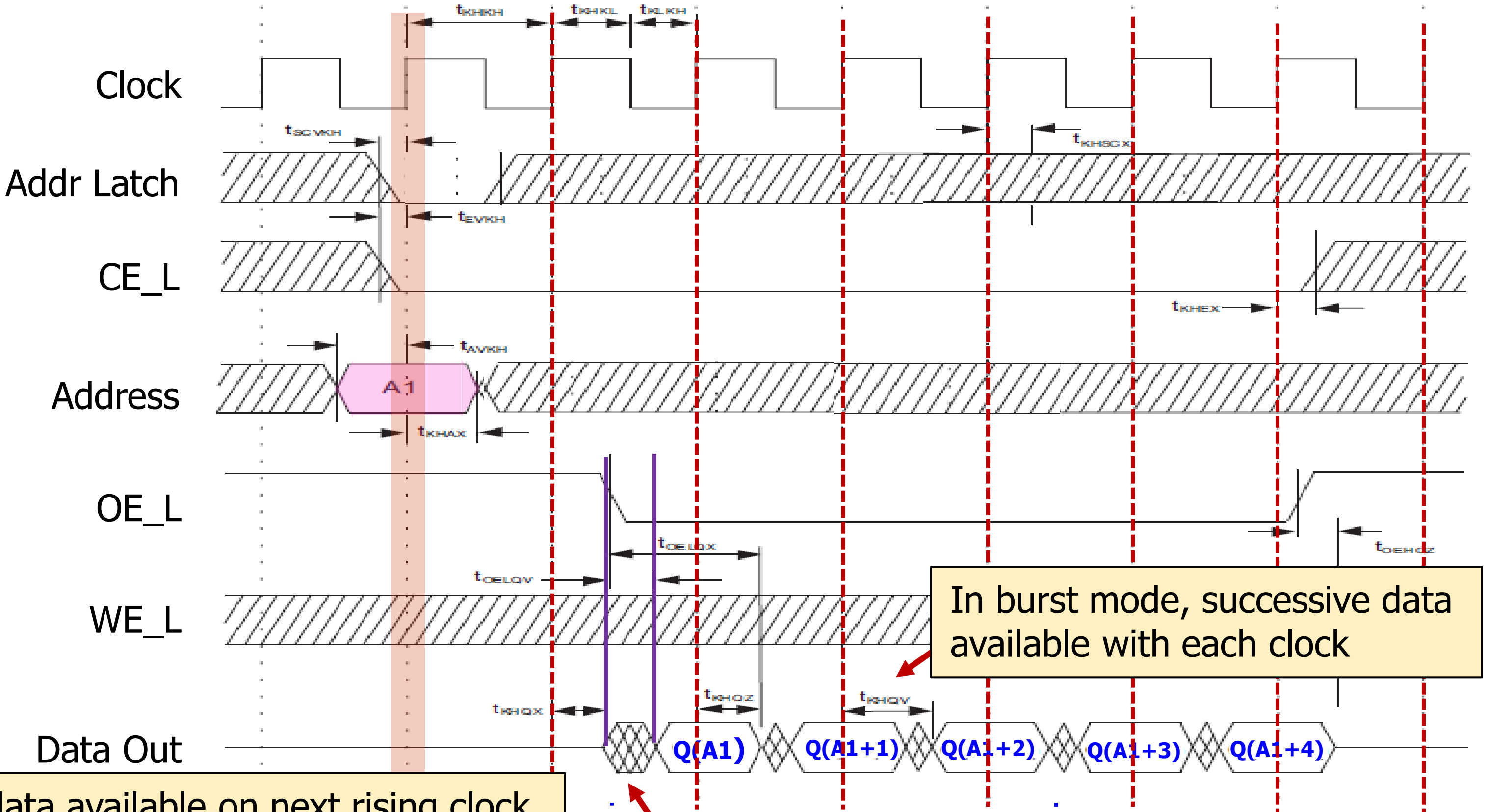
Tri-State Memory Outputs

Class Exercise 2:

- Why do memory chips typically need **tri-state outputs**?
- What is wrong with just connecting the **memory outputs** directly back to the CPU **data inputs**?

Synchronous Sram Read Operation Timing

- An example of a ssram chip read cycle timing is shown below for the Microsemi WED2DL32512V 512k x 32 **pipelined, burst mode** device.
- CE, WE, Addr Latch and Address signals latched on rising clock edge.
- In non burst mode, data out available on next clock edge, subject to no delay in asserting OE_L



In burst mode, successive data available with each clock

1st item of data available on next rising clock, but can be delayed by late arrival of OE_L

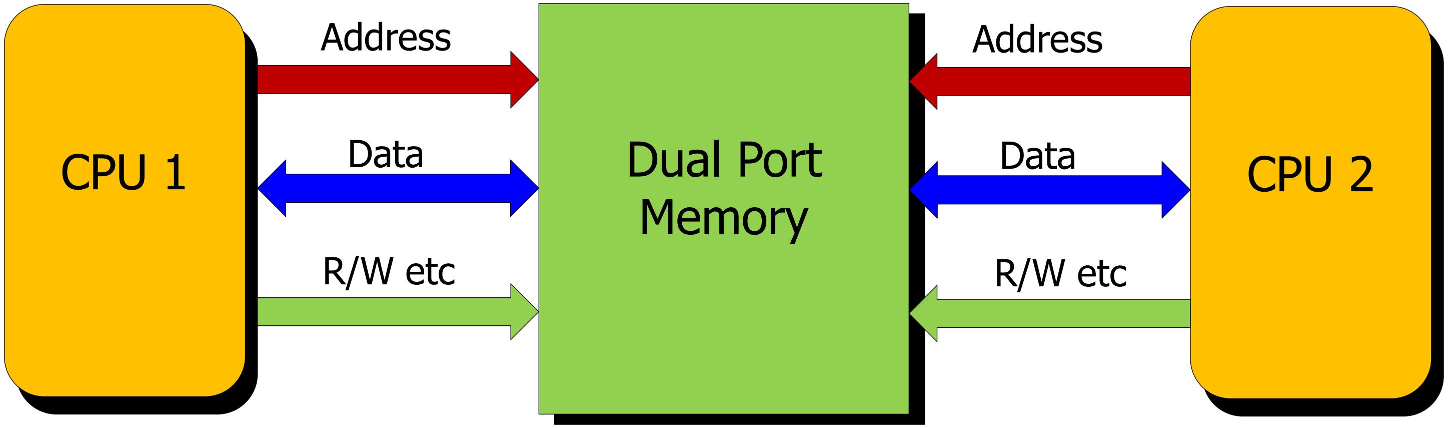
Output buffer turn on time from OE_L

Dual Ported Sram

- Dual Port Sram has two address, two data and two control buses to allow two devices to simultaneously share the same memory without external synchronisation or *multiplexing* of those signals.
- **Rare** to see this for ***external*** chips due to the number of pins required, however it's common on ASICs and FPGAs.

Application 1: - Multi-processor applications:

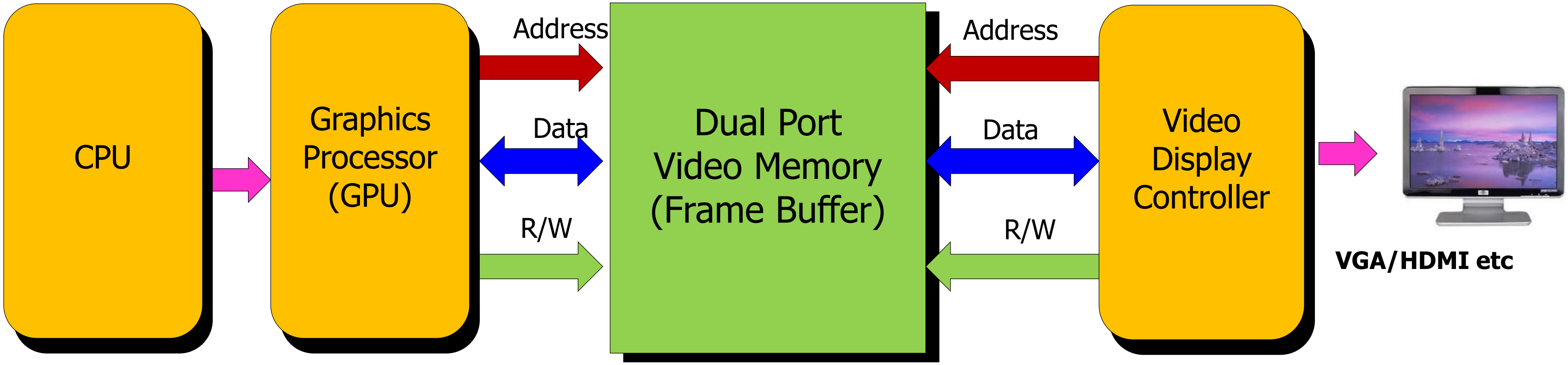
- Two CPUs can share data through the use of dual port Ram - Some Sram chips have built in hardware semaphores for processor synchronisation so that read/modify/writes by two processors to the same locations do not interfere e.g. two processors trying to say increment the data in the same location at the same time.



Dual Ported Sram

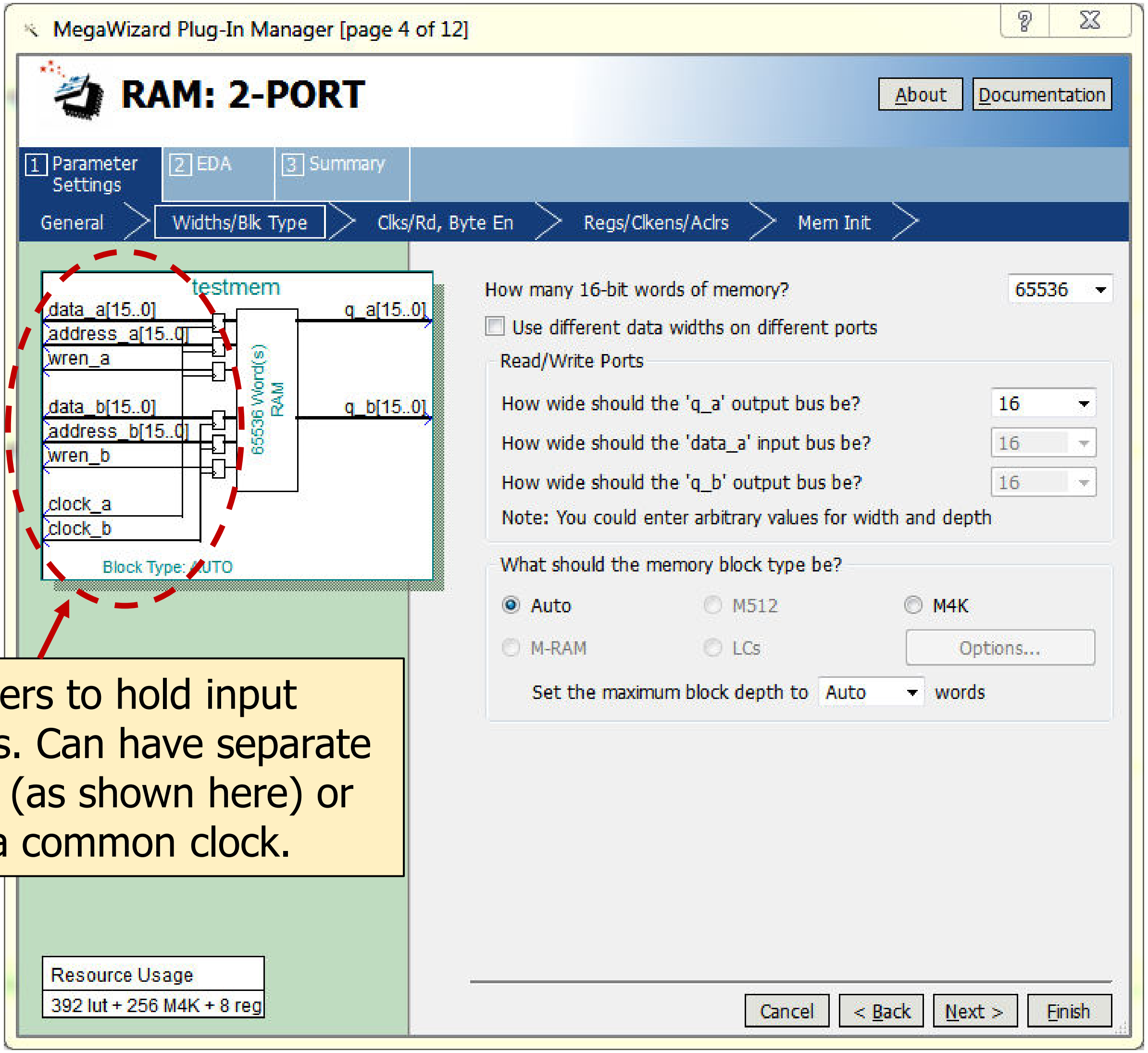
Application 2: Video Memory frame buffer

- Dual port memory also enables a video display controller to display images at the **same time** as the CPU/GPU is updating the image (i.e. writing to video memory).
- This reduces ***flicker*** that might occur on the display monitor if **single port memory** had been used (*due to the GPU taking control of the memory chip's only/single port during drawing/writing*.) and **eliminates** the external multiplexer that would be required to switch the memory between Video controller and GPU.
- It also means the GPU can draw a new image at full speed without having to wait for the **horizontal** and **vertical** "**sync**" periods when the video controller is not displaying an image (*at the end of a line or frame*).



Dual Ported Synchronous Sram

- Quartus can synthesize **dual port** *synchronous* sram on an FPGA



MegaWizard Plug-In Manager [page 4 of 12]

RAM: 2-PORT

How many 16-bit words of memory? 65536

☐ Use different data widths on different ports

Read/Write Ports

How wide should the 'q_a' output bus be? 16

How wide should the 'data_a' input bus be? 16

How wide should the 'q_b' output bus be? 16

Note: You could enter arbitrary values for width and depth

What should the memory block type be?

● Auto ○ M512 ○ M4K

○ M-RAM ○ LCs Options...

Set the maximum block depth to Auto words

Resource Usage

392 lut + 256 M4K + 8 reg

Registers to hold input signals. Can have separate clocks (as shown here) or have a common clock.