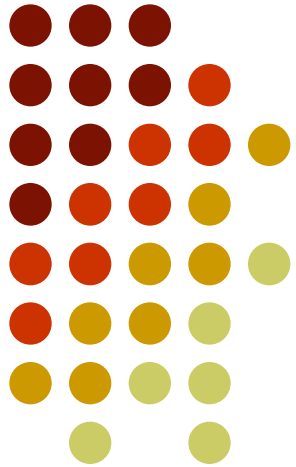
Static Memory Systems Design

- **The 68000 Asynchronous Bus Cycle**
  - 68k Read and Write Bus cycles
    - 68k Timing Diagrams, Min/Max Parameters etc.
    - Data Exchange Protocols: Dtack Timing, Wait States.
  - Analysis of 68000 and Static Ram Timings for compatibility.
    - Soft Core 68k Simulation of Read/Write Operation in Quartus
  - Design of Simple, Flexible DTACK* Generator Circuit.

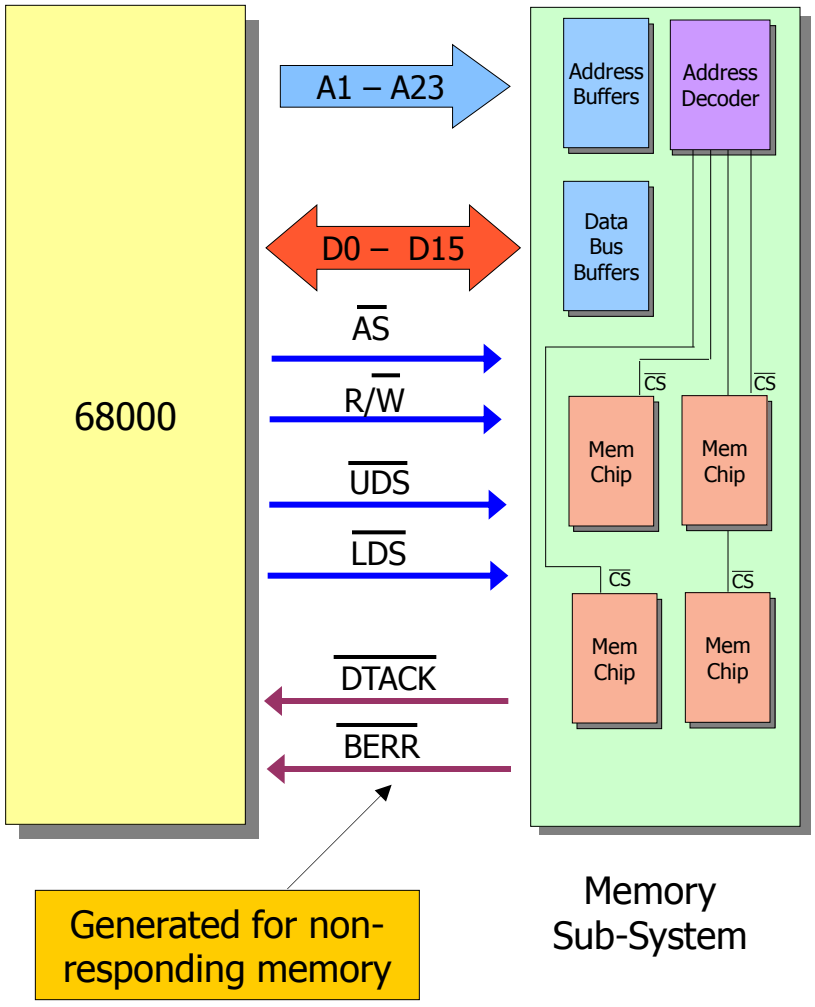


68000 Asynchronous Bus interface

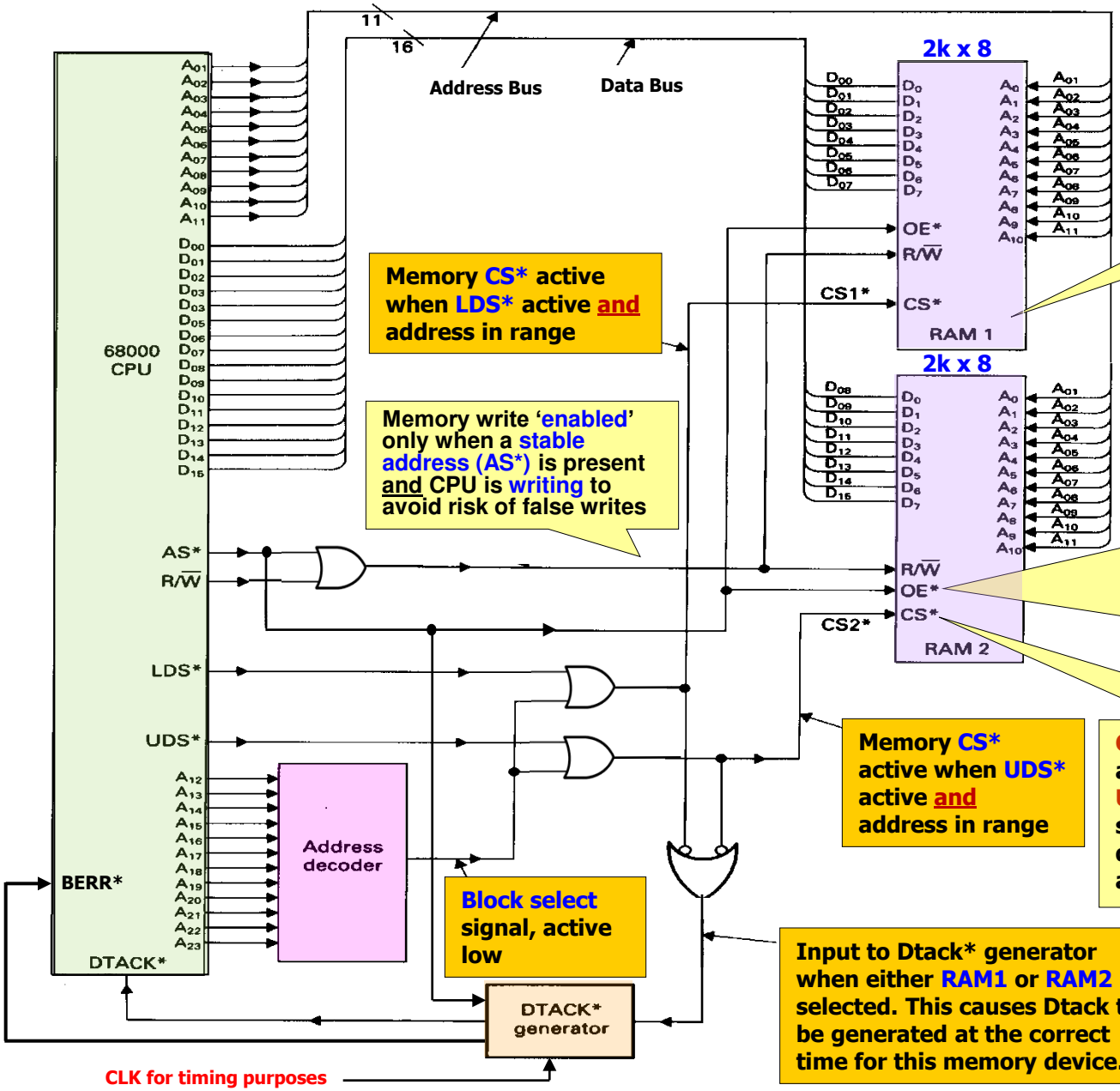
- The 68k as we know uses an *asynchronous* memory interface, the basics of which are:-
  - The 68000 outputs an address over A1-A23.
  - Address Strobe ($\overline{AS}$) asserted when the address is valid and stable.
  - R/$\overline{W}$ is issued to indicate direction of data transfer.
  - $\overline{UDS}$ and $\overline{LDS}$ used to indicate the size of data.
  - Data transferred over D0-D15
  - Memory confirms *read or write* with $\overline{DTACK}$
  - When 68000 receives the $\overline{DTACK}$ it terminates the bus cycle by negating $\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$ and $\overline{LDS}$ and removing the address.
- Bus Error ($\overline{BERR}$) is asserted for non-responding memory (*i.e. illegal address*).

Advantages/Disadvantages

- An advantage of an asynchronous memory protocol is that it allows the CPU to interface with both fast and slow memories at whatever speed they are able to communicate (*they just provide the DTACK* when they are ready*).
- A disadvantage is that the need for handshaking (via $\overline{DTACK}$) can slow down the overall data rate in very high speed systems.



Typical Circuit Diagram for 68000 to SRam, e.g. 2k x 16



Two byte wide chips working in parallel each on their own half of the data bus.

Memory CS* active when LDS* active and address in range

Memory write 'enabled' only when a stable address (AS*) is present and CPU is writing to avoid risk of false writes

Memory OE* active when AS* is valid during a read cycle

(Memory chip s*till requires that CS* is active and R/W* = 1 before it will enable its output buffers). Use of AS* to drive OE* means output definitely disabled between successive bus cycles to avoid contention as AS* is removed early at the end of the bus cycle, well before starting another one.*

Memory CS* active when UDS* active and address in range

CS1*/CS2* derived from the address decoder logic gated with UDS*/LDS*. That is chip is only selected when the address and data width indicate that the chips are being selected. IMPORTANT

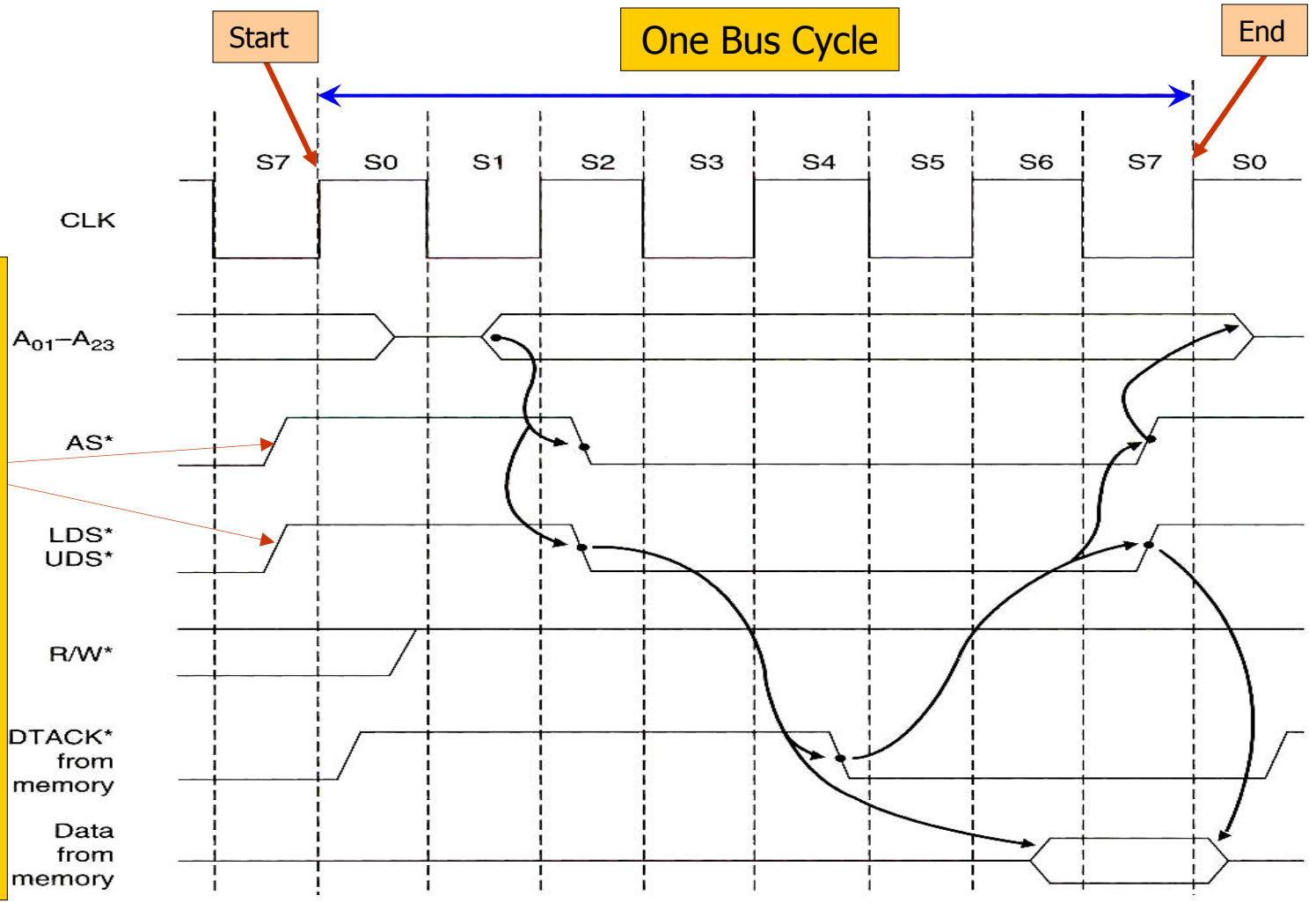
Block select signal, active low

Input to Dtack* generator when either RAM1 or RAM2 selected. This causes Dtack to be generated at the correct time for this memory device.

68k Asynchronous READ Cycle Timing Analysis

More Detailed 68000 Asynchronous Read Cycle Timing Analysis

- Each 68000 read cycle consists of 4 CPU clock cycles, comprising 8 internal states labelled S0-S7.
- Each read cycle starts with the leading edge of state S0 and ends with the trailing edge of state S7.
- Internally, the 68000 has a state machine to generate this timing each time an access takes place.

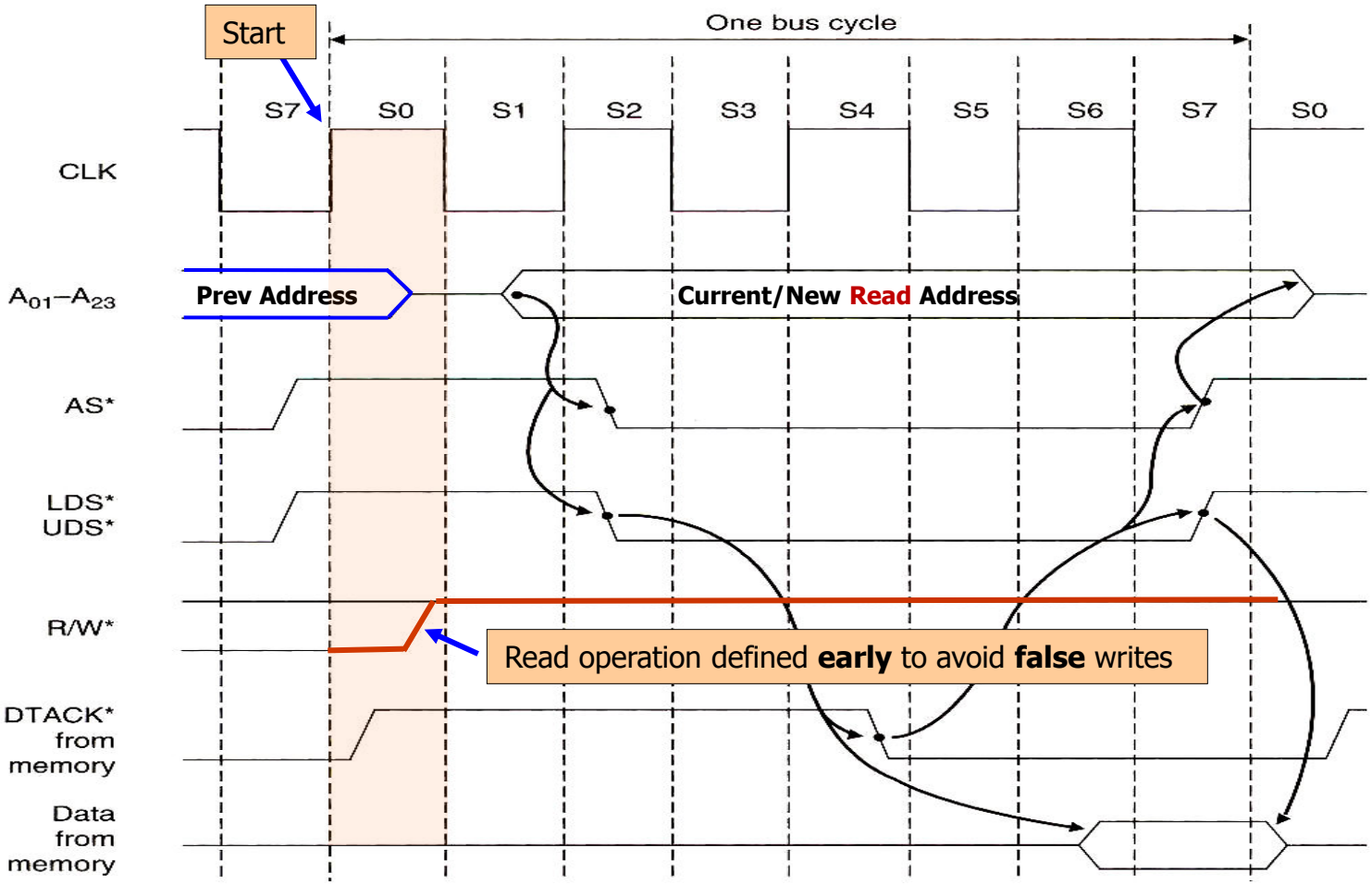


Note that AS* (driving memory $\overline{OE}$) and LDS*/UDS* (driving memory $\overline{CS}$) are all negated ***between*** bus cycles, giving us a guaranteed 'dead band' between access to two different chips, thus avoiding bus contention between unique memory chips that might be accessed in successive bus cycles

68k Asynchronous READ Cycle Timing Analysis

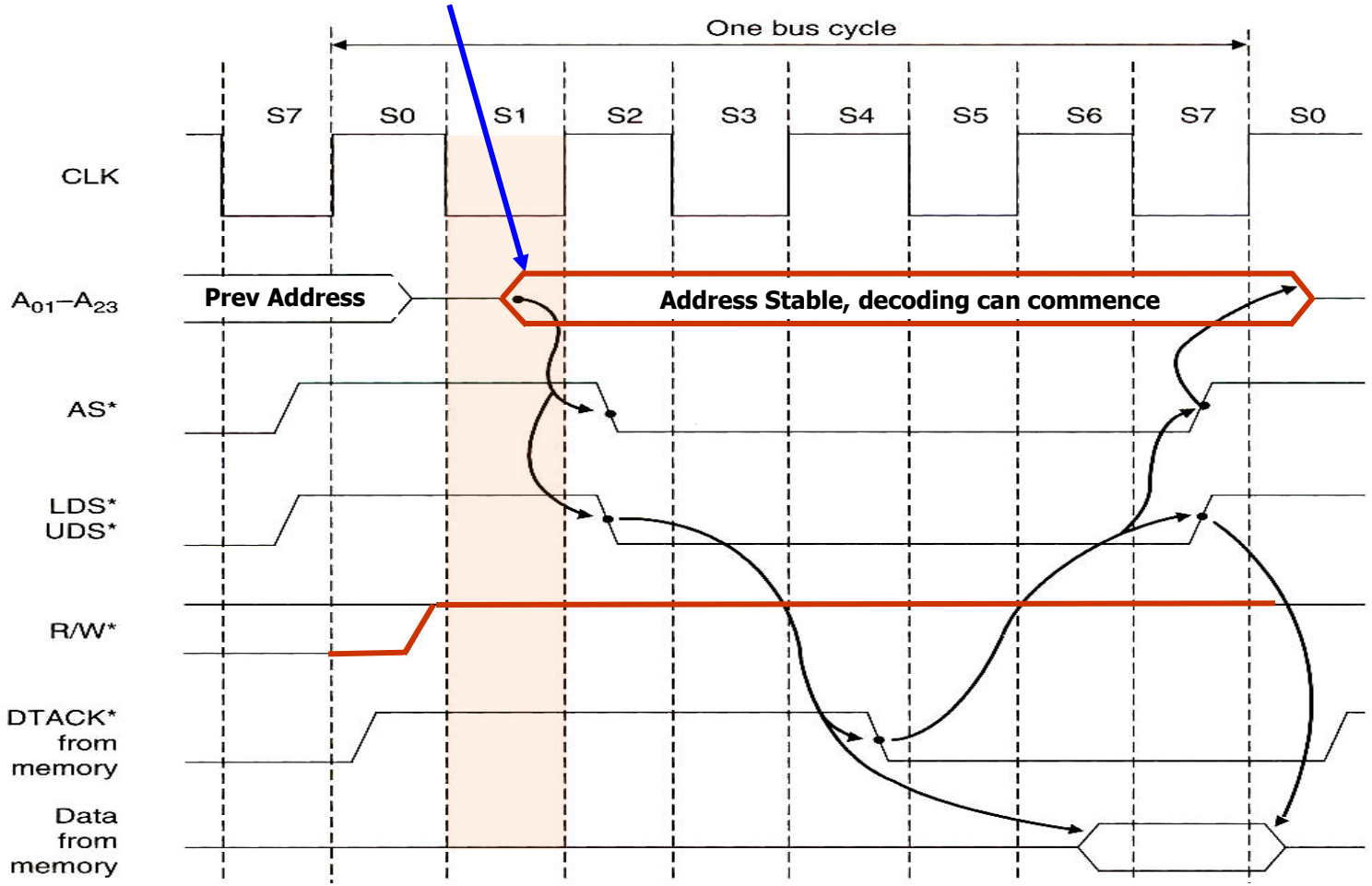
Detailed Description [**S0**] : Read cycle operation becomes defined

- At the start of a read cycle, all signals have been negated with the *exception* of the Address Bus and R/W* whose value carries over from previous cycle. Address bus tri-stated some time in [S0].
- For a read, R/W* taken high before new address and AS* are asserted to prevent accidental writing (the memory chip **only** receives a write signal when AS* = 0 **and** R/W* = 0, - *see circuit page 3*).



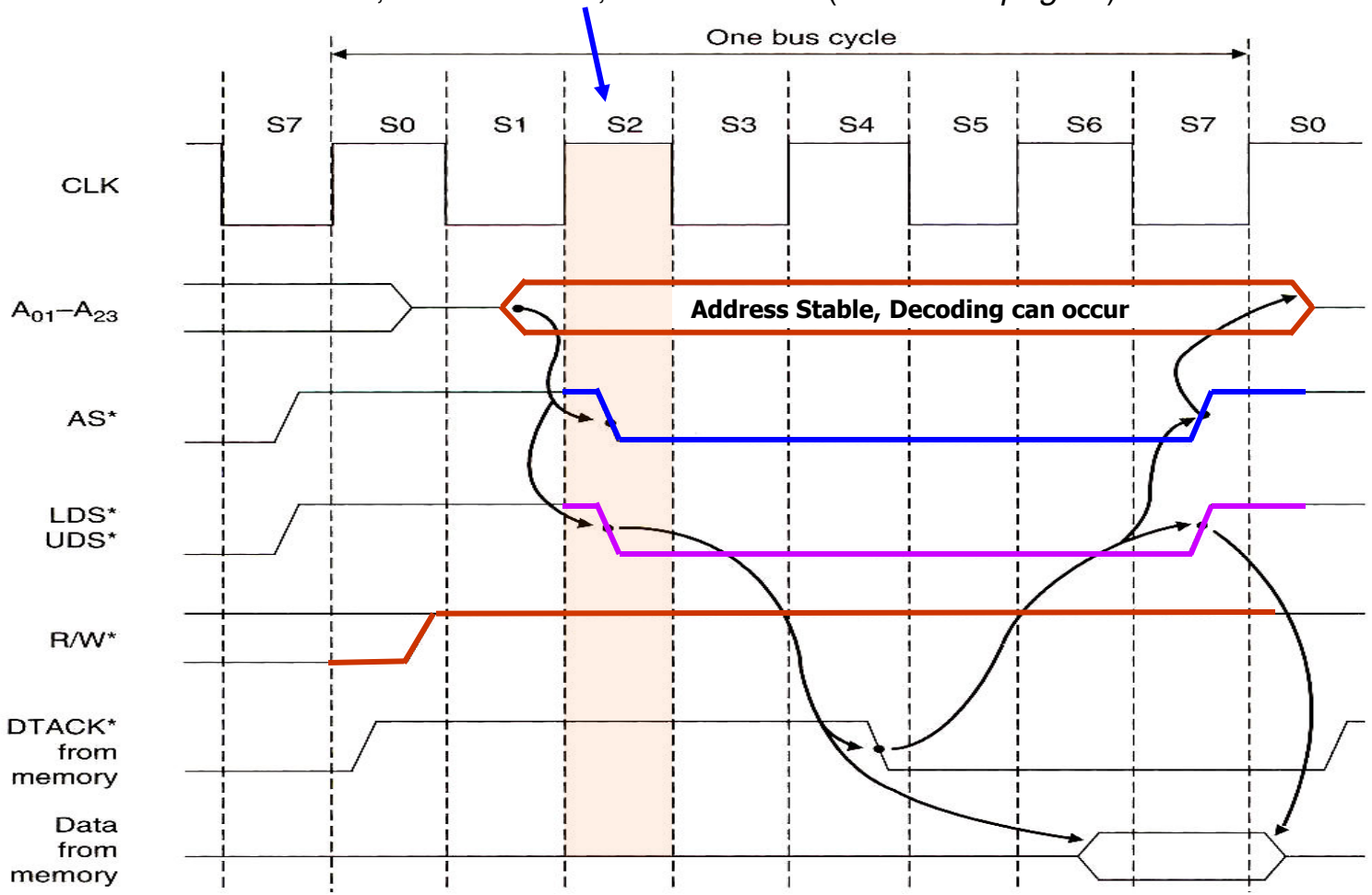
68k Asynchronous READ Cycle Timing Analysis

- [**S1**] : Address A1-A23 becomes valid some specified time during state [**S1**] and will remain so into state [**S0**] of next cycle (*long after the CPU has read the data from the memory in State [**S7**]*). This ensures the address is stable at the SRam and address decoder until after the data is read.
- External address decoders can begin to produce a Block Select signal for a memory bank during S1.
- The memory chips can also start to internally decode the CPU address, regardless of the state of their CS*, R/W* or OE* signals so we can start our critical 'access time' analysis from here.



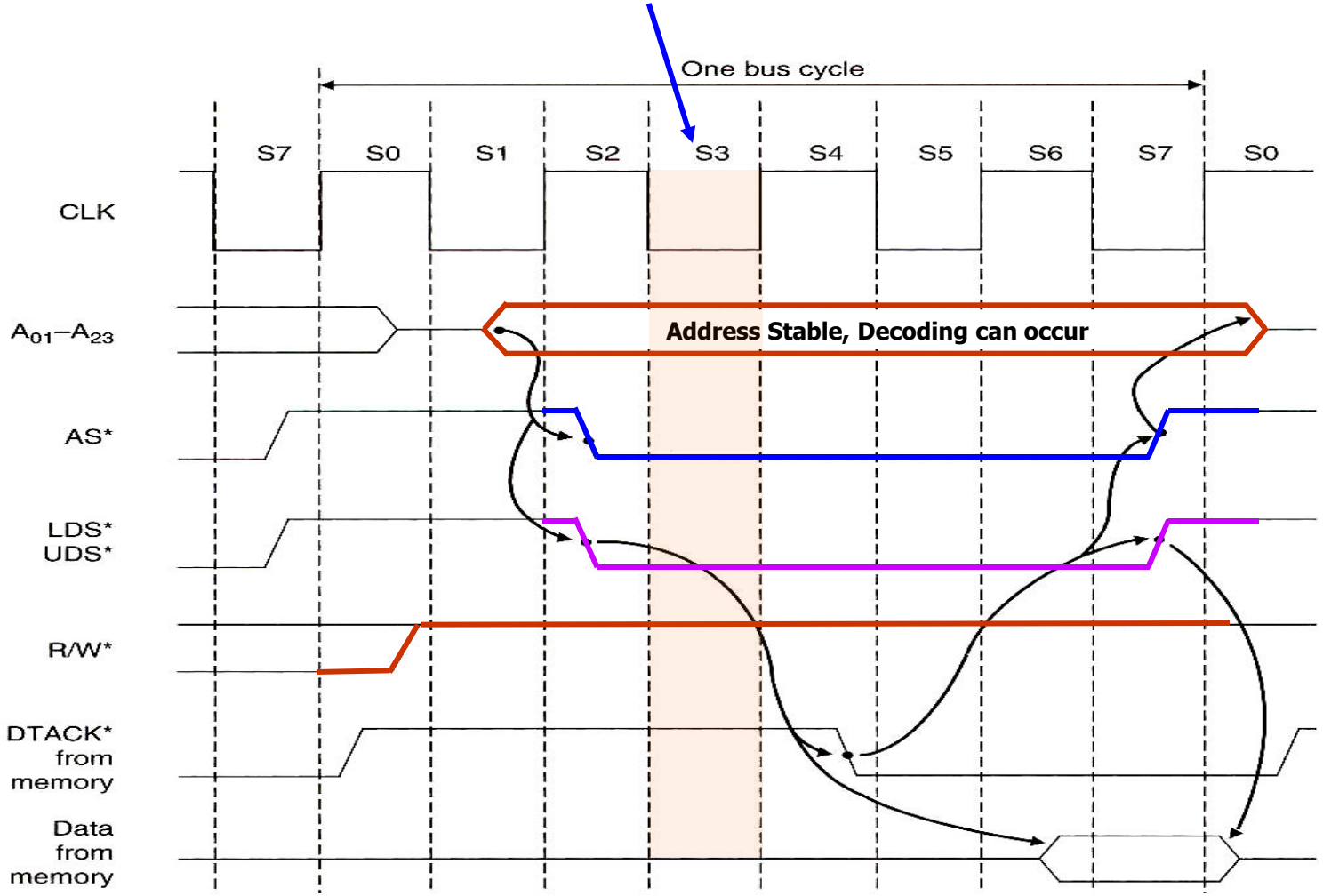
68k Asynchronous READ Cycle Timing Analysis

- [S2] :
  - AS* is asserted to indicate that the contents of the address bus are valid. AS* combined with the R/W*=1, provides the OE* and Read signal to the memory chip (*see circuit page 3*).
  - UDS* and LDS* are asserted to define data width, and are usually combined with the address decoder output to provide the CS* signal for the memory devices (*see circuit page 3*).
  - During state [S2] the memory chip should have **all** the signals it needs to provide the data, i.e. a Stable Address, defined R/W*, CS* and OE* (*see circuit page 3*).



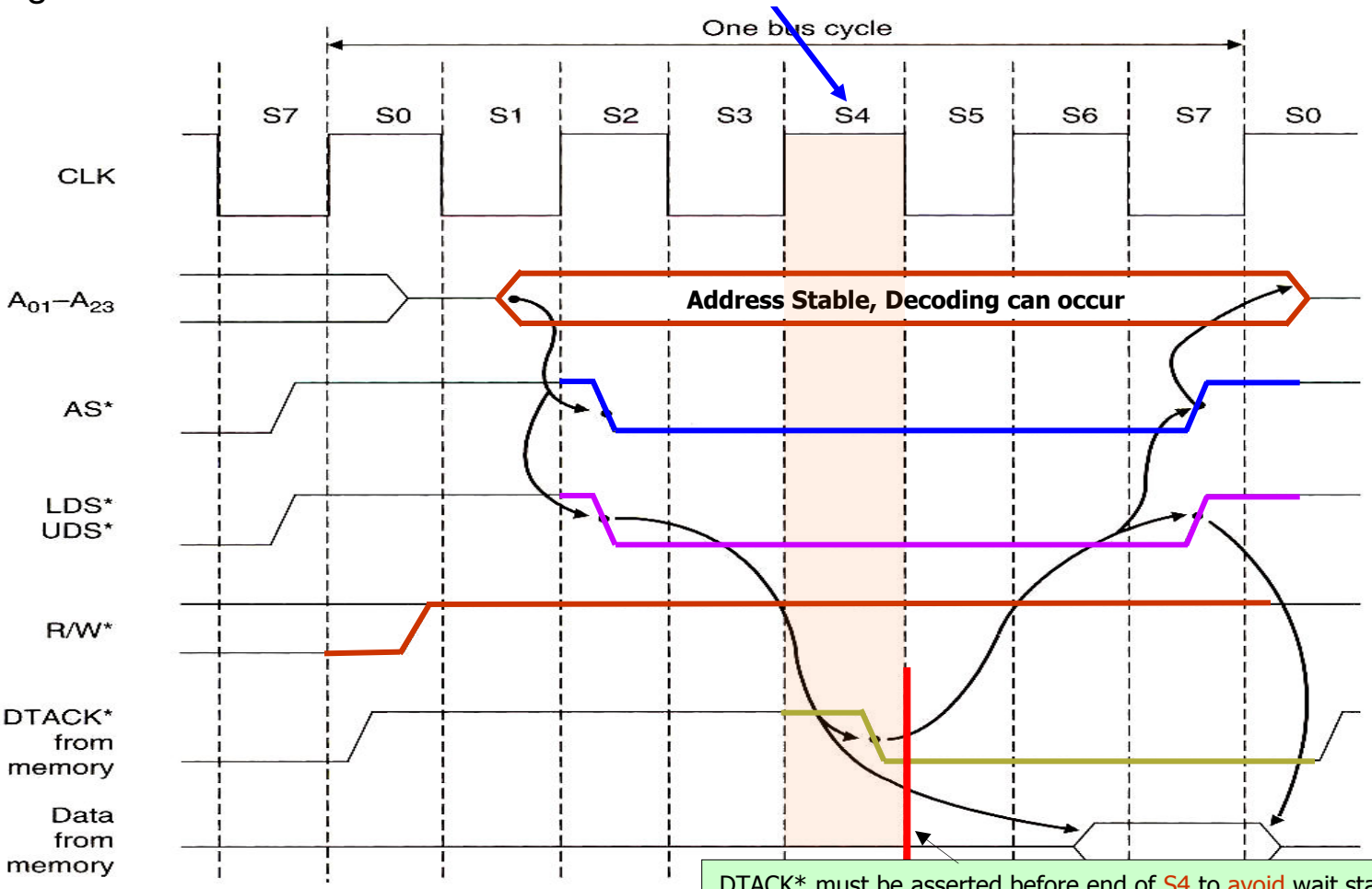
68k Asynchronous READ Cycle Timing Analysis

- [**S3**] : Nothing new happens. This is extra time that can be used by:-
  - The selected memory chip to continue internally decoding the address.
  - The Dtack Generator to determine if the memory chip is fast enough, or whether it will be necessary to introduce wait states. This will be crucial in state [S4].



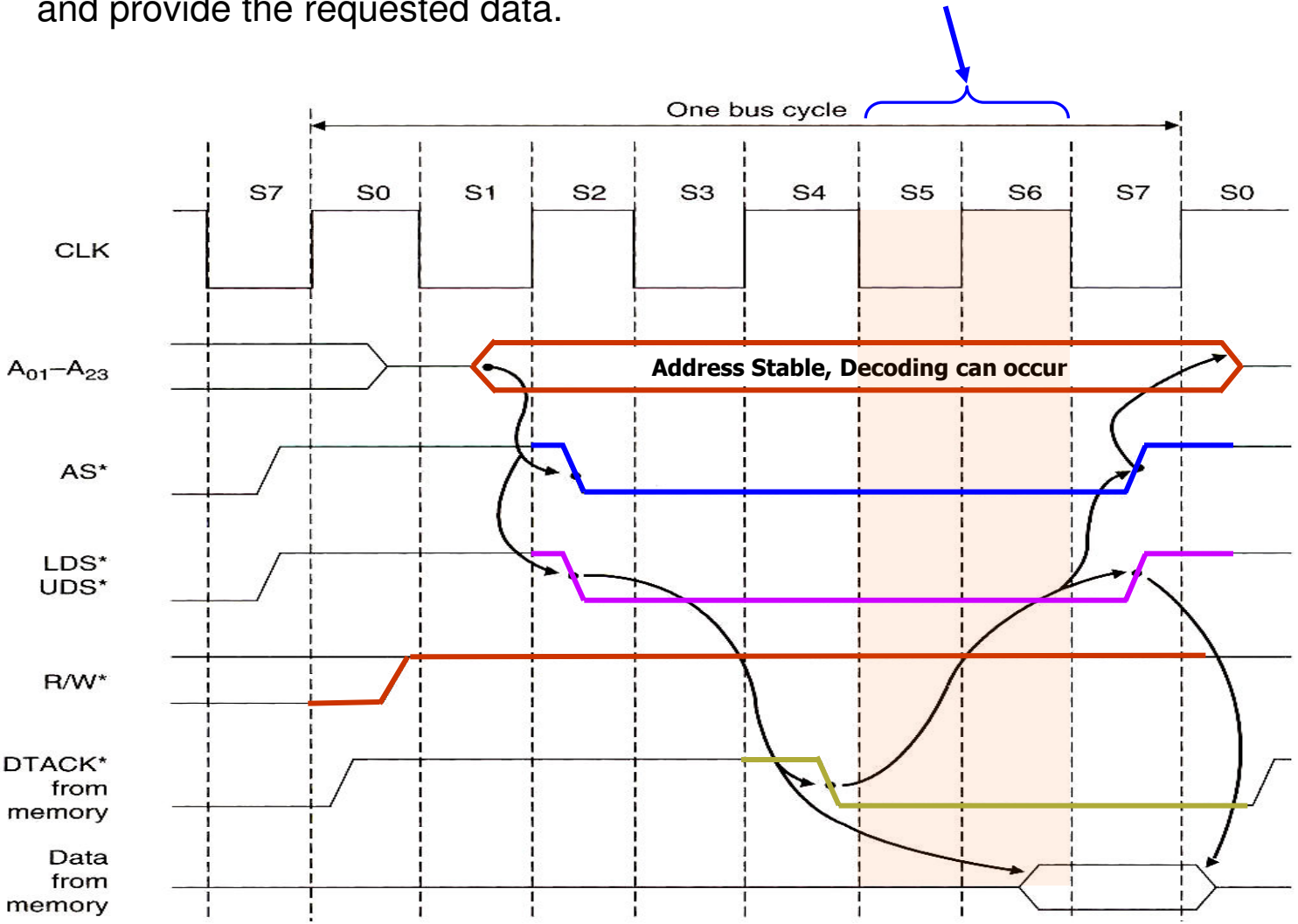
68k Asynchronous READ Cycle Timing Analysis

- [**S4**] : Acknowledgement (DTACK*) from memory system sampled by 68000
- Dtack Generator should assert DTACK* (*based on address*) before end of [S4].
- If DTACK* is **not** received by the 68k by the end of [S4] then 'wait' states (*additional, dummy [S4,S5] states will be introduced by 68k to extend the bus cycle beyond 8 states*).
- **Note:** DTACK* **can** be generated **at any time**, we do **not** have to wait until [S4] to generate it.



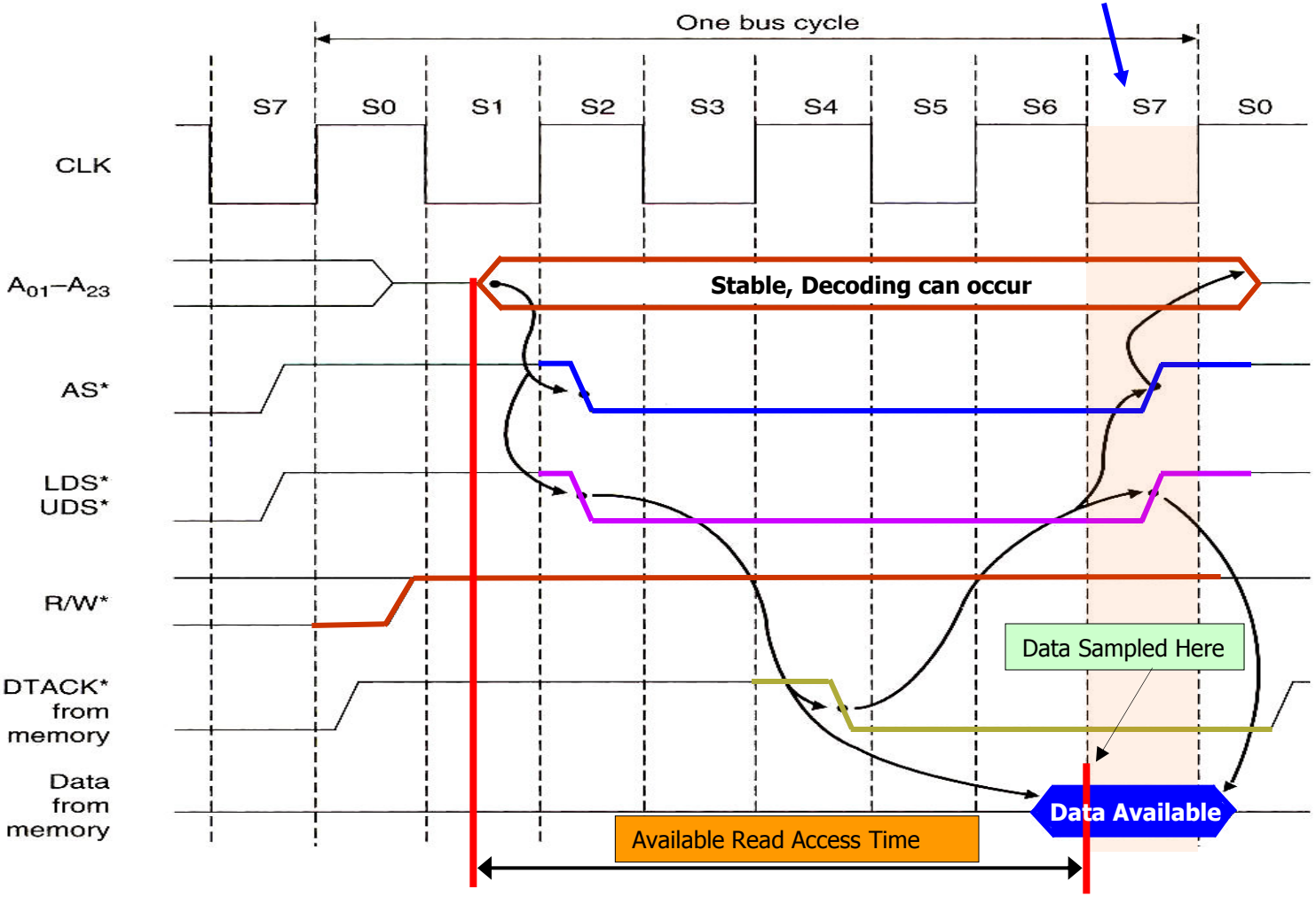
68k Asynchronous READ Cycle Timing Analysis

- States [**S5, S6**] : Nothing new happens.
- This idle time is to allow the selected memory device more time to decode the address and provide the requested data.



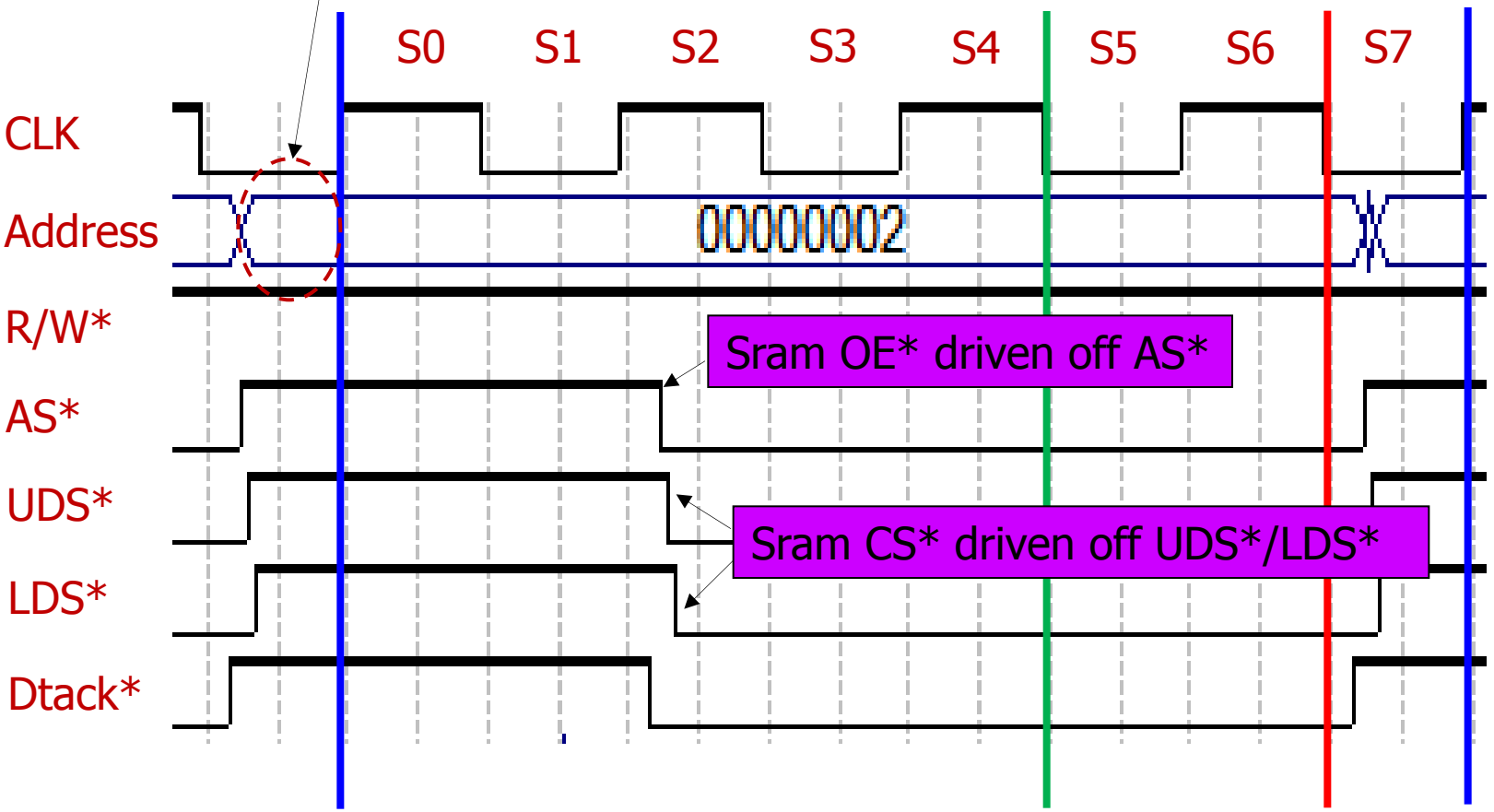
68k Asynchronous READ Cycle Timing Analysis

- [S7]: Data Read by 68000
  - Data latched internally by 68000 at start of [S7]. Memory **must** have supplied data by this time.
  - AS* Negated, turning off OE* on the Memory chips to prevent contention with other memory chips that might be accessed immediately following this bus cycle (*see circuit page 3*).
  - UDS* and LDS* strobes also negated, leading to the removal of CS* on memory. Data is then removed from the data bus by the memory. Dtack* is negated.



68k Asynchronous READ Cycle Timing Analysis

- Quartus simulations of Soft Core 68k Read cycle on DE2 almost identical to real 68k
- Address does not tri-state between accesses, so new address is issued ***before*** start of S0 (*which actually helps as memory chip can decode address internally for longer*)
- Notice how Dtack* can be asserted almost *immediately* (*driven simply off address*) and sampled at end of S4. Data latched/read by 68k at start of S7 (all signals at the Sram still value).



68k Detailed Timing Analysis

- Let's add some actual timing figures to this waveform, taken from the data sheet for the original 68k and do some simple calculations regarding the available access time of the chip to see if any particular chip is fast enough to be interfaced to the 68k with or without wait states.

- We will assume that no address or data buffers exist (*as per page 2 of this lecture*) and thus the propagation delays inherent in these two components can be ignored (*the effect of these real delays can easily be taken into account later if required*).

68k Detailed Timing Analysis

Here are the important timings to consider during a memory read

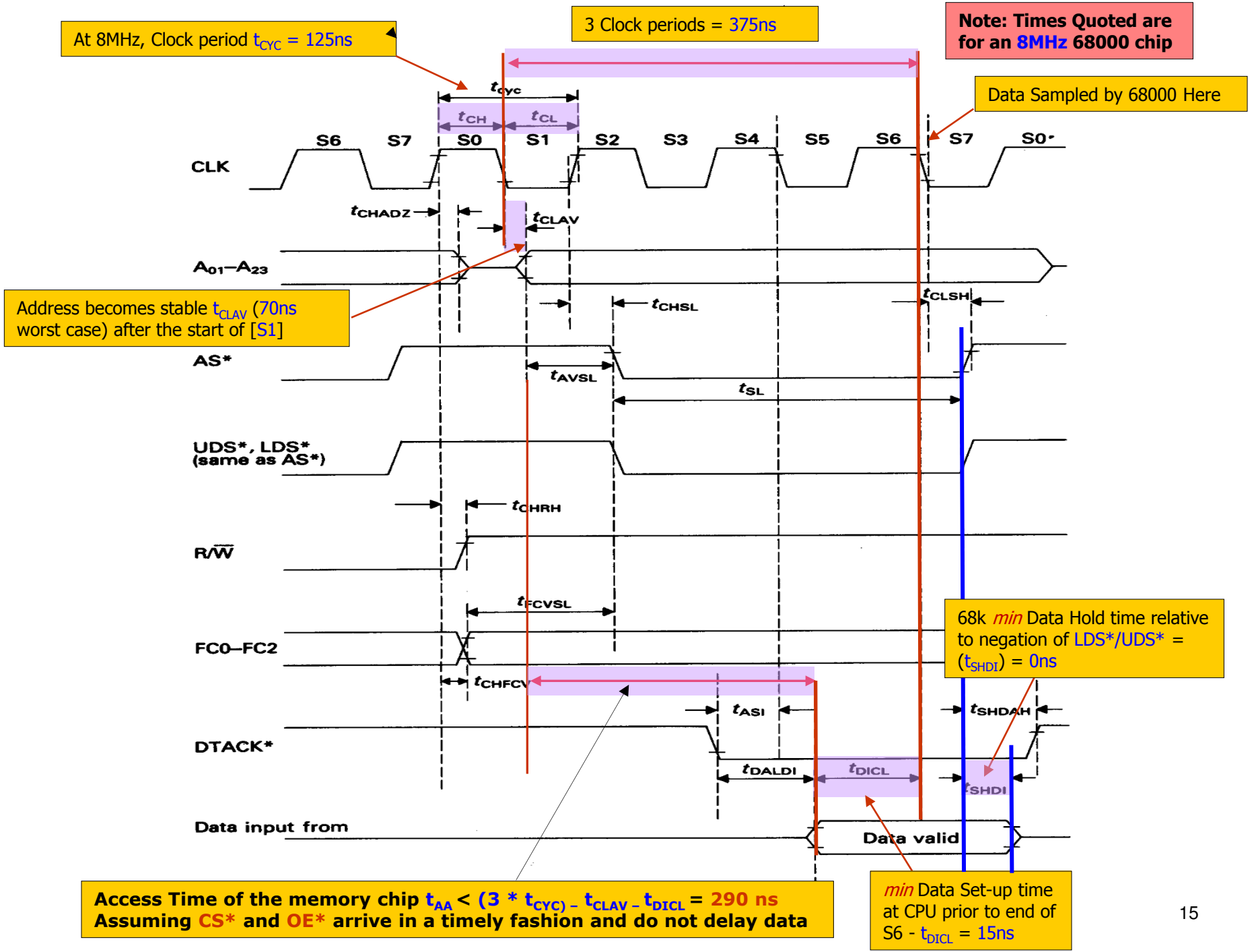
- Can our memory system decode the 68k address, provide the data back to the CPU and meet the data set up time at the 68000 data pins (*see next timing diagram*) before start of S7 ?

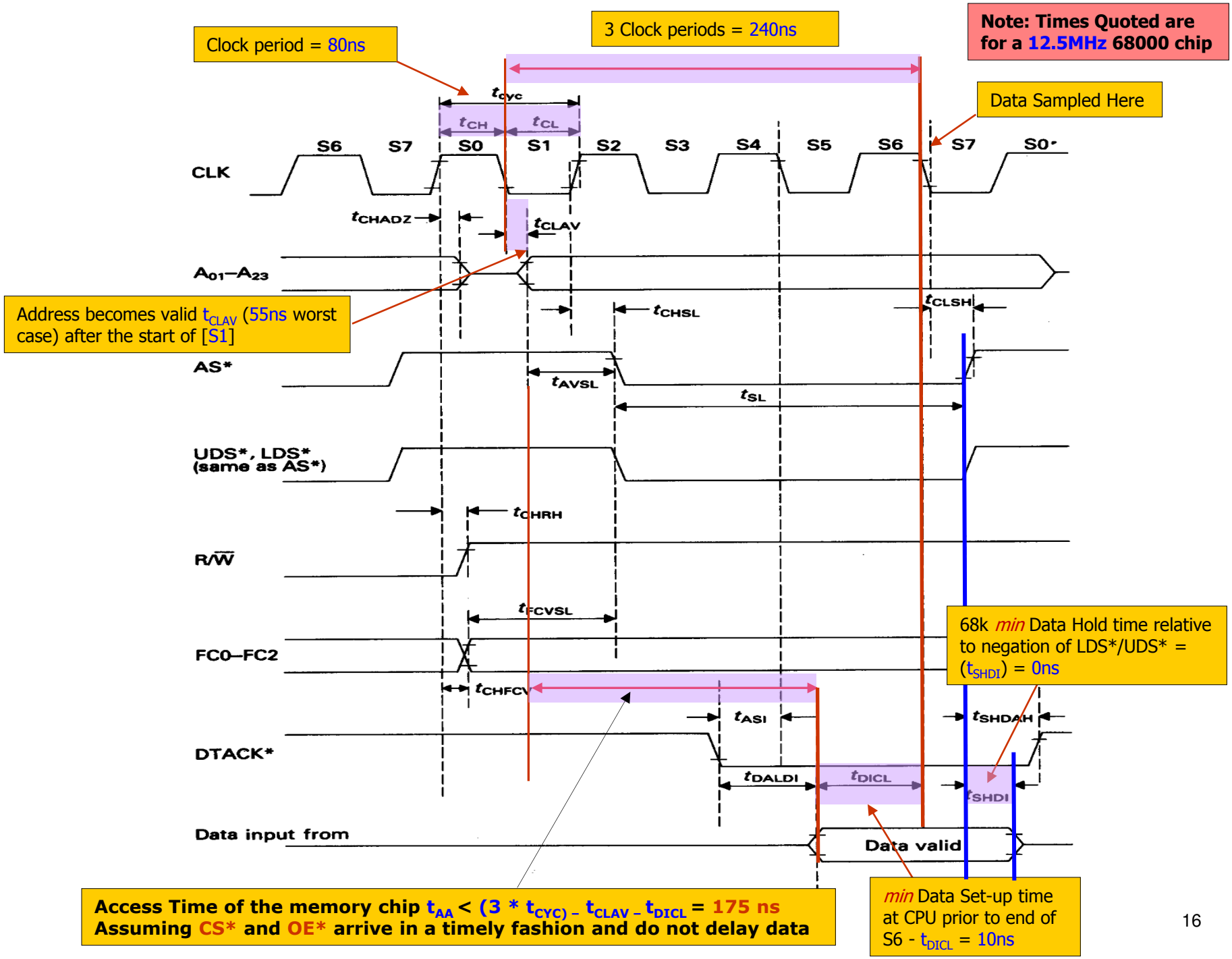
- Can we ensure that DTACK* is generated before the end of S4 (*see DTACK* generator circuit later*) ?

- Can we enable the memory "chip selects" (CS*) and turn on the memory chip output buffers (OE*) early enough so as not to delay the data coming from the memory – generally this last point is *relatively* easy.

  Referring back to Slide 3 we see that OE* is driven off AS* (which is asserted very early in the bus cycle [S2]) so turning on OE* will not be a problem.

  In addition, CS* is driven from the address decoder and the LDS*/UDS* signals, all of which are generated quite early in the bus cycle, so unless your decoder is ridiculously slow, the CS* signal will be dictated by the timing of LDS*/UDS* rather than the address itself and this should ensure it reaches the memory early enough so that is does not delay the enabling of the memory and delay the data back to the CPU.


At 8MHz, Clock period t_{CYC} = 125ns
3 Clock periods = 375ns
Note: Times Quoted are for an 8MHz 68000 chip
Data Sampled by 68000 Here
t_{CYC}
t_{CH}
t_{CL}
S6
S7
S0
S1
S2
S3
S4
S5
S6
S7
S0
CLK
t_{CHADZ}
t_{CLAV}
A_{01}–A_{23}
Address becomes stable t_{CLAV} (70ns worst case) after the start of [S1]
t_{CHSL}
t_{CLSH}
AS*
t_{AVSL}
t_{SL}
UDS*, LDS* (same as AS*)
t_{CHRH}
R/$\overline{W}$
t_{FCVSL}
FC0–FC2
t_{CHFCV}
t_{ASI}
t_{SHDAH}
DTACK*
t_{DALDI}
t_{DICL}
t_{SHDI}
Data input from
Data valid
68k *min* Data Hold time relative to negation of LDS*/UDS* = (t_{SHDI}) = 0ns
Access Time of the memory chip t_{AA} < (3 * t_{CYC}) – t_{CLAV} – t_{DICL} = 290 ns
Assuming CS* and OE* arrive in a timely fashion and do not delay data
min Data Set-up time at CPU prior to end of S6 - t_{DICL} = 15ns


Clock period = 80ns
3 Clock periods = 240ns
Note: Times Quoted are for a 12.5MHz 68000 chip
Data Sampled Here
t_{CYC}
t_{CH}
t_{CL}
S6
S7
S0
S1
S2
S3
S4
S5
S6
S7
S0
CLK
t_{CHADZ}
t_{CLAV}
A_{01}–A_{23}
Address becomes valid t_{CLAV} (55ns worst case) after the start of [S1]
t_{CHSL}
t_{CLSH}
AS*
t_{AVSL}
t_{SL}
UDS*, LDS* (same as AS*)
t_{CHRH}
R/$\overline{W}$
t_{FCVSL}
FC0–FC2
t_{CHFCV}
68k *min* Data Hold time relative to negation of LDS*/UDS* = (t_{SHDI}) = 0ns
t_{ASI}
t_{SHDAH}
DTACK*
t_{DALDI}
t_{DICL}
t_{SHDI}
Data input from
Data valid
Access Time of the memory chip t_{AA} < (3 * t_{CYC}) – t_{CLAV} – t_{DICL} = 175 ns
Assuming CS* and OE* arrive in a timely fashion and do not delay data
min Data Set-up time at CPU prior to end of S6 - t_{DICL} = 10ns

68k Read Cycle Timings (Motorola Data Sheet)

Note: Times Quoted are for an 8MHz 68000 chip

PARAMETER NAME	SYMBOL	MINIMUM	MAXIMUM
Clock period	t_{cyc}	125	250
Clock width (low)	t_{CL}	55	125
Clock width (high)	t_{CH}	55	125
Clock high to address bus high impedance	t_{CHADZ}		80
Clock low to address valid	t_{CLAV}		70
Address valid to AS* low	t_{AVSL}	30	
Clock high to AS*, DS* low	t_{CHSL}	0	60
Clock low to AS*, DS* high	t_{CLSH}		70
AS*, DS* with low	t_{SL}	240	
Clock high to R/$\overline{W}$ high	t_{CHRH}	0	70
Clock high to FC valid	t_{CHFCV}		70
FC valid to AS* low	t_{FCVSL}	60	
Asynchronous input DTACK* setup time	t_{ASI}	20	
AS*, DS* high to DTACK* high	t_{SHDAH}	0	245
Data in to clock low setup-time	t_{DICL}	15	
DS* high to data invalid (data hold-time)	t_{SHDI}	0	
DTACK* low to data in setup-time	t_{DALDI}		90

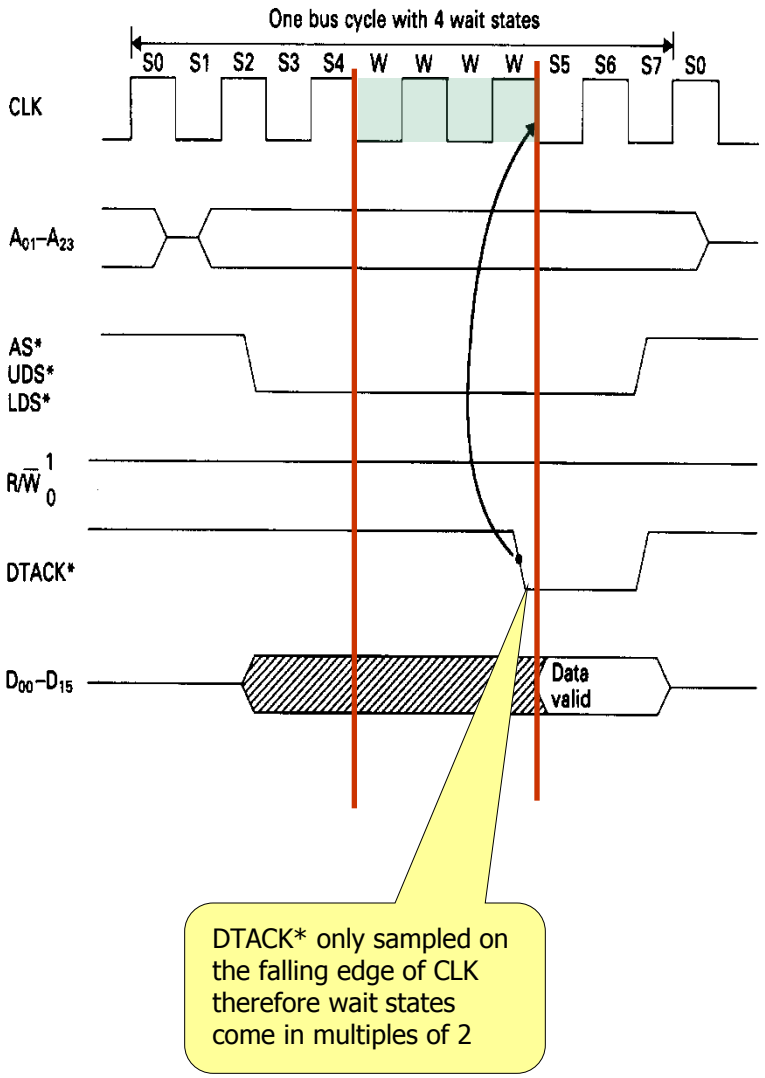
68k Asynchronous Bus Cycles with Wait States

Inserting Wait States

- The 68k can extend a bus cycle **indefinitely**, to cater for slow memory systems that are not able to supply their data before end of [S6].

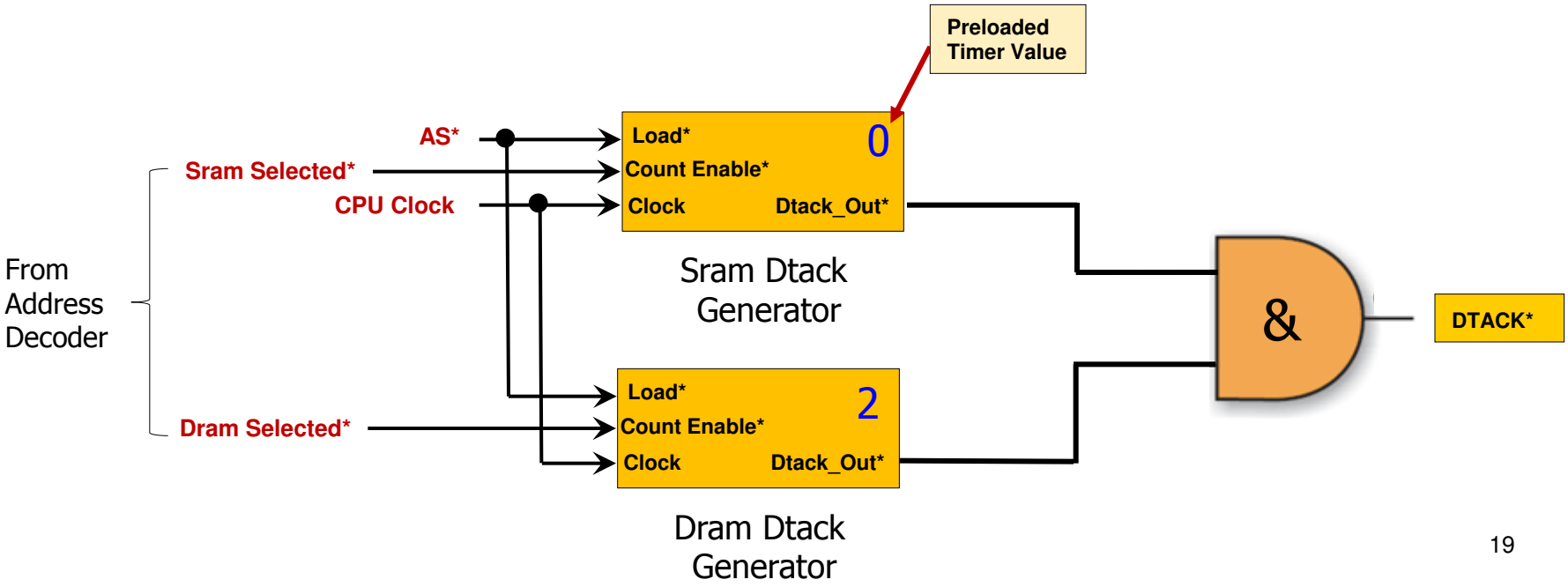
Description

- If DTACK* is **not** asserted prior to the end of state [S4], the 68000 will insert **wait states** into the bus cycle between states [S4] and [S5]. The timing diagram opposite illustrates this.
- Note DTACK* signal is only sampled by the 68000 on the falling edge of the CLK, thus the minimum number of wait states that can be introduced is 2, i.e. 1 full clock cycle or 2 wait states.
- When a suitable number of wait states have been inserted and the slow memory has been given the appropriate extended time to respond it must then generate DTACK* to terminate the bus cycle.
- In reality it is always the address decoder logic, rather than the memory chip itself, that supplies DTACK*.
- Thus DTACK* is usually derived from some delay logic associated with a particular address map.



A Simple and Flexible 68k DTACK* Generator Circuit

- There are many ways to make a DTACK generator, but they all involve some kind of timer that will produce a DTACK after a pre-defined number of clock cycles.
- Each device or family of devices with the same access speed could have it's own DTACK generator.
- For fast devices the DTACK can be generated straight away. For slower devices, the access to the device could enable a timer to count down from some predefined count value and produce a DTACK out
- The counter clock would be driven off the CPU clock, so can be used to time the various CPU states and if necessary delay beyond state S4 if the device is slow.
- The active low Dtacks from each of the various timers in the system can be **ANDED'ed** together to produce the eventual active low DTACK* back to the CPU.
- The timer can be enabled (*to count down to zero*) when it detects a valid address being issued for the device (i.e. via the address decoder output being active and AS* being '0') and reset back to some initial count value at then end of a bus cycle when AS* goes high. There are many solutions.



68k Asynchronous Sram Timing Diagrams – Write Cycle Operation

- For a write to take place the Data must be presented **to** the memory chip and an address presented and $\overline{WE}$ and $\overline{CE}$ taken low on the memory chip.

- An actual Sram **write** operation begins as **soon** as its $\overline{WE}$ and $\overline{CE}$ lines are both driven low, so the address must be valid at this point to avoid false writes (both $\overline{WE}$ and $\overline{CE}$ driven off AS*). $\overline{OE}$ is ignored during writes.

- The write operation is terminated and data is finally stored when either $\overline{WE}$ or $\overline{CE}$ is negated, (*which ever comes first*) at the Sram chip.

- Address **must** be valid throughout the write cycle, to avoid false writes thus a write to the memory **must** be terminated when AS* goes high.

- This is why the Sram memory R/W* signal is gated with the 68k's AS* signal.

- Note the 16 bit wide Sram chip on the DE2 has separate $\overline{UB}$ and $\overline{LB}$ signals for upper byte and lower byte enable, these act just like additional chip enable signals for each half of the 16 bit data bus, so each half will only respond when its $\overline{UB}$ or $\overline{LB}$ signal respectively is low

68k Asynchronous Sram Timing Diagrams – Write Cycle Operation

- In designing our memory system, we must be careful **not** to simultaneously assert $\overline{WE}$ and $\overline{CE}$ at the Sram chip until the address is stable to avoid writing to an incorrect location.

- There is also an address *set up time* at the memory chip given by the parameter t_{SA} to be met prior, to initiating a write operation and an *address hold time* t_{HA} after write termination.

- All memory chips have a minimum write time, defined by the parameter t_{PWE} i.e. a minimum period for which both $\overline{WE}$ and $\overline{CE}$ are active.

- We shall see from the timing diagram next, that the memory write operation is in principle dictated by the assertion and negation of the 68000 signals $\overline{LDS}/\overline{UDS}$ and/or $\overline{AS}$ (*which control the memory's* $\overline{CE}$ *and* $R/\overline{W}$ *signal, see circuit page 3*) rather than the CPU's $R/\overline{W}$ (*which is initiated earlier and removed later*).

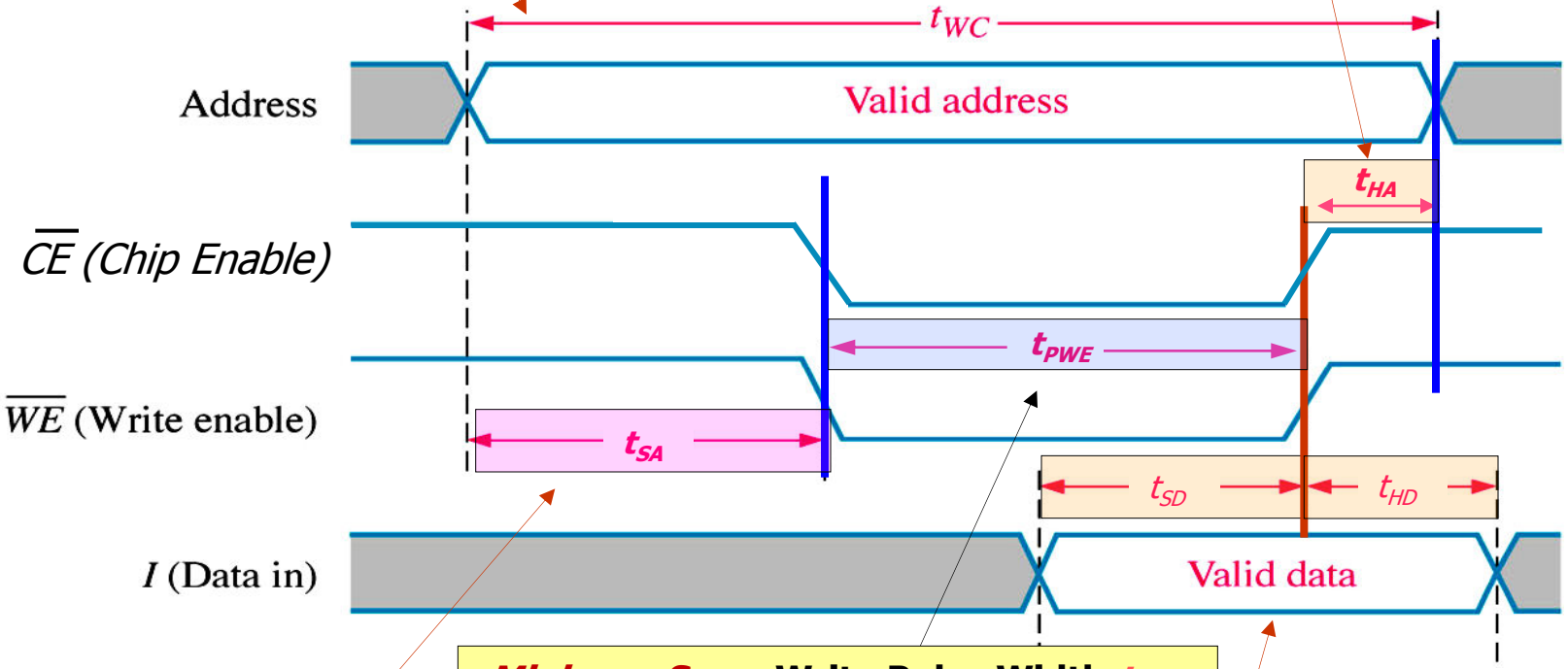
- Finally there is a data set-up time t_{SD} and data hold time t_{HD} that must be met at the memory data pins associated with write termination.

- These are shown in the timing diagram on the next page.

Asynchronous Static Memory Timing Diagrams – Write Cycle

Sram Write cycle time t_{WC} is the *minimum* time for a guaranteed write operations

***Minimum* Sram Address hold time t_{HA} following termination of write cycle ($\overline{CE}$ or $\overline{WE}$ negated)**

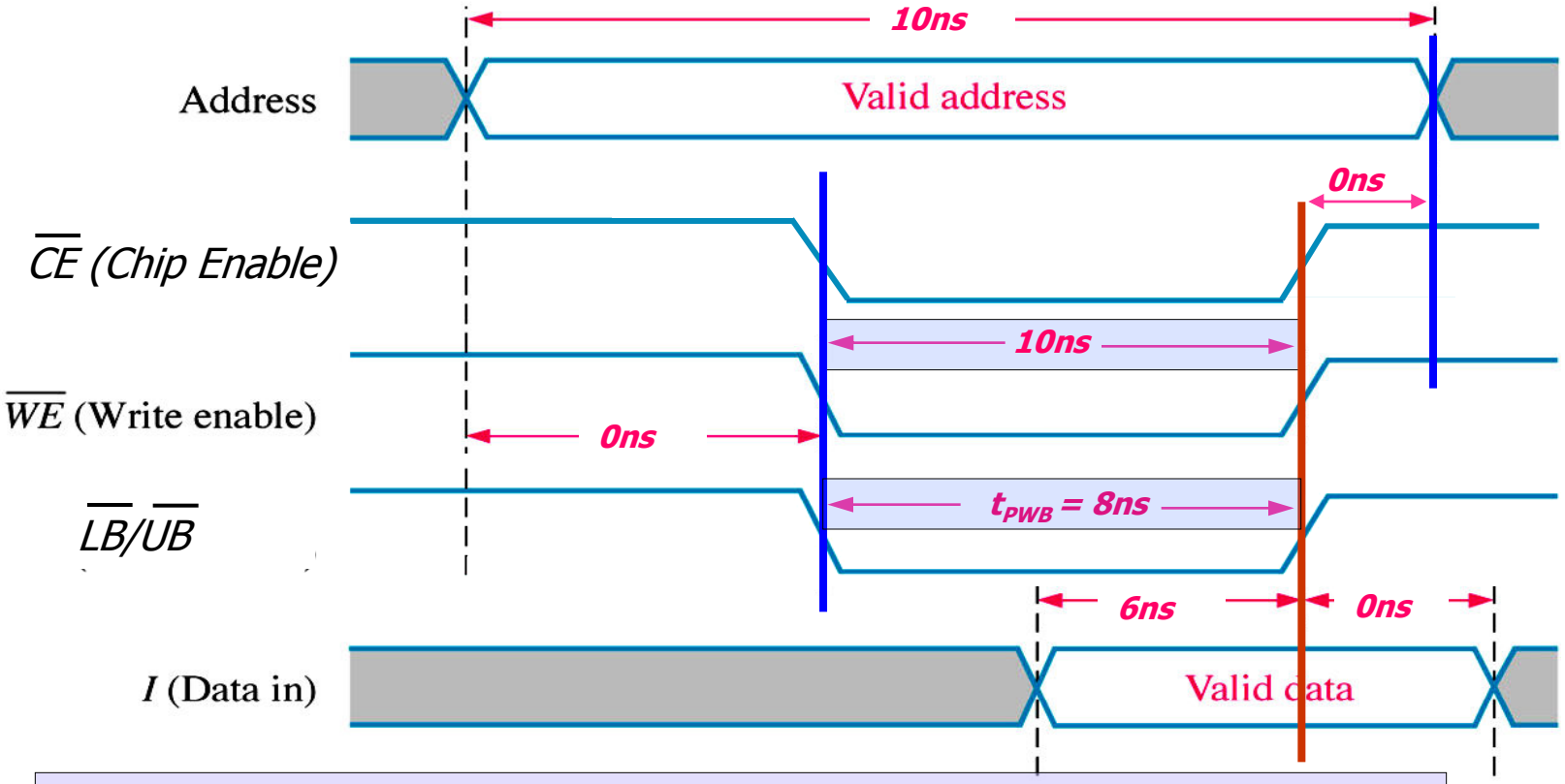


***Minimum* Sram Write Pulse Width t_{PWE}**

***Minimum* Address set up time at Sram t_{SA} must be met before $\overline{WE}$ and $\overline{CE}$ simultaneously become active**

Sram Data in must meet *minimum* set-up t_{SD} and hold t_{HD} times at memory chip relative to $\overline{WE}$/$\overline{CE}$ whichever is removed first

ISSI 256k x 16 Static Memory Timing Diagrams – Write Cycle



Min Timing Parameter	Symbol	ISSI 256k x 16
Write Cycle Time	t_{WC}	10 ns (*min*)
Address Set-up Time	t_{SA}	0 ns (*min*)
Address Hold Time	t_{HA}	0 ns (*min*)
Data Set-up Time	t_{SD}	6 ns (*min*)
Data Hold Time	t_{HD}	0 ns (*min*)
Write Signal Pulse Width	t_{PWE}	10 ns (*min*)
Byte Select to end of Write	t_{PWB}	8 ns (*min*)

Detailed 68000 Write Cycle Timing Diagram

- The 68k's write cycle timing is very similar to that of reading, except of course that the 68k is now driving data onto the data bus and the memory chip is expected to store it.

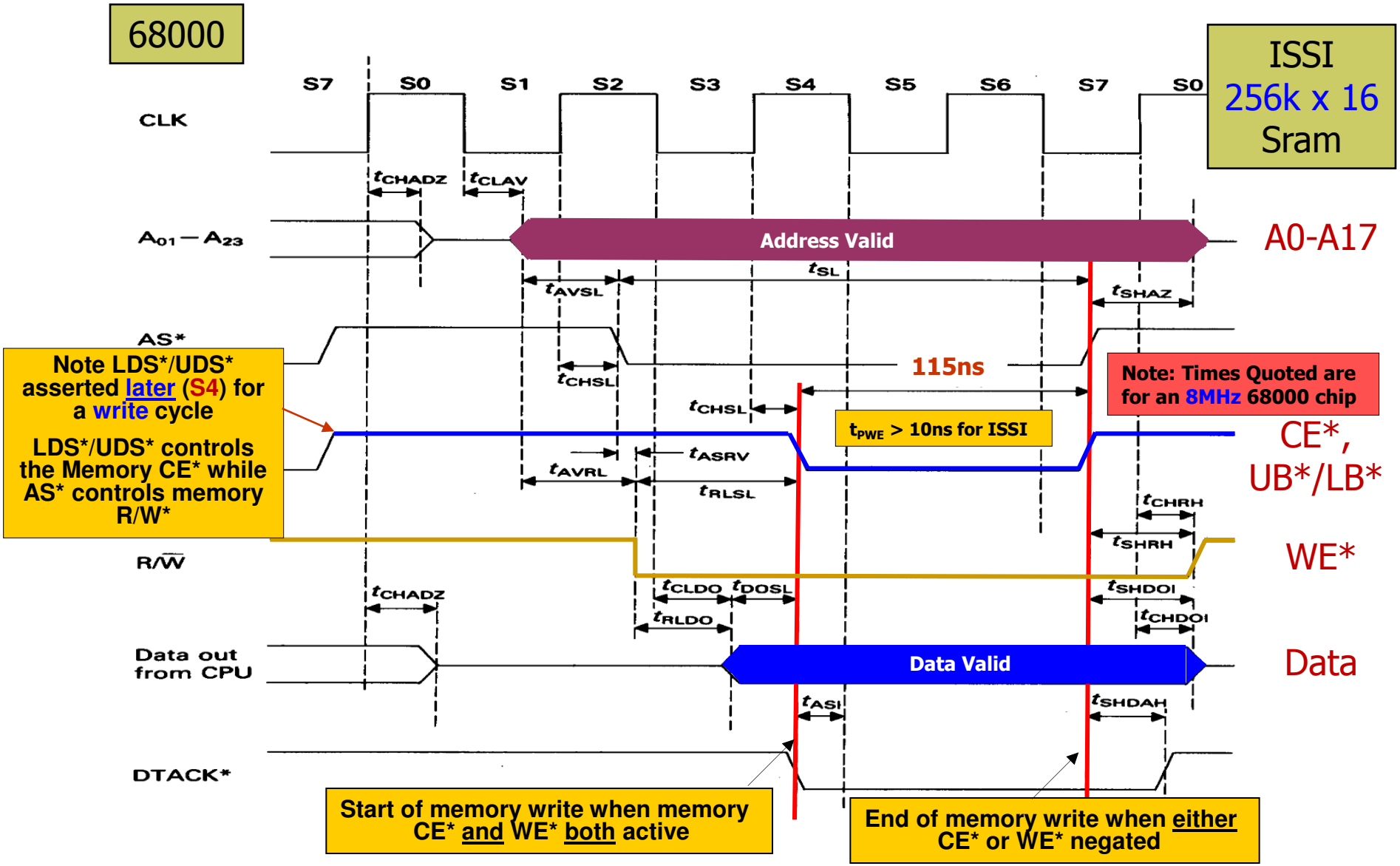
- There are two other minor differences:-

  - R/W* obviously is **low** during a write cycle. This takes place during state (S2) to ensure the address is stable prior to any kind of write signal being issued.
  - UDS*/LDS* are asserted **later** in a write bus cycle state (S4) compared to a read bus cycle state (S2), these two signals drive CS* on the memory chip to prevent the false writing of data to the wrong address (see page 3).

- Referring to the original circuit diagram (page 3), the **key signals** here are **UDS*/LDS*** not **R/W*.** That is, the memory chip write cycle begins and ends with the assertion and negation of **LDS*/UDS*** rather than **R/W***. Thus memory write analysis must be measured relative to **LDS*/UDS***.

- Let's check that the design on Page 3 does not violate any of the memory chips critical timings

Detailed 68000 Write Cycle Timing Diagram for ISSI Sram

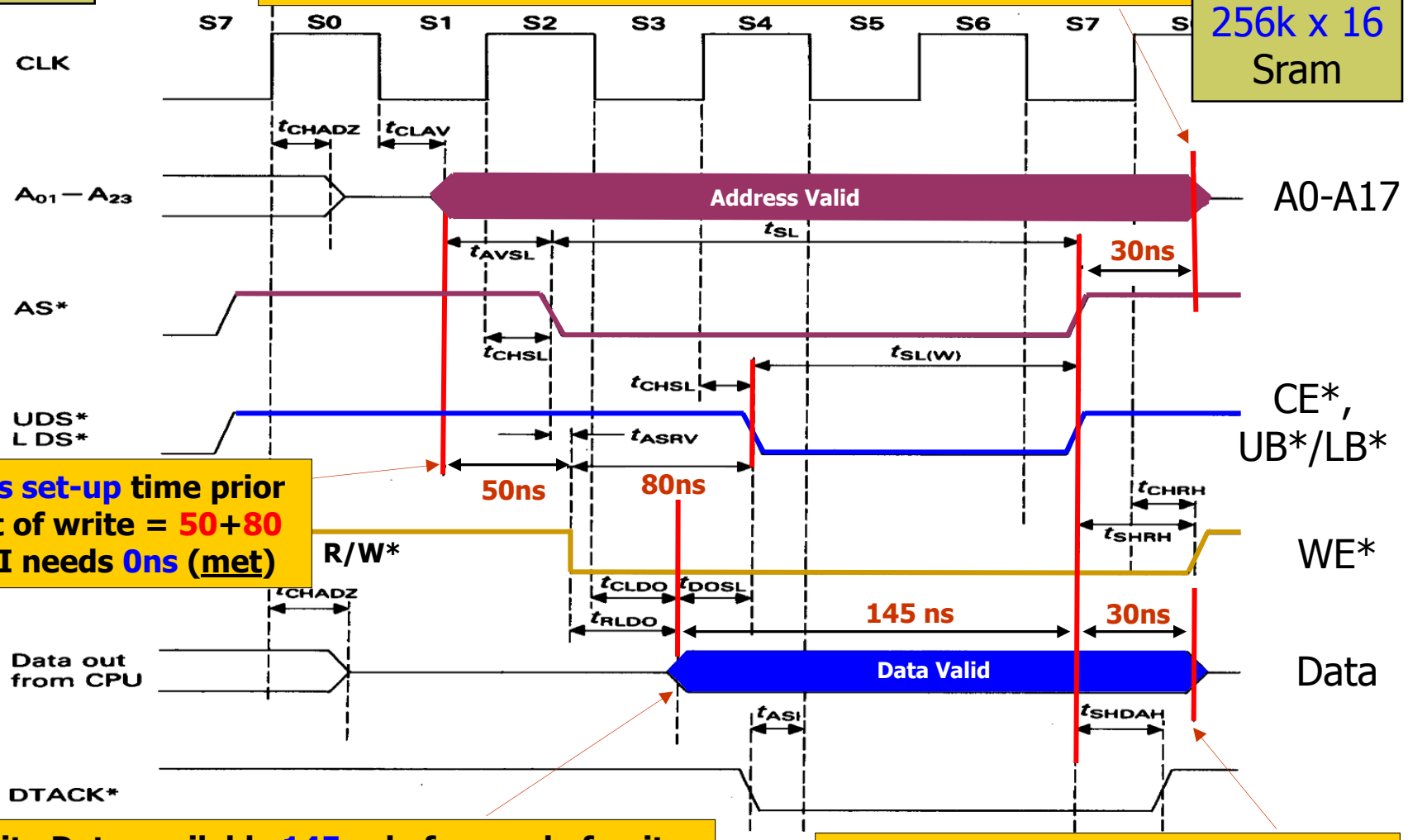


Detailed 68000 Write Cycle Timing Diagram

68000

Address hold time relative to the end of write (LDS*/UDS*)
68000 = 30ns min, ISSI chip needs 0ns (timing met)

ISSI
256k x 16
Sram



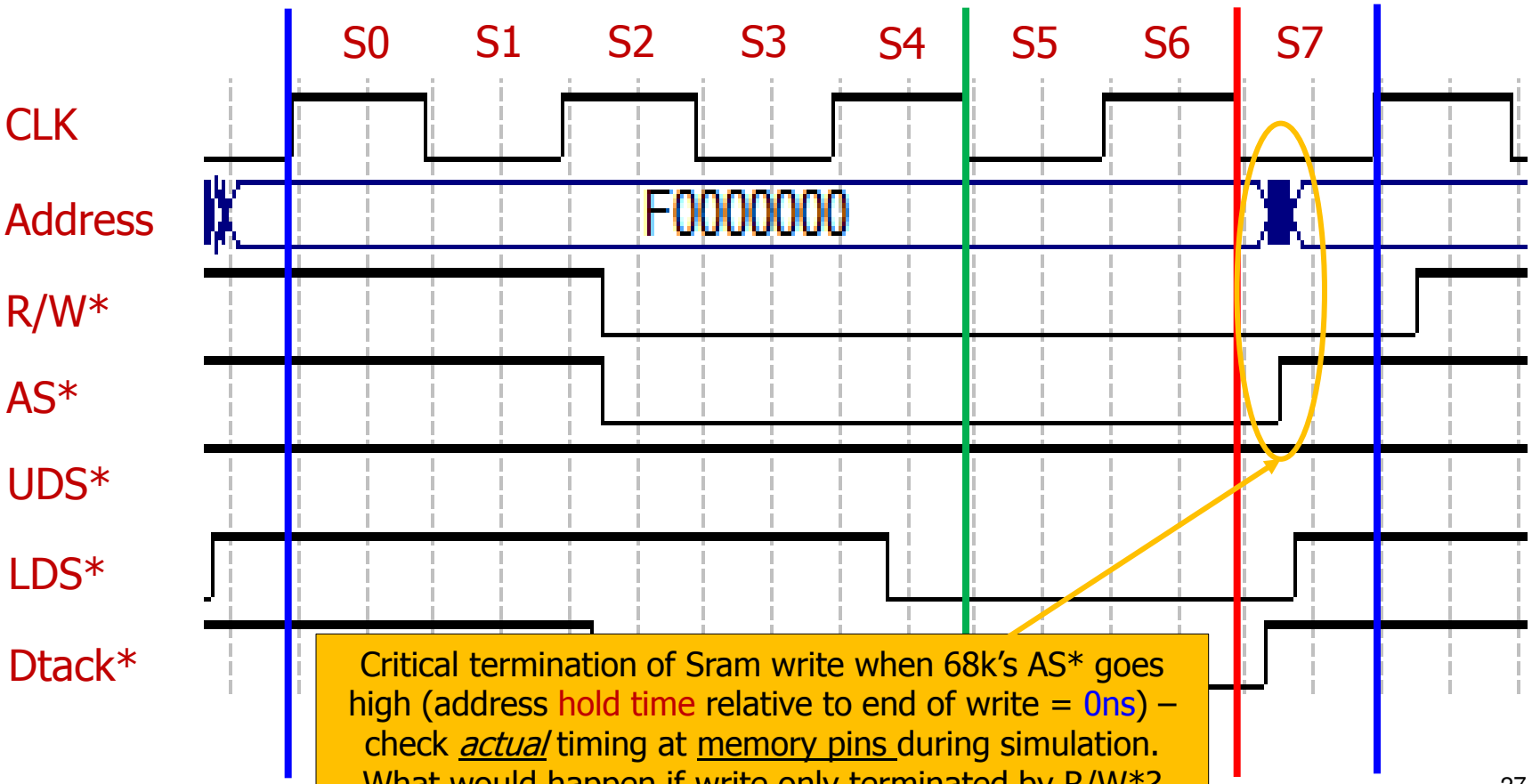
Address set-up time prior to start of write = 50+80 ns. ISSI needs 0ns (met)

68k Write Data available 145ns before end of write. ISSI data setup time = 6ns (timing met)

Data held by 68000 for 30ns min ISSI needs 0ns (timing met)

Quartus Soft Core 68000 Write Cycle Timing Diagram

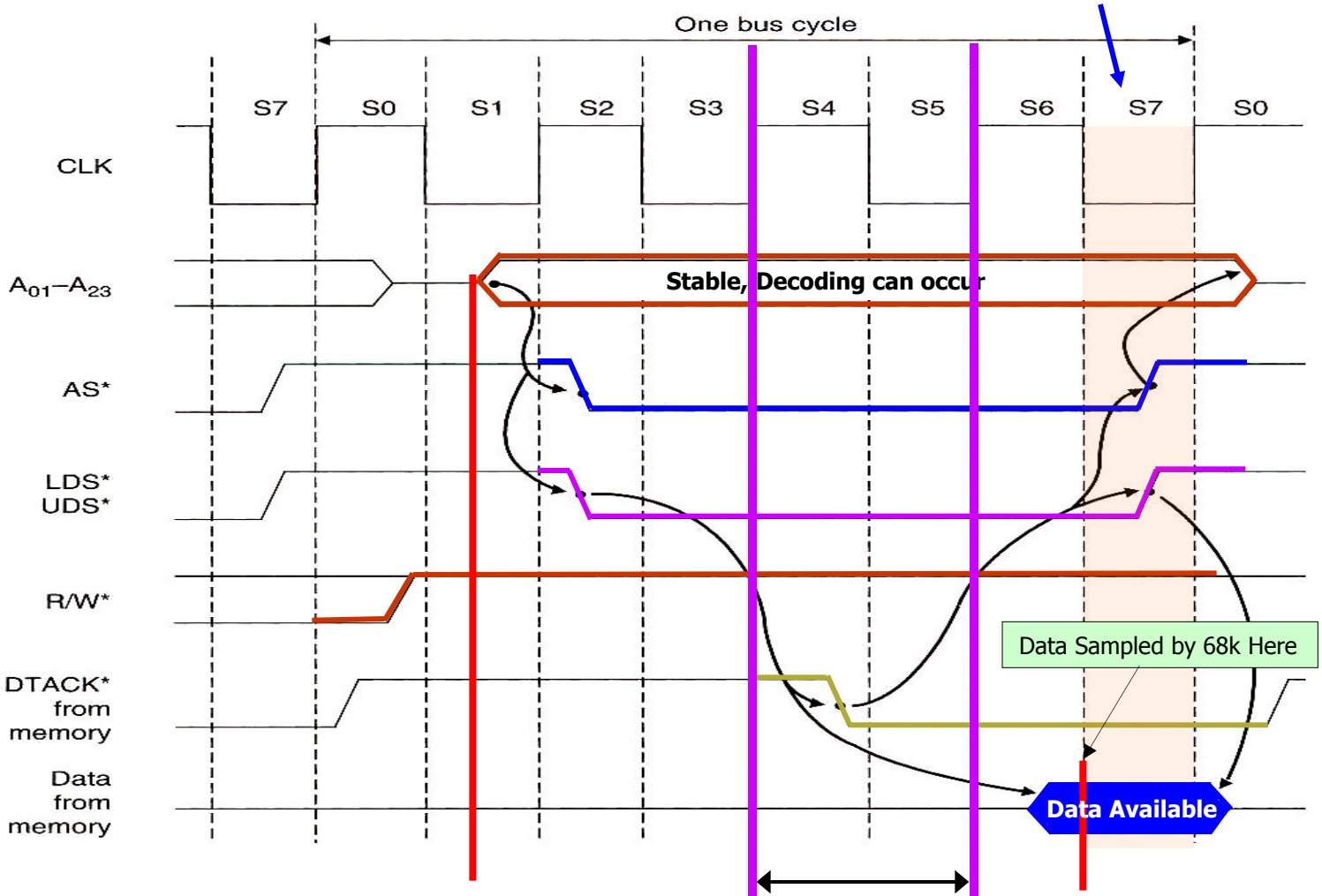
- Quartus Simulation of soft core 68k write cycle.
- Note R/W* defined in S2 right through to S0 of next bus cycle.
- LDS* (*as this is for an 8 bit write*) goes low late in S4 and defines width of memory write operation (*see circuit page 3*)



68000 Write Cycle Timings

PARAMETER NAME	SYMBOL	8 MHz		12.5 MHz	
		MIN.	MAX.	MIN.	MAX.
Clock period	t_{cyc}	125	250	80	250
Clock high to data and address bus high impedance	t_{CHADZ}		80		60
Clock low to address valid	t_{CLAV}		70		55
Clock high to AS*, DS* low	t_{CHSL}	0	60	0	55
Address valid to AS* low	t_{AVSL}	30		0	
Clock low to AS*, DS* high	t_{CLSH}		70		50
AS*, DS* high, to address invalid	t_{SHAZ}	30		10	
R/$\overline{W}$ low to DS* low (write)	t_{RLSL}	80		30	
AS* width low	t_{SL}	240		160	
DS* width low (write cycle)	$t_{SL(w)}$	115		80	
Address valid to R/$\overline{W}$ low	t_{AVRL}	20		0	
AS* low to R/$\overline{W}$ valid	t_{ASRV}		20		20
Clock high to R/$\overline{W}$ high	t_{CHRH}	0	70	0	60
AS*, DS* high to R/$\overline{W}$ high	t_{SHRH}	40		10	
Clock low to data out valid	t_{CLDO}		70		55
R/$\overline{W}$ low to data bus low impedance	t_{RLDO}	30		10	
Data out valid to DS* low (write)	t_{DOSL}	30		15	
Data hold from clock high	t_{CHDOI}	0		0	
DS* high to data out invalid	t_{SHDOI}	30		15	
DTACK* setup-time	t_{ASI}	20		20	
AS*, DS* high to DTACK*	t_{SHDAH}	0	245	0	150

68k READ Cycle Timing Analysis with Synchronous Sram Assuming SSRam clock driven from CPU clock

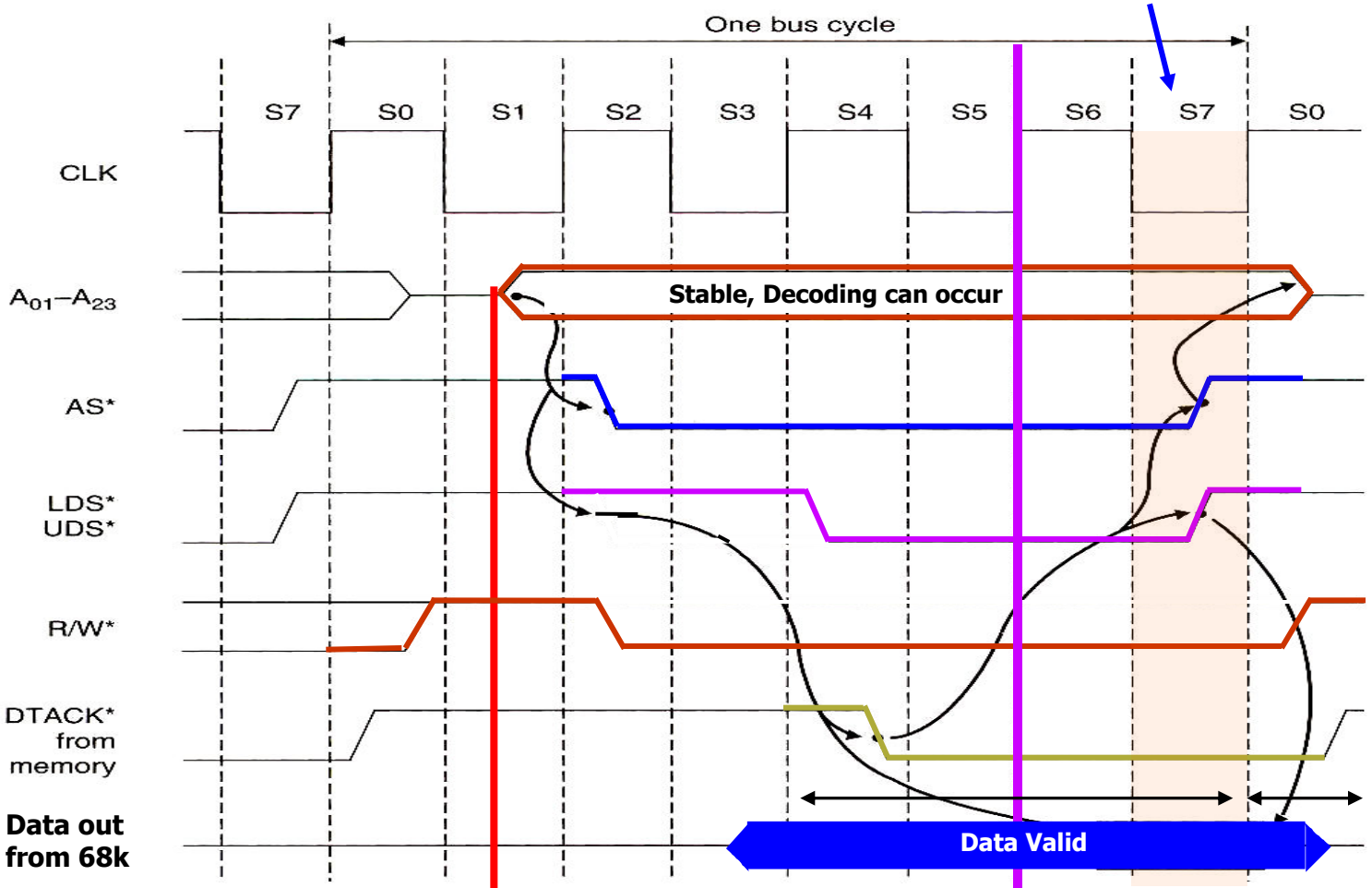


Address and control signals set up to latch here on rising edge of clock once AS*, Address, UDS*, LDS*, R/W* stable etc.

Available Read Access Time: 80ns for 12.5Mhz CPU clock

Data out available here on next rising edge

68k WRITE Cycle Timing Analysis with Synchronous Sram Assuming SSRam clock driven from CPU clock



Address and control signals set up to latch here on rising edge of clock, once AS*, Address, UDS*, LDS*, R/W*, Data out from 68k stable etc. Data written internally by chip.

With Synchnous memory it's better to control it with memory controller based on a state machine, (see Dram lecture later)