Shared data bus

General

COFFEE processor core allows sharing it's data memory bus with other devices. In simple applications all peripherals and data memory can be connected directly to COFFEE core data bus. Multiprocessor systems with shared memory are also possible. However, it should be noted that having too many devices communicating via shared bus eventually reduces performance since core will be stalled every time it needs the bus but it's not available.

Handshaking and accessing

Before performing access, external device has to signal a request to COFFEE core. If multiple devices are connected, a bus arbitrer must be used. Here, we assume that arbitrer or some other device is communicating with core. Two signals are used for handshaking: **bus_req**(input to core) and **bus_ack**(output from core). Handshaking scheme is basicly requesting and passing '*token*', that is the ownership of the bus.

An external device has to request the bus by asserting **bus_req** -signal. It can access the bus when core responds by asserting **bus_ack** -signal. This will happen immediately if core is not accessing the bus or after current access if the bus is in use. An external device must hold **bus_req** -signal active throughout it's access. It can perform several consecutive data transfers if core does not request the bus back (which is seen from **bus_ack** -signal going low).When core requests the bus by driving **bus_ack** -low an external device can finish current access, but must pass the token to core(this is only a recommendation, solution will depend on application). This is signalled by driving **bus_req** low. If an external device has finished it's access and has no further transfers, the token is automatically passed to core even if it does not request it. This happens when the device deasserts **bus_req** –signal. This guarantees that the core can access the bus

immediately if there’s no traffic(in other words, core has the token by default). After reset, core owns the token.

This simple scheme should be adequate for most applications: It's fair since the bus cannot be locked by one device and it's predictable because the bus will be available as soon as an active access is finished. Also the core, which is assumed to use the bus most, does not have to request the token if there's no traffic on bus.

When an external device owns the token, core floats both address and data lines, read and write controls will be low (signals rd,wr,pcb_rd and pcb_wr). Likewise, any other device must float the bus when core owns the token. Note that when core has the token but is not using the bus (bus idle cycle), it holds values of the last access on data and address lines(power saving feature).

Note, that read and write –signals from multiple devices must be connected via OR –gate because they are not tri-state signals. For detailed description of data memory interface signals, see document “COFFEE_interface”.

Timing

Figures T.5 through T.9 illustrate the timing of signals of the shared bus interface. Table 1 explains mnemonics and keywords.

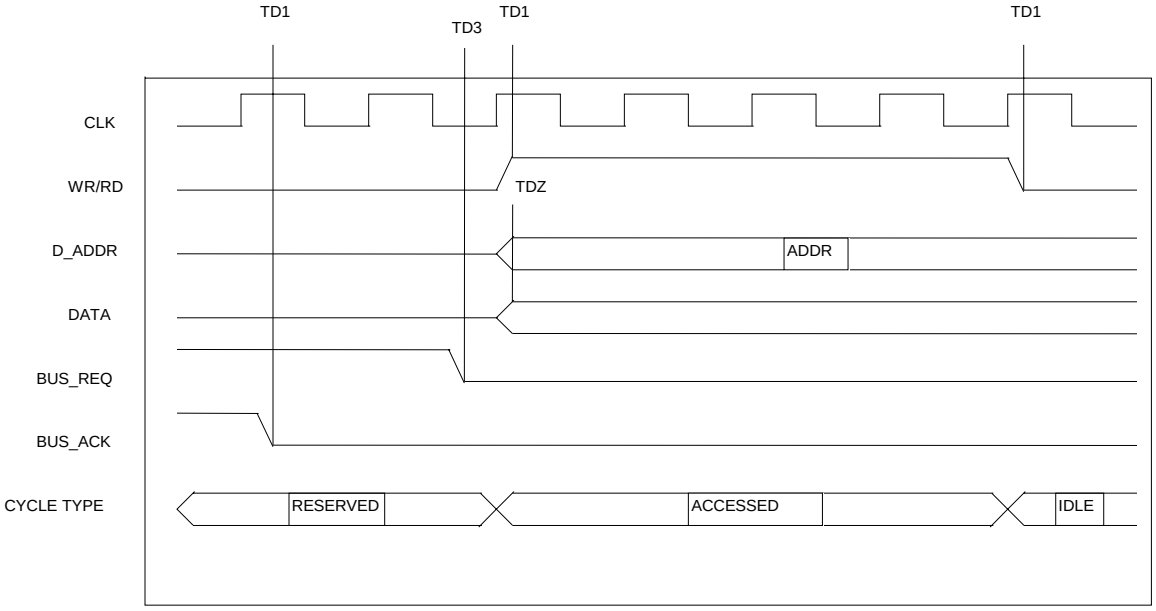

FIGURE T.5, DATA BUS STATE TRANSITION: RESERVED => ACCESSED

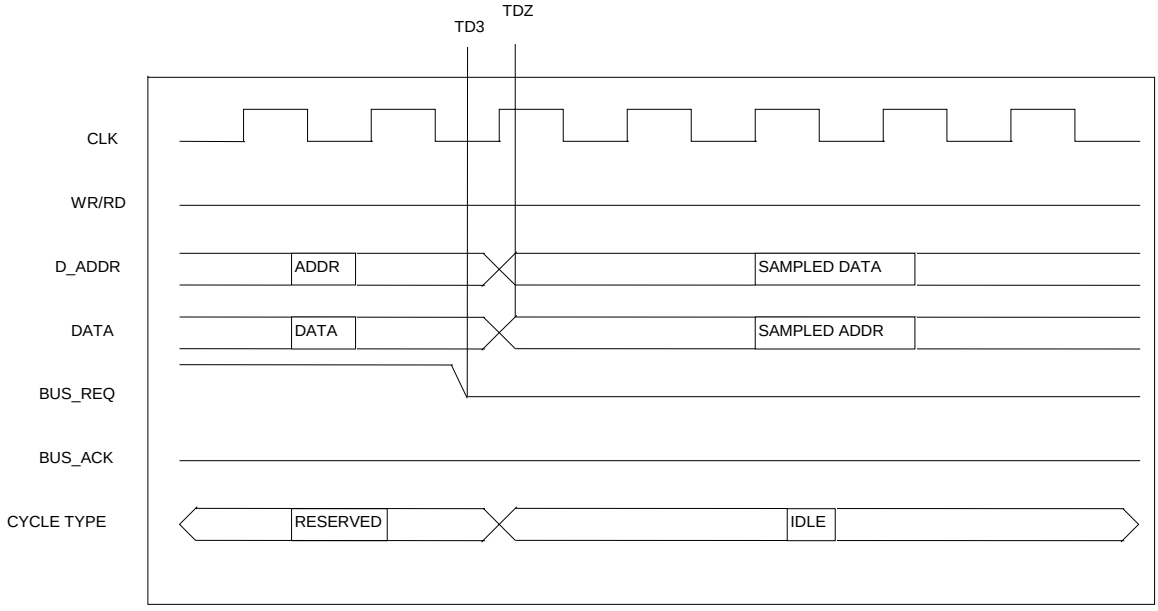

FIGURE T.6, DATA BUS STATE TRANSITION: RESERVED => IDLE

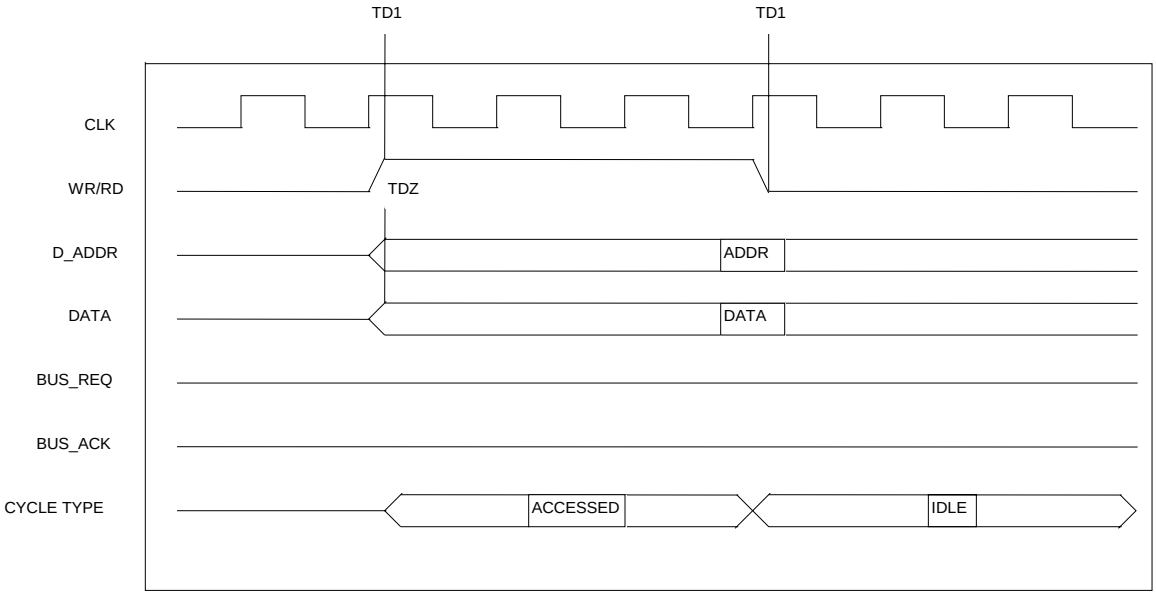

FIGURE T.7, DATA BUS STATE TRANSITION: ACCESSED => IDLE

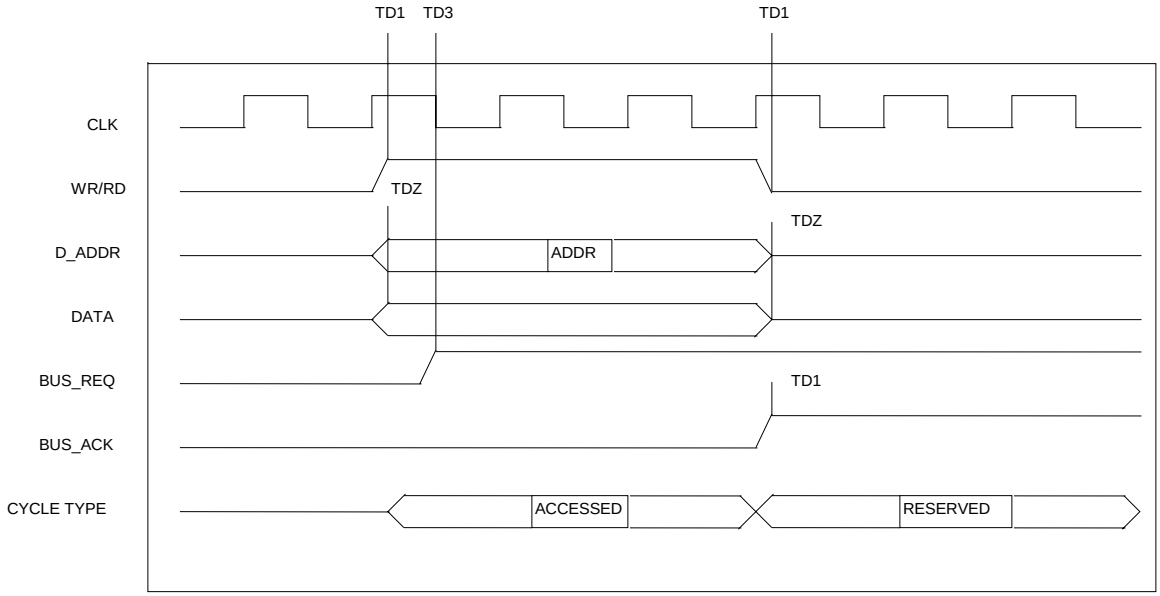

FIGURE T.8, DATA BUS STATE TRANSITION: ACCESSED => RESERVED

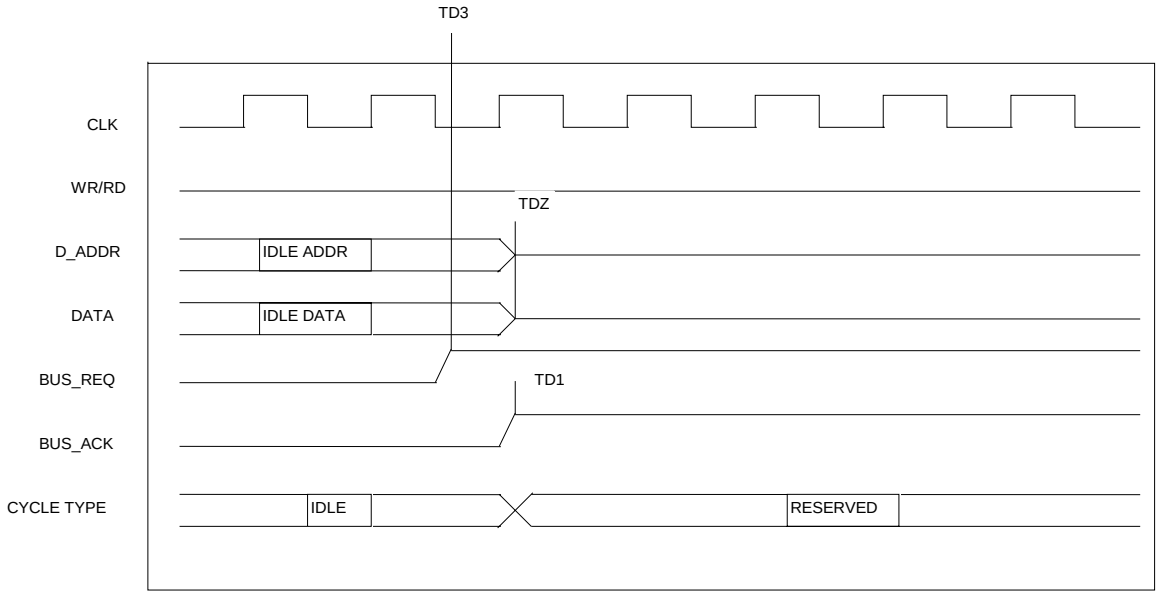

FIGURE T.9, DATA BUS STATE TRANSITION: IDLE => RESERVED

Table 1, Mnemonics and keywords in figures T.5 through T.9

keyword	explanation	notes
TD1	Delay from rising clock edge to the moment when data is valid on output of a D flip-flop.	Technology dependent.
TD3	Maximum delay of the bus_req –signal. Refer to results of preliminary synthesis.	Suitable synthesis constraints for bus_req – input must be set. See interface specification.
TDZ	TD1 + delay of a tri-state gate	See synthesis notes about handling tri-state control signals.
IDLE DATA	Data and address which are driven to bus by core when bus is in idle state. Should be the values from previous access unless the bus is floated during the last RESERVED –cycle.	
IDLE ADDR		
SAMPLED DATA		
SAMPLED ADDR		
ADDR	Valid address or data of an active access.	
DATA		
IDLE	Idle cycle of the bus, no active accesses. Core drives last values on bus.	

ACCESSED	Core owns the bus and is performing write or read access.	
RESERVED	An external device owns the bus.	