Synchronous memory access module

G.J. van Loo

28-February-2017

Introduction

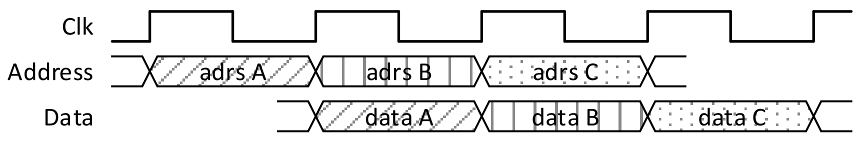
The Synchronous memory access module sits between an AHB bus and a synchronous memory. Its main purpose is to eliminate dead cycles due to the incompatibility of the AHB bus and the timing requirements of a synchronous memory. It also converts the AHB bus signals into byte-write strobes for the memory.

AHB synchronous memory write buffer.

The timing of the AHB bus gives excellent access to synchronous memory on reads. However without precautions, the system incurs a one cycle delay for each read-follows-write cycle. The main task of the AHB_SRAM module is to remove the one cycle delay by using a write-buffer. The details of this are explained in the next sections.

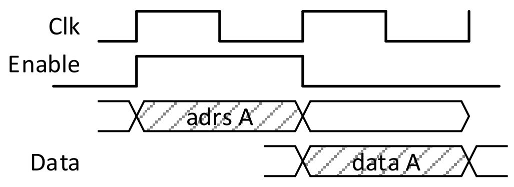
Read-follows-write delay

Below is the timing diagram of an AHB bus cycle. The address is preceding the data by one clock cycle for both read and write.

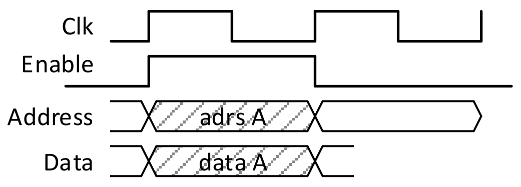


Standard AHB bus cycles

These are the timing diagrams of a read and write access to a synchronous memory:

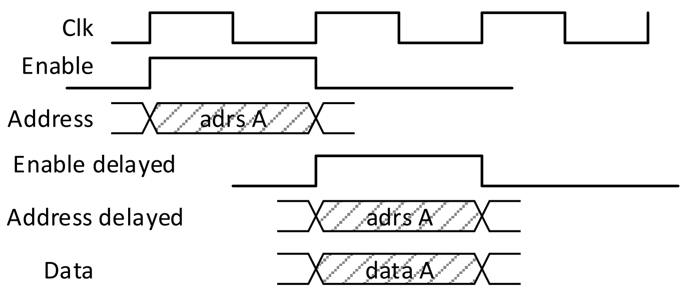


Memory read



Memory write

As you can see a memory read access corresponds to an AHB bus cycle. However the write access does not. In order to write to a synchronous memory from an AHB bus we have to delay[1] the address by one cycle in order for the address and data to line up:

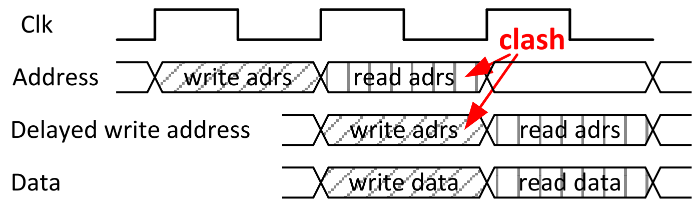


AHB write to sync. memory

[1] Unfortunately we cannot move the data forward as we have not yet found a way to look into the future.

Thus on a write we present the *delayed* AHB address to the memory instead of the AHB address.

This is all very well but if we have a read following the write we have to present ***two*** addresses to the memory ***at the same time:***



Memory read-follows-write

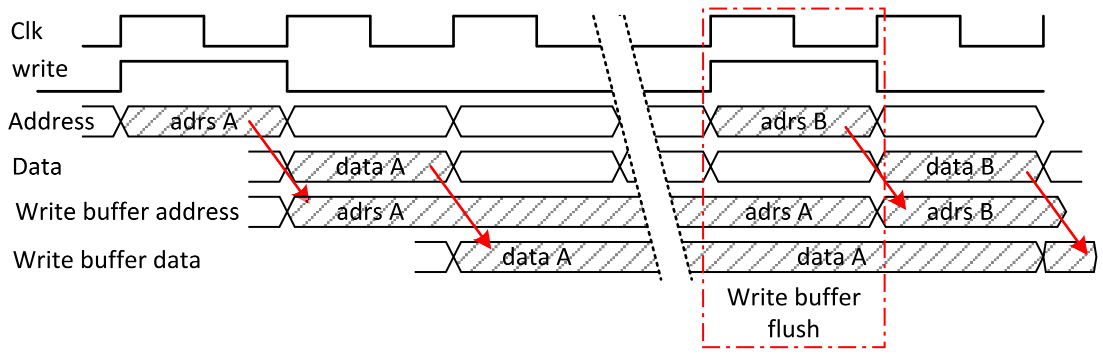
The only way around that is to insert a wait state on the bus, to delay the read by once cycle, probably delaying the processor also for a cycle.

Solution.

The one cycle delay can be eliminated by adding a write buffer. Upon writing to the memory the data is not written to the memory directly. Instead the write address and data go into a write buffer.

The write buffer is flushed to the memory when a new write request comes in. Because the address and the write data are both available in the write buffer, the write can be done in the same cycle as where the new write address arrives.

This is shown in the diagram below.



The write buffer is also flushed if no memory access takes place.

Buffer read-back.

The write buffer causes an inconsistency in that the memory contents does not reflect the required status as one location may be incorrect. Thus if a read request comes in, the read address is compared against the write buffer address. If there is a match the data comes (partially) from the write buffer, not the memory. The term ‘partially’ is used as the write strobes can indicate that only part of the write buffer should be written and thus only part should be used in the read-back. The module will mix the data from the memory with that of the write buffer to return the actual required information.

Copyright

February 28, 2017 Fen Logic Ltd.