FastMemoryLink (FML) bus specifications

Sébastien Bourdeauducq

December 2009

1 Introduction

The FastMemoryLink bus is designed to provide a high-performance interface between a DRAM controller and peripherals that need to access large amounts of data.

FML buses are referred to as *bxw FML*; which means that the bus operates with a burst length of b and that the width (in bits) of each unidirectional data line is w. For example, Milkymist uses a 4x64 FML bus; which means that each transfer with the DRAM controller is made up of four 64-bit chunks.

Its main features are the following:

- Synchronism. The bus is meant to be used in FPGA-based devices, whose architectures are designed for synchronous (clock-driven) systems.
- Burst oriented. Each cycle begins with an address phase, which is then followed by several data chunks which are transferred on consecutive clock edges (the data phase). The length of the burst is fixed.
- Pipelined transfers. During the data phase of a cycle, the control lines are free and can be used to initiate the address phase of the next cycle.

2 Specifications

2.1 Signals

A FastMemoryLink interface is made up of the following signals:

Signal	Width	Direction	Description
a	User-spec.	Master to slave	Address signals. They are used to specify the location in DRAM to be accessed.
stb	1	Master to slave	Strobe signal. This signal qualifies a cycle. Once it has been asserted, it cannot be deasserted until the cycle has been acknowledged by the slave.
we	1	Master to slave	Write enable signal.
ack	1	Slave to master	Acknowledgement signal. This signal is asserted for one cycle by the slave when it is ready to begin the data phase.
dw	User-spec. (w)	Master to slave	Write data.
dr	User-spec. (w)	Slave to master	Read data.

2.2 Single read cycle

The master initiates a read cycle by presenting a valid address, deasserting `we`, and asserting `stb`.

At least one clock cycle later, the slave presents valid data on `dr` and asserts `ack` to mark the beginning of the data phase. During the b-1 subsequent cycles, the slave keeps sending nearby data using the `dr` lines only (see the "Burst ordering" section below).

Here is an example single read timing diagram for a bus using a burst length of 4.

a	X	A	A	X	X	X	X
stb	0	1	1	0	0	0	0
we	X	0	0	X	X	X	X
ack	0	0	1	0	0	0	0
dw	X	X	X	X	X	X	X
dr	X	X	A+0	A+1	A+2	A+3	X

X = don't care

2.3 Single write cycle

The master initiates a write cycle by presenting valid address and data, asserting `we`, and asserting `stb`.

At least one clock cycle later, the slave asserts `ack` to mark the beginning of the data phase. During the b-1 subsequent cycles, the master keeps sending nearby data using the `dw` lines only (see the "Burst ordering" section below).

Here is an example single write timing diagram for a bus using a burst length of 4.

a	X	A	A	X	X	X	X
stb	0	1	1	0	0	0	0
we	X	1	1	X	X	X	X
ack	0	0	1	0	0	0	0
dw	X	A+0	A+0	A+1	A+2	A+3	X
dr	X	X	X	X	X	X	X

2.4 General restrictions

While waiting for the `ack` signal to become active, the master must continue to assert `stb` and keep `a` and `we` constant.

On the cycle following the assertion of the `ack` signal, the slave must deassert `stb` unless it wants to start a new (pipelined) cycle (see below).

The slave is not allowed to assert `ack` when `stb` has not been asserted for at least one cycle.

2.5 Pipelined read cycles

To maximize bus utilisation and reduce latency, the master is allowed to start the address phase of the next cycle during the data phase of the current cycle.

The slave must not acknowledge the new cycle until all data from the current cycle have been transferred.

Here is an example timing diagram.

a	X	A	A	B	B	B	B	X	X	X	X
stb	0	1	1	1	1	1	1	0	0	0	0
we	X	0	0	0	0	0	0	X	X	X	X
ack	0	0	1	0	0	0	1	0	0	0	0
dw	X	X	X	X	X	X	X	X	X	X	X
dr	X	X	A+0	A+1	A+2	A+3	B+0	B+1	B+2	B+3	X

2.6 Pipelined write cycles

Writes can also be pipelined. However, the master cannot present its data immediately on the write lines since they are already busy. Instead, it must present its data as soon as possible; that is, immediately after the last word of the current cycle has been transferred.

Here is an example timing diagram.

a	X	A	A	B	B	B	B	X	X	X	X
stb	0	1	1	1	1	1	1	0	0	0	0
we	X	1	1	1	1	1	1	X	X	X	X
ack	0	0	1	0	0	0	1	0	0	0	0
dw	X	A+0	A+0	A+1	A+2	A+3	B+0	B+1	B+2	B+3	X
dr	X	X	X	X	X	X	X	X	X	X	X

2.7 Overlapping reads and writes

The FML bus allows overlapping read and write cycles. This typically requires a relatively complex DRAM controller which implements a write queue.

The slave must assert the `ack` signal at least two clock cycles after it asserted it to acknowledge the previous bus cycle.

The following timing diagram shows a read cycle which is overlapped by a write cycle.

a	X	A	A	B	B	X	X	X
stb	0	1	1	1	1	0	0	0
we	X	0	0	1	1	X	X	X
ack	0	0	1	0	1	0	0	0
dw	X	X	X	X	B+0	B+1	B+2	B+3
dr	X	X	A+0	A+1	A+2	A+3	X	X

2.8 Burst ordering

The modulo b of the address is used to specify the burst ordering.

It is strongly suggested that b should be a power of 2, so that the modulo and quotient can be computed by simply slicing the address bit vector.

The bus uses a linear wrapping burst ordering. For example, on a bus with a burst length of 4, an access starting at address 128 yields the addresses `128; 129; 130; 131`, and an access starting at address 129 yields the addresses `129; 130; 131; 128`.

Burst reordering can be used to implement critical-word-first in caches.

Copyright notice