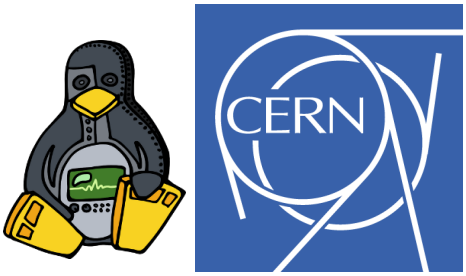
m25p32 manager HDL core

Carlos Gil Soriano
BE-CO-HT
carlos.gil.soriano@cern.ch

June 10, 2012



Abstract

A module for accessing m25p32 EEPROM memory. It is able to perform writes and reads thanks to *SPI multifields HDL core*.

The present documentation address the following subjects:

- The registers to control the module.
- Step-by-step instructions for proper use. access.

Revision history		
HDL version	**Module**	**Date**
0.1	m25p32 manager core (Preliminary)	July 25, 2012
0.9	m25p32 manager core	Oct. 25, 2012

Contents

1 Structure

The m25p32 manager module contains several blocks related the following way:

– m25p32_pkg.vhd
– m25p32_top.vhd
—— m25p32_master_regs.vhd
—— m25p32_core.vhd
——— spi_master_core.vhd

1.1 Dependencies

m25p32 core depends on *spi_master_core.vhd*, thus, all the dependencies of *spi_master_core.vhd* must be met. The corresponding files from *ctdah_lib* must be added to the project.

2 Registers

2.1 FMI

The FMI, *Flash Memory Instruction Register* is a write-read register. It can be write-read in all the bit fields but OPA and OPF, which are read-only. When performing a write into the *FMI* register, bits OPA and OPF will not be written on.

Bits	Field	Meaning	Default
0	OPR	OPeration Requested	**'0'**
1	OPA	OPeration Attended	**'0'**
2	OPF	OPeration Finished	**'0'**
5-3	OP	OPeration to perform	**x**
13-6	PG	PaGe number	**0**
19-14	SCT	SeCTor number	**0**
23-20	y	Reserved	**x0**

2.1.1 Available FMI SPI Operations

The table below shows the available operations that can be commanded through field *OP* in *FMI* register.

Operation	Meaning	OP code
x	Reserved	**"000"**
ERS	ERase sector. Sets everything to FF	**"001"**
RDP	ReaD Page. Not implemented by now	**"010"**
WRP	WRite Page	**"011"**
RDSR	ReaD Status Register	**"100"**
WRSR	WRite Status Register	**"101"**

2.1.2 SPI Instructions implemented

The following SPI instructions have been implemented:

Instruction	Description	Address Bytes	Data Bytes	SPI Code
WREN	WRite ENable	0	0	0x06
WRDI	WRite DIsable	0	0	0x04
RDSR	ReaD Status Register	0	1	0x05
WRSR	WRite Status Register	0	1	0x01
READ	READ data bytes	3	4	0x03
PP	Page Program	3	256	0x02
SE	Sector Erase	3	0	0xD8
BE	Bulk Erase	0	0	0xC7

It should be noted that the bytes read back from a READ instruction have been bounded to four consecutives bytes. This helps to reduce the complexity of the design and easily let the data to be read from the wishbone interface.

3 Internal memory mapping

The internal registers map is as follow:

Address prefix	Padding	Address suffix	Register	Access
'0'	**0's**	**"00"**	*FMI register*	Write-read
'0'	**0's**	**"01"**	*m25p32 SR*	Write-only
'0'	**0's**	**"10"**	*MISO bytes read*	Read-only
'1'	**0's**	**"00"**	*Least significant word to be written into MOSI*	Write-read
'1'	**...**	**"..."**	*Word 'n' to be written into MOSI*	Write-read
'1'	**1's**	**"11"**	*Most significant word to be written into MOSI*	Write-read

4 How to use it

ADDITIONAL In the case that data will be outputted in the MOSI pin, write all the internal memory area allocated for MOSI writes. In the case of a WRSR instruction this will take wishbone transaction. For PP instructions, it will take the number of words that can fit in one page.

1.- Write the operation to be performed to the flash memory via the SPI command in **FMI register**.