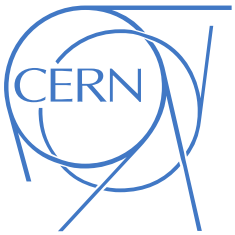
ELMA I^{2}C to Wishbone bridge

July 22, 2013



Theodor-Adrian Stana (CERN/BE-CO-HT)

Revision history

Date	Version	Change
26-06-2013	0.01	First draft

Contents

List of Figures

List of Tables

List of Abbreviations

FSM	Finite-State Machine
I^2C	Inter-Integrated Circuit (bus)
SysMon	ELMA crate System Monitor board
VME	VERSAmodule Eurocard

1 Introduction

This document describes the *elma_i2c* module, an I^{2}C to Wishbone bridge HDL core for VME64x crates from ELMA. These crates offer the possibility of accessing boards in VME slots via either VME, or I^{2}C. Boards not using the VME lines on a slot can implement the *elma_i2c* module on an FPGA; implements an I^{2}C slave and translates I^{2}C accesses into Wishbone [1] accesses to a Wishbone slave device.

A typical system where the *elma_i2c* module is employed is shown in Figure 1. ELMA VME crates contain a SysMon (system monitor) board [2], that is mainly used for monitoring VME voltages and controlling the fans of the VME crate. The SysMon can be connected to via either a serial connection or Telnet. Then, sending specific commands (see Section 3) via one of the two are translated by the SysMon into I^{2}C accesses following the protocol described in Section 4.

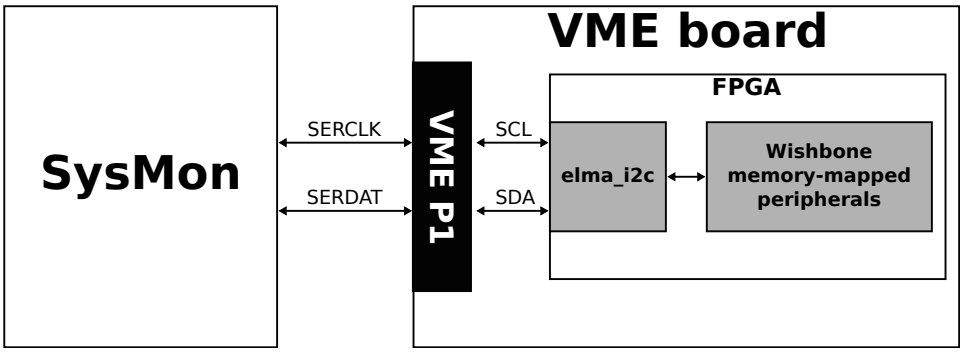


Figure 1: Typical system for the *elma_i2c* module

2 Instantiation

The ports of the *elma_i2c* module are shown in Table 1. The I^{2}C signals should be connected to tri-state ports, as shown in Figure 2; Wishbone slaves should be connected to the Wishbone master interface ports, prefixed with *wbm*.

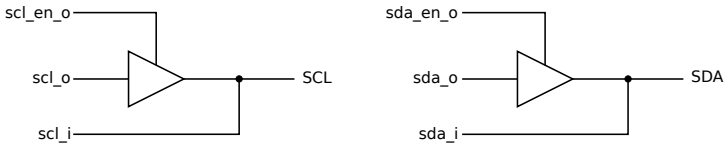


Figure 2: I^{2}C port external connections

Table 1: Ports of *elma_i2c* module

Port	Size	Description
clk_i	1	Clock input
rst_n_i	1	Active-low reset input
sda_en_o	1	SDA line output tri-state enable
sda_i	1	SDA line input
sda_o	1	SDA line output
scl_en_o	1	SCL line tri-state enable
scl_i	1	SCL line input
scl_o	1	SCL line output
i2c_addr_i	7	I^2C slave address on ELMA I^2C bus
i2c_done_o	1	High for one clk_i cycle when an I^2C transfer is finished
i2c_err_o	1	High for one clk_i cycle when an error occurs in the ELMA protocol or an attempt is made to access an non-existing register on the Wishbone bus
wbm_stb_o	1	Wishbone data strobe output
wbm_cyc_o	1	Wishbone valid cycle output
wbm_sel_o	4	Wishbone byte select output
wbm_we_o	1	Wishbone write enable output
wbm_dat_i	32	Wishbone data input (to master)
wbm_dat_o	32	Wishbone data output (from master)
wbm_adr_o	32	Wishbone address output
wbm_ack_i	1	Wishbone acknowledge signal input
wbm_rty_i	1	Wishbone retry signal input
wbm_err_i	1	Wishbone error signal input

3 Testing the *elma_i2c* module

After proper synthesis and download to the FPGA, a Telnet or serial connection should be made to the SysMon board. Commands can then be sent to the boards via the SysMon. The two commands relevant for accessing board registers are *readreg* and *writereg*, outlined in Table 2.

Register (*reg*) numbers in these commands are decimal numbers starting from 1. The SysMon translates *reg* numbers into word-aligned addresses, thus in order to obtain the actual register addres, the following relation should be used:

$$addr = (reg - 1) * 4$$

Table 3 shows the *reg* numbers of registers in the address space 0x00 to 0x20.

Table 2: The *readreg* and *writereg* commands

Command	Description
writereg *slot reg val*	Writes the value *val* to register number *reg* of board in slot number *slot*
readreg *slot reg*	Returns the value of register number *reg* of board in slot number *slot*

Table 3: Translating *reg* numbers to addresses

reg	**Address**
1	0x00
2	0x04
3	0x08
4	0x0C
5	0x10
6	0x14
7	0x18
8	0x1C
9	0x20

The example below shows how to connect to an ELMA crate at IP address 1.2.3.4, obtaining the value of a register at address 0x10 in a board in VME slot 2, writing the decimal value 12 to the same register and reading it back to check for proper modification.

```
$ telnet 1.2.3.4
Trying 1.2.3.4...
Connected to 1.2.3.4.
Escape character is '^]'.
login:user
password:**********
%>readreg 2 5
 Read Data: 00ABCDEF
%>writereg 2 5 12
 Done!
%>readreg 2 5
 Read Data: 0000000C
```

4 ELMA I²C Protocol

Using the I²C lines on the VME P1 connector, one can access boards placed in a VME crate. For this purpose, ELMA has defined a higher-level protocol [3] that uses I²C as a low-level protocol.

Figure 3 shows a write operation from the SysMon to a VME board. The process starts with the control byte, containing the board's I²C slave address and the read/write bit cleared, indicating an I²C write. After the slave's ACK, the following two bytes send the 12-bit address in little-endian order (most significant byte first). After the address has been acknowledged, the following four I²C transfers are used to transmit the 32-bit data to be written to the board register. Data transmission occurs in big-endian order (least significant byte first).

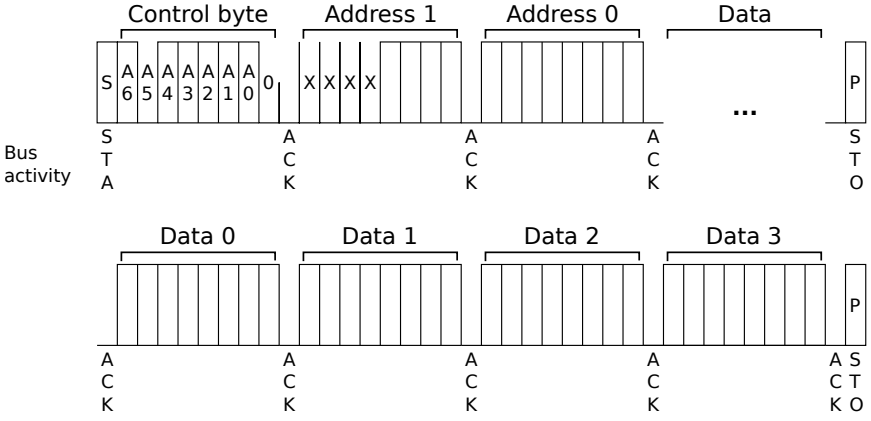


Figure 3: SysMon write operation

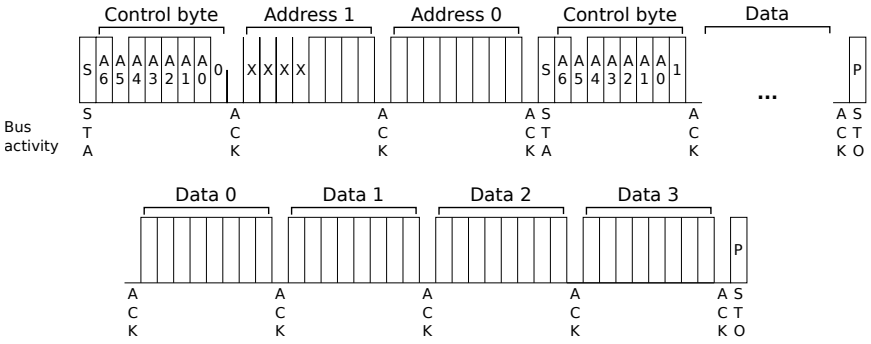


Figure 4: SysMon read operation

A read transfer (Figure 4) from a VME board is similar to the write transfer. The differences lie in the retransmission of the control byte after the register address, this time with the read/write bit set, to indicate an I²C

read. Following the ACK from the slave, the transfer direction changes and the SysMon will read the four data bytes sent by the VME board. As with the write transfer, the data bytes are sent by the VME board in big-endian order.

5 Implementation

In order to perform low-level I^2C transfers, the *i2c_slave* module REFERENCE? is instantiated and used within the *elma_i2c* module. The outputs of the *i2c_slave* module are used as controls for an eight-state finite state machine (FSM), a simplified version of which is shown in Figure 5. Table 4 also lists the states of the state machine.

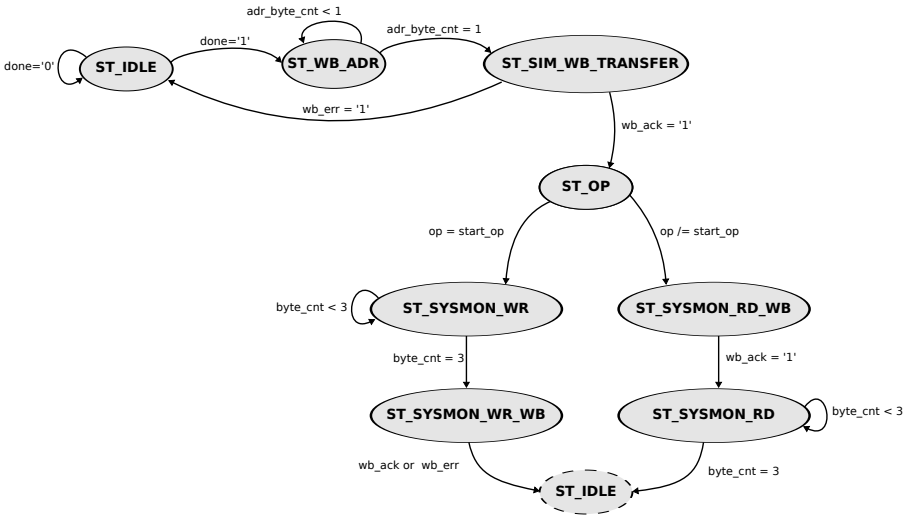


Figure 5: Main FSM of *elma_i2c* module

When the *i2c_slave* module finishes a transfer (signaled by a *done_p_o* pulse), the status is checked and if it is as expected (e.g., a *address good* in the *ST_IDLE* state), the FSM advances to the next state. It should be noted that where the SysMon appears in the state names, it indicates what the SysMon action is. For example, if the state of the FSM is *ST_SYSMON_WR*, this means the SysMon is writing and the *elma_i2c* is reading.

To better understand how the FSM operates, Figures 6 and 7 can be consulted, where the state of the FSM is shown during reads and writes from the SysMon.

When the SysMon writes (Figure 6), the *elma_i2c* module waits in the *ST_IDLE* state until the I^2C address is received, then, while in the *ST_WB_ADR* state, it shifts in the Wishbone address. A Wishbone transfer is then simulated with the received the address and if this address exists (a Wishbone *ack* is received), the first byte is shifted in while in the *ST_OP*

Table 4: States of *elma_i2c* FSM

State	Description
ST_IDLE	Wait for the *i2c_slave* module to receive the I^2C address and go to *ST_WB_ADR*. The starting value at the *op_o* output of the *i2c_slave* module is stored for checking in *ST_OP*
ST_WB_ADR	Shift in the two address bytes sent via I^2C and go to *ST_SIM_WB_TRANSF*
ST_SIM_WB_TRANSF	Start a Wishbone read transfer from address received in previous state and go to *ST_OP* if Wishbone address exists (Wishbone *ack* received), or *ST_IDLE* otherwise (Wishbone *err* received)
ST_OP	Check the *op_o* output of the *i2c_slave* module. If different from the value at the start, go to *ST_SYSMON_RD_WB* state (SysMon is reading from *elma_i2c*), otherwise continue shifting in bytes (SysMon writing to *elma_i2c*)
ST_SYSMON_WR	Continue reading up to four bytes sent by the SysMon and go to *ST_SYSMON_WR_WB*
ST_SYSMON_WR_WB	Perform a Wishbone write transfer to the register with the address obtained in *ST_WB_ADR*
ST_SYSMON_RD_WB	Perform a Wishbone read transfer from the address obtained in *ST_WB_ADR* and go to *ST_SYSMON_RD*
ST_SYSMON_RD	Shift out the four bytes of the Wishbone register when the *i2c_slave* module successfully finishes a write

state, followed by the next three bytes while in the *ST_SYSMON_WR* state. Finally, the register is written to in the *ST_SYSMON_WR_WB* state.

When the SysMon reads (Figure 7), the first few steps are the same as for a read. The address is shifted in and checked in the Wishbone transfer simulation state. In the case of a SysMon reading from a board, however, the I^2C transfer is restarted and the order is reversed (SysMon starts reading). Thus, while in *ST_OP*, the FSM detects a different value of *op_o* and goes into the *ST_SYSMON_RD_WB* state. The value of the register is read here and sent via I^2C in the *ST_SYSMON_RD* state.

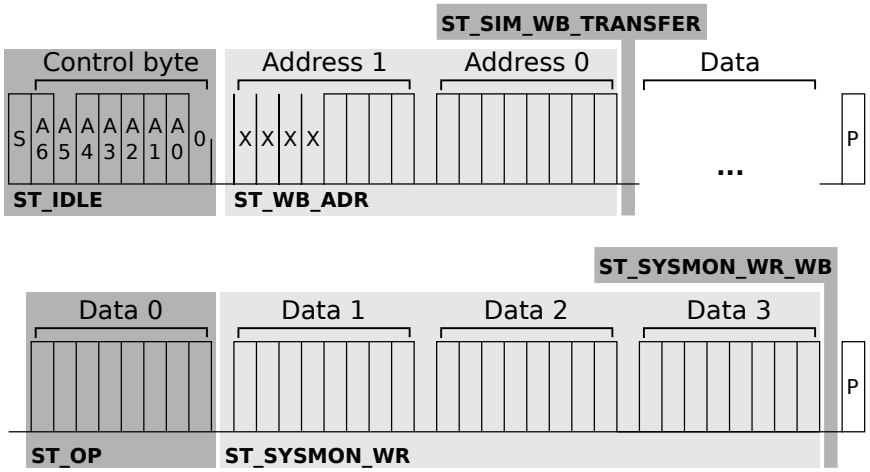


Figure 6: FSM states when the SysMon writes to the *elma_i2c*

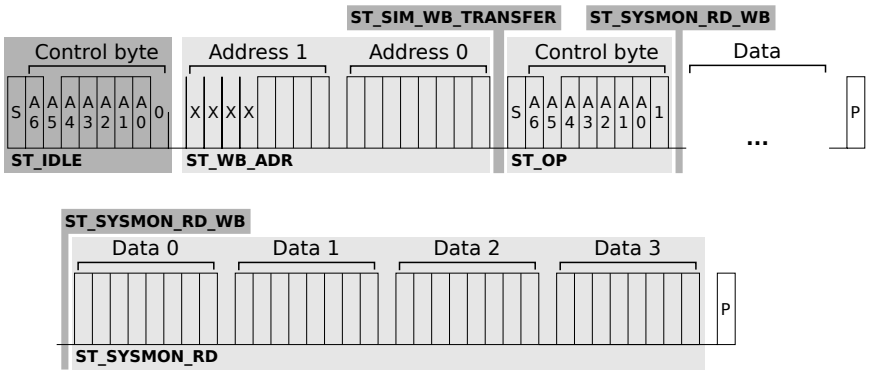


Figure 7: FSM states when the SysMon reads from the *elma_i2c*

References

[1] OpenCores, "Wishbone System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores." http://cdn.opencores.org/downloads/wbspec_b4.pdf.

[2] ELMA, "New SysMon User Manual Rev. 1.11." http://www.ohwr.org/documents/226.

[3] ELMA, "Access to board data using SNMP and I2C." http://www.ohwr.org/documents/227.