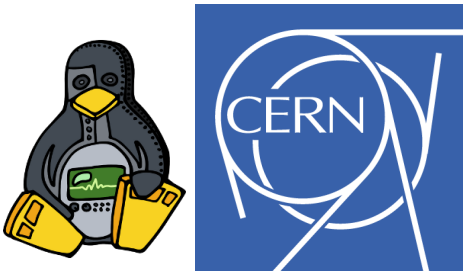
I2C Slave to Wishbone Master module

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

November 26, 2012



Abstract

An I2C Slave to Wishbone Master module is depicted in this document. Architecture-independent core kindly written in VHDL.

The following subjects are addressed:

- Control registers of the module
- Step-by-step instructions for proper setup and use.

Revision history		
HDL version	**Module**	**Date**
0.1	I2C slave	February 23, 2012
0.9	I2C slave	November 26, 2012

Contents

List of Tables

List of Figures

1 Structure

The i2c module contains several blocks related the following way:

– i2c_slave_top.vhd
—— i2c_regs.vhd
—— i2c_slave_core.vhd
——— gc_counter.vhd
——— gc_ff.vhd
——— i2c_bit.vhd
————— i2c_debouncer.vhd
The I2C slave functionalities are:

- Addresseable at both general or individual address.
- Three independent fields (indirect wishbone address, write bytes into i2c slave and read bytes from slave) can be configured to help integration.
- A watchdog timer can be enable to increase reliability.

It is recommended to take a look to the I2C standard before continue reading.

1.1 Intended use

I2C Slave to Wishbone Master module is intended to be used in both CONV-TTL-BLO [1] and CONV-TTL-RS485 [2] projects to allow FPGA be remotely reprogrammed. Read and write operations have been specified as follows:

1.1.1 I2C Master write into slave

Start The master will issue a start condition.

Byte 1 The master will issue a write operation to a given I2C slave address. Less significant bit is driven first into SDA line.

Ack 1 An ack is expected from slave.

Byte 2 The master will drive the SDA line with the upper-half part of the wishbone addres to be read. Most significant bit is driven first into the SDA line.

Ack 2 An ack is expected from slave.

Byte 3 The master will drive the SDA line with the lower-half part of the wishbone addres to be read. Most significant bit is driven first into the SDA line.

Ack 3 An ack is expected from slave.

Byte 4 The master will drive the SDA line with the highest byte to be written at the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 4 An ack is expected from slave.

Byte 5 The master will drive the SDA line with the second highest byte to be written at the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 5 An ack is expected from slave.

Byte 6 The master will drive the SDA line with the second lowest byte to be written at the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 6 An ack is expected from slave.

Byte 7 The master will drive the SDA line with the second lowest byte to be written at the given wishbone address. Most significant bit is driven first into the SDA line.

[1] EDA-02446
[2] EDA-02541

Ack 7 An ack is expected from slave.

Refer to appendix A for a detailed waveform diagram.

1.1.2 I2C Master read from slave

Start The master will issue a start condition.

Byte 1 The master will issue a write operation to a given I2C slave address. Less significant bit is driven first into SDA line.

Ack 1 An ack is expected from slave.

Byte 2 The master will drive the SDA line with the upper-half part of the wishbone addres to be read. Most significant bit is driven first into the SDA line.

Ack 2 An ack is expected from slave.

Byte 3 The master will drive the SDA line with the lower-half part of the wishbone addres to be read. Most significant bit is driven first into the SDA line.

Ack 3 An ack is expected from slave.

Restart The master will issue a (re)start condition.

Byte 4 The master will issue a read operation to a given I2C slave address. Less significant bit is driven first into SDA line.

Byte 5 The slave will drive the SDA line with the highest byte to be read from the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 5 An nack is expected from master.

Byte 6 The slave will drive the SDA line with the second highest byte to be read from the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 6 An nack is expected from master.

Byte 7 The slave will drive the SDA line with the second lowest byte to be read from the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 7 An nack is expected from master.

Byte 8 The slave will drive the SDA line with the lowest byte to be read from the given wishbone address. Most significant bit is driven first into the SDA line.

Ack 8 An nack is expected from master.

Pause The master will issue a pause condition.

Refer to appendix B for a detailed waveform diagram.

1.2 Interrupting lines offered from top or core modules

To increase functionality of the core, some interrupt lines are offered so the IP core can be used together with a soft-core.

1.2.1 pf_wb_addr_o

One-clock signal to request prefetch of the data specified by upon the reception of the wishbone address in the i2c.

1.2.2 rd_done_o

The slave has finished to read the SDA line. Hence, this signal informs that a **i2c master write operation into the slave** has completed.

1.2.3 wr_done_o

The slave has finished to write into the SDA line. Hence, this signal informs that a **i2c master read operation from slave** has completed.

2 Registers

2.1 CTR0

CTR0 register is a **write-read** register.

It controls the indirect addres and holds the I2C address (which in the case of *CONV-TTL-BLO* will be connected to VME64x geographical address pins).

Bits	Field	Meaning	Default
0	I2C_OP	I2C OPeration	**'0'**
7-1	I2C_ADDR	I2C address	**See below**
11-8	BIA	Bytes of Indirect Addressing	**X"2"**
19-12	BRD	Bytes to be ReaD from FPGA	**X"4"**
27-20	BWR	Bytes to be WRitten to FPGA	**X"4"**
31-28	x	Reserved	**X"0"**

Table 1: *CTR0* register

The default value of I2C_ADDR is **c_I2C_GENERAL_ADDR**. It can be found in *i2c_slave_pkg.vhd*.

2.2 LT

LT stands for Line Timing register. It is a **read-only** register.

By accessing to this register, the speed of the line can be inferred. The value of *WBCP* is fixed prior synthesis, whereas *SCLP* it is updated on every transaction carried out in the I2C bus. SCLP returns the number of wishbone cycles within an I2C clock period:

$$I2C_{speed} = \frac{SCLP}{WBCP} \tag{1}$$

It should be noted that SCLP is an averaged value of the wishbone counter among the first eight bits received (the ones that correspond to I2C address plus write/read bit).

Bits	Field	Meaning	Default
7-0	WBCP	WishBone Clock Period	**User dependant**
31-8	SCLP	SCL Period	**X"000000"**

Table 2: *LT* register

2.3 DTX

The *DTX* register is a **write-read** register.

It holds the data to be sent when a read request from the master is issued. [3]

Bits	Field	Meaning	Default
31-0	data	data to send	**Not valid**

Table 3: *DTX* register

2.4 DRXA

The *DRXA* register is a **read-only** register.

It holds the last four received bytes through the I2C. LSB ordered.

Bits	Field	Meaning	Default
31-0	data	Data RX register	**Not valid**

Table 4: *DRXA* register

2.5 DRXB

The*DRXB* register is a **read-only** register.

It holds the second group of last four received bytes through I2C. LSB ordered.

[3]Contents are left uninitialized to reduce IP size.

Bits	Field	Meaning	Default
31-0	data	Data RX register	**Not valid**

Table 5: *DRXB* register

3 Internal Memory Mapping

The wishbone interface is addressed with 4-bit depth. Memory maps into:

Address	Register	Access
0x0	*CTR0*	Write-read
0x1	*LT*	Read-only
0x2	*DTX*	Write-read
0x3	*DRXA*	Read-only
0x4	*DRXB*	Read-only

Table 6: I2C internal memory map

4 (Re)Initilization

It takes place when:

- An external reset is issued.
- The watchdog timer runs out.
- An I2C transaction is finished.

Subsequently, *CTR0*, which controls the behaviour of the IP core, (re)initializes to:

Bits	**Field**	**Default**
0	I2C_OP	**'0'**
7-1	I2C_ADDR	**See below**
11-8	BIA	**X"2"**
19-12	BRD	**X"4"**
27-20	BWR	**X"4"**
31-28	x	**X"0"**

Table 7: *CTR0* register

I2C_ADDR is initialized to the value of the **i2c_addr_i** port in *i2c_slave_top*.

For a better understanding of the i2c finite state machine, please take a look to the appendix.

4.1 Watchdog timer

A fixed-length watchdog timer has been included in the module.

The watchdog timer is able to reinitialize the module whenever a fixed deadline is met. The value of this fixed deadline can be controlled *before synthesis* in i2c_slave_pkg.vhd.

Constant	**Meaning**
c_WB_CLK_PERIOD	Wishbone clock period (ns)
c_WATCHDOG_DEADLINE	Deadline period (ns)
c_WATCHDOG_ENABLE	Watchdog enable
c_WATCHDOG_WIDTH	Bit length of watchdog counter

Table 8: Watchdog constants

5 Modifiying length of the I2C fields

This module was conceived to let the user easily modify the length of three main fields.

Due to the integration into the project it was requiered for, the selectable-length fields are:

I2C field	**Select length**
Wishbone address	*CTR0.**BIA***
Data to read from slave	*CTR0.**BRD***
Data to write into slave	*CTR0.**BWR***

Table 9: I2C multifield capability

A I2C master write into slave

A I2C master write into a slave is shown. *CTR0* is configured as:

- **CTR0.I2C_ADDR**: X"3"
- **CTR0.BIA**: X"2"
- **CTR0.BWR**: X"4"

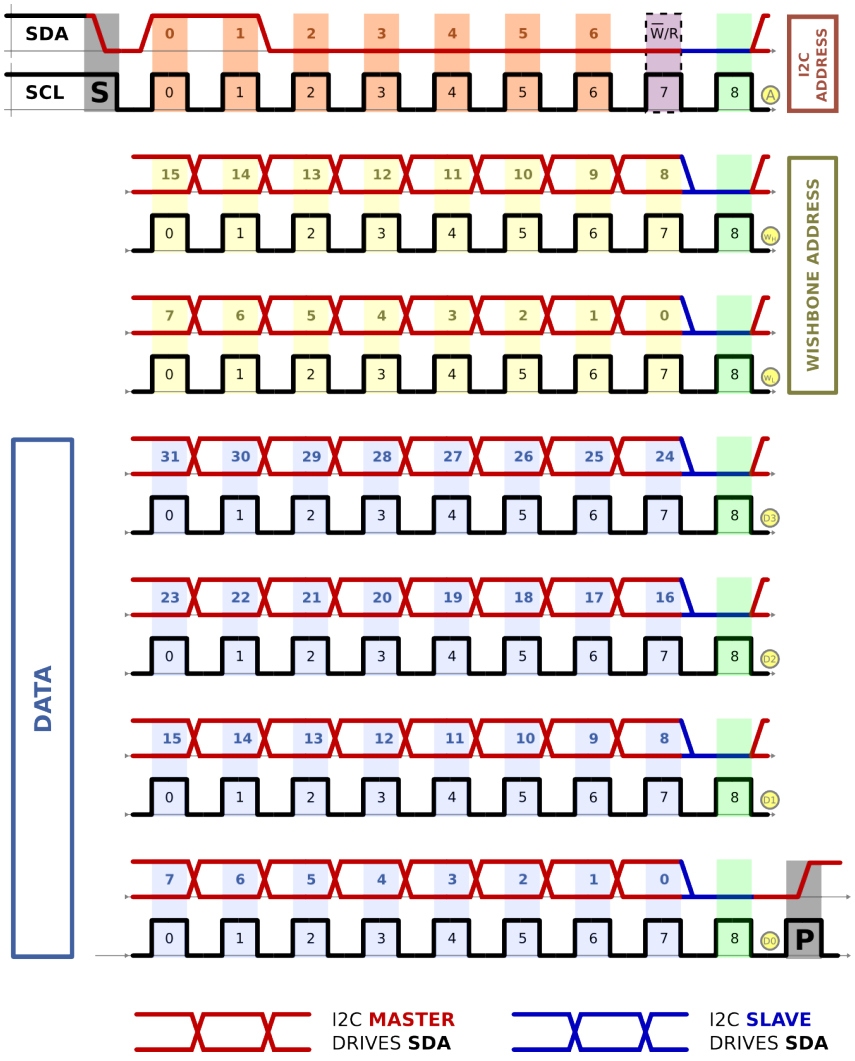


Figure 1: I2C master write into slave

B I2C master read from slave

A I2C master write into a slave is shown. *CTR0* is configured as:

- **CTR0.I2C_ADDR**: X"3"
- **CTR0.BIA**: X"2"
- **CTR0.BRD**: X"4"

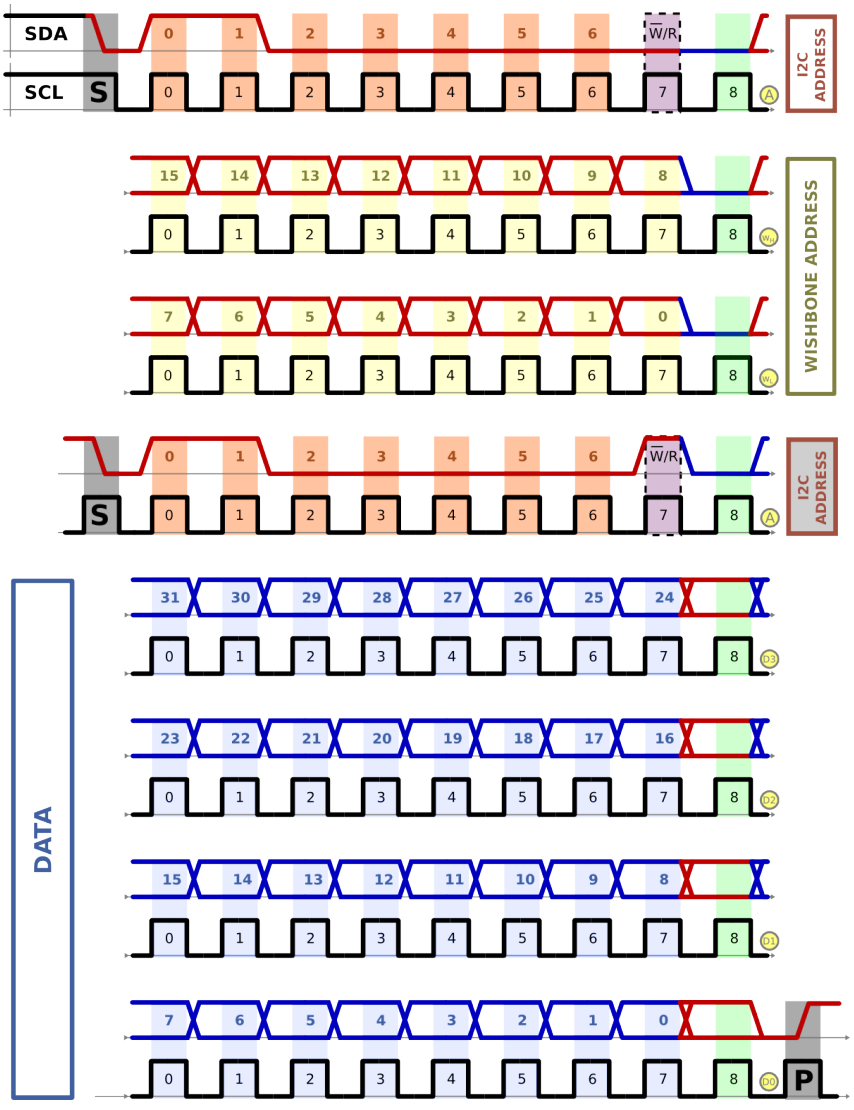


Figure 2: I2C master read from slave

C i2c_slave_core Finite State Machine

The finite state machine found in *i2c_slave_core* is shown below. Please note consistency with the colour schema in previous appendixes.

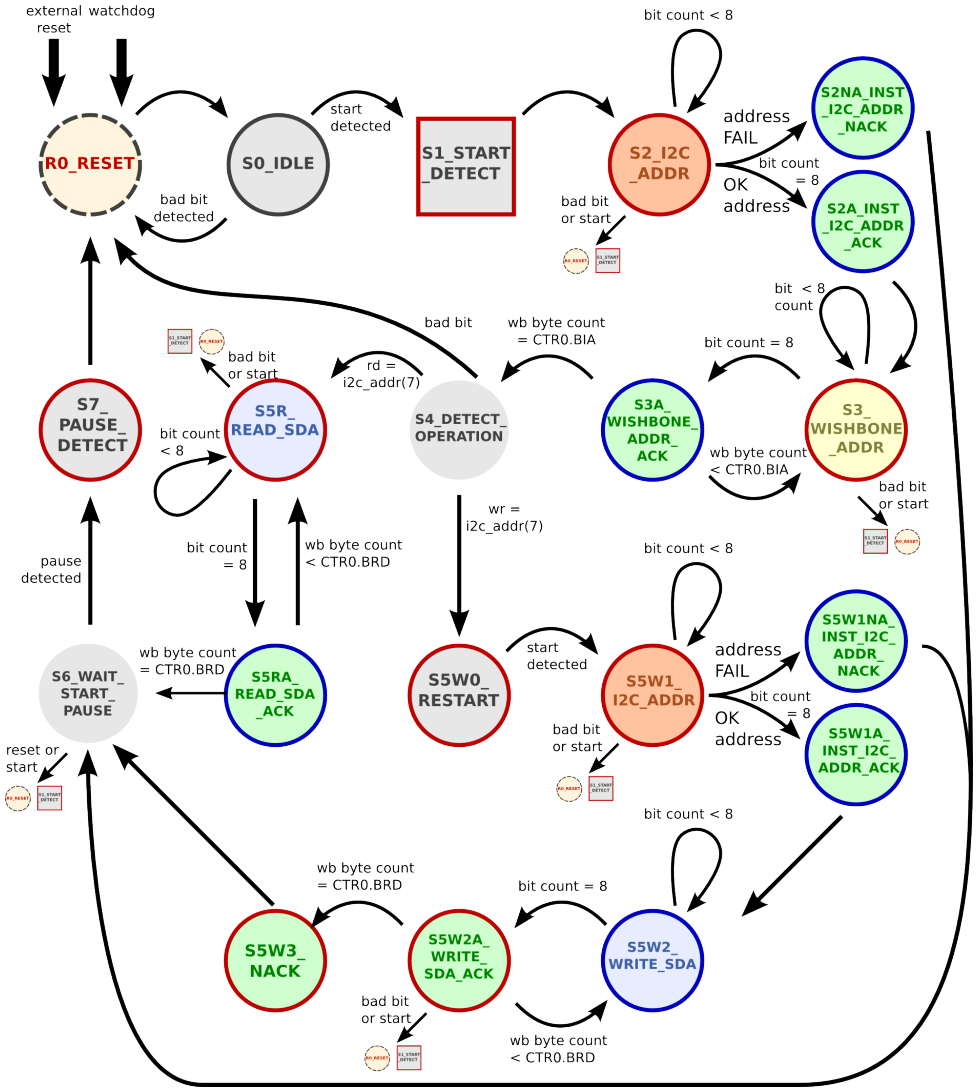


Figure 3: *i2c_slave_core* fsm