## Key Design Features

- Synthesizable, technology independent IP Core for FPGA and ASIC
- Supplied as human readable VHDL (or Verilog) source code
- Phillips® I2C-bus compliant
- User defined I2C slave address
- Configurable number of 8-bit read/write configuration registers up to a maximum of 128
- Configurable number of 8-bit read-only status registers up to a maximum of 128
- Features standard I2C register addressing common to most I2C peripherals
- Fully configurable clocking allows Standard (100kHz), Fast (400kHz) and custom data rates exceeding 20 MHz
- Compatible with a wide range of I2C master devices such as Micro-controllers and COTs ICs

## Applications

- I2C slave communication
- Inter-chip board-level communications
- Standard 2-wire comms between a wide range of I2C peripherals, micro-controllers and COTs ICs.

## Pin-out Description

| Pin name | I/O | Description | Active state |
|---|---|---|---|
| clk | in | Synchronous clock | rising edge |
| reset | in | Asynchronous reset | low |
| scl | in | I2C input SCL clock pin | as per I2C specification |
| sda | i/o | I2C bi-directional SDA data pin | as per I2C specification |
| ad_flag | out | Address register write flag | pulse high |
| wr_flag | out | Config register write flag | pulse high |
| rd_flag | out | Config register read flag | pulse high |
| ro_flag | out | Status register read flag | pulse high |
| config [num_config*8 - 1:0] | out | Configuration register output bits | data |
| status [num_status*8 - 1:0] | in | External status register input bits | data |

## Block Diagram

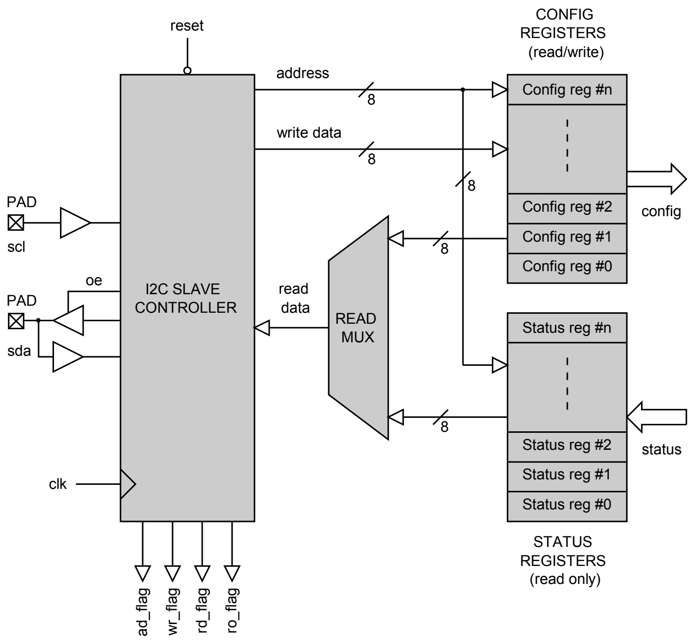

*Figure 1: I2C slave serial interface controller architecture*

## Generic Parameters

| Generic name | Description | Type | Valid range |
|---|---|---|---|
| num_config | Number of configuration registers | integer | 2 ≤ regs ≤ 128 |
| num_status | Number of Status registers | integer | 2 ≤ regs ≤ 128 |
| slave_address | 8-bit slave address of the device on the I2C bus (LSB is don't care) | std_logic vector | "0000000X" to "1111111X" |

## General Description

The I2C_SLAVE IP Core is a Philips® I2C compliant slave interface controller. The controller decodes the SCL and SDA bus signals and converts them into a simple series of 8-bit read/write commands for accessing a set of user-defined registers. These registers are defined as either *configuration* registers or *status* registers.

Config registers provide general purpose read/write bits for the control of external logic. Status registers are read-only and allow the state of external pins to be monitored via the I2C interface. Both the config and status bits are visible at the controller top-level ports.

The SCL port is an input driven by the I2C Master. The SDA port is connected to a bi-directional tristate buffer. When the I2C controller is inactive, both the SCL and SDA lines will be in a high impedance (high-Z) state.

Note: The SCL and SDA pins should have external (or internal) pullups as per the I2C specification.

The I2C slave controller is comprised of three main blocks as described by Figure 1. These blocks are the I2C Slave Controller core the Configuration register bank and the Status register bank.

### *I2C Slave Controller Core*

The slave controller core is a state-machine that continually monitors the state of the SCL and SDA lines and generates the appropriate signals on the I2C bus.

In order for the slave to function correctly the system clock frequency to SCL clock frequency must have a ratio of 8:1 or greater. That is, the following formula must be satisfied such that:

$$f_{CLK} \geq f_{SCL} * 8$$

$$f_{CLK} = system\,clock\;frequency(Hz)$$
$$f_{SCL} = I2C\,clock\;frequency(Hz)$$

To begin a data transfer, the state machine looks for a start command which is defined by a stable SCL high signal with a falling SDA line. On receipt of a start command, the slave will latch the slave address and the r/w flag over the next eight consecutive bits.

If the slave address on the bus corresponds to the generic parameter *slave_address*, then the controller will generate an acknowledge signal and a data transfer may commence. If the slave address mismatches, then the controller will not acknowledge the master and it will revert back to it's idle state waiting for the next start condition.

Once the controller has been addressed correctly, the master may send one of two possible commands to the slave. These commands are either a register write or a register read. The I2C command sequences understood by the slave controller are summarized by Figure 2 opposite.

### *Configuration and Status Register banks*

The number of configuration registers accessible by the slave controller is specified in the generic parameter *num_config*. Likewise, the number of status registers is specified in the *num_status* generic.

Configuration registers are read/write addressable, while status registers are read-only. Both types of registers are 8-bits wide with a maximum of 128 registers in each bank.

The type of register accessed is determined by the MSB of the register address that is latched in the I2C command sequence (see Figure 2). This means that the addresses from 0x00 to 0x7F access the configuration register bank and the addresses from 0x80 to 0xFF are the status register bank.

If the I2C master writes to a register address that doesn't exist then nothing will happen. Reading a register that doesn't exist will result in the value 0xFF being returned on the I2C bus.

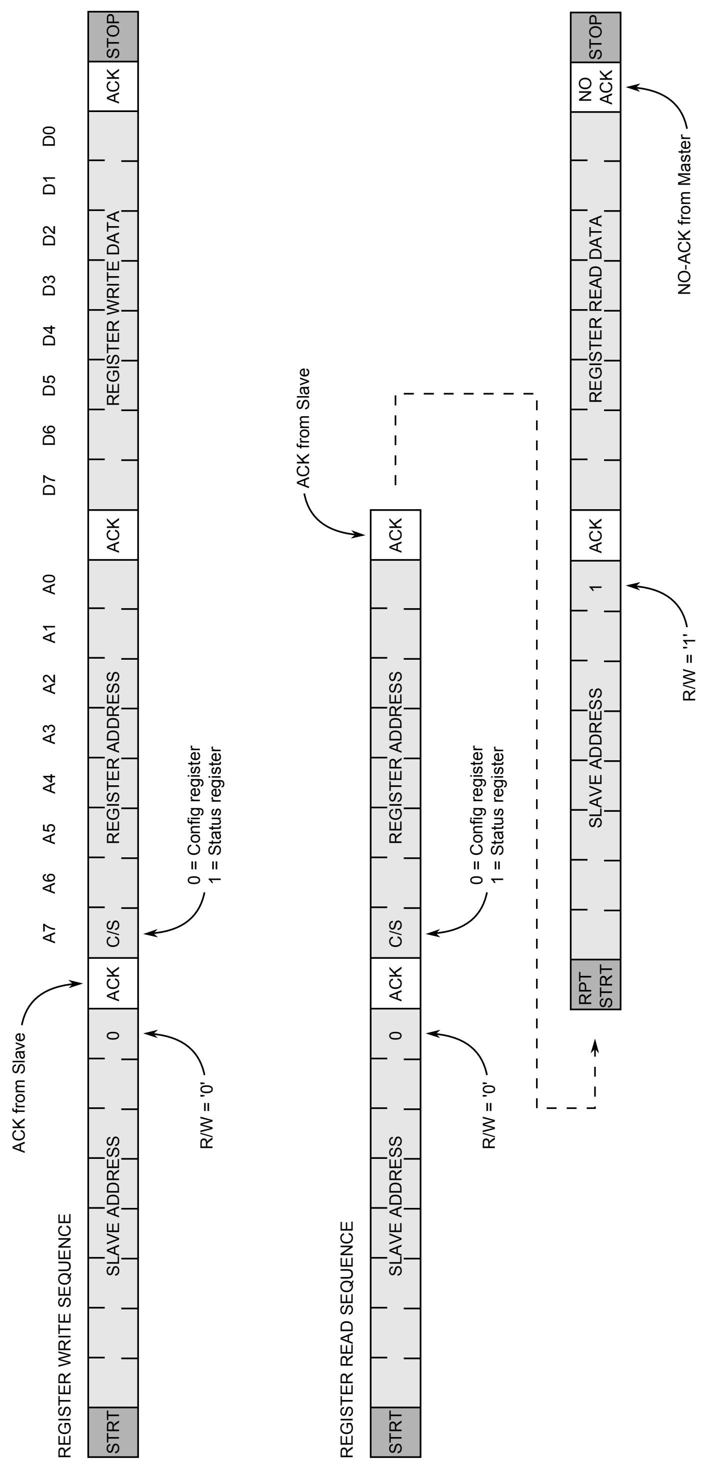

*Figure 2: I2C command sequence for a register byte write and a register byte read*

## *Functional Timing*

The slave controller IP Core accepts standard I2C bus timing. However, in order to avoid spurious START and STOP commands being decoded then the SDA line should only change on the SCL falling edge.

The SDA line should have a positive hold time (t_h) relative to the SCL falling edge. It is recommended that this hold time is equal to at least one system clock cycle. In addition, the SCL line should have a 50-50 duty cycle +/- 10%.

The I2C timing specifications are shown in Figure 3 below.

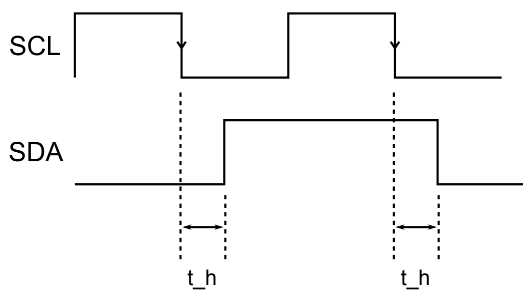

*Figure 3: I2C bus timing specifications*

## *Source File Description*

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

| ***Source file*** | ***Description*** |
|---|---|
| i2c_slave_stim.txt | Input stimulus text file |
| i2c_inbuf.vhd | Input buffer |
| i2c_iobuf.vhd | Bi-directional tristate buffer |
| i2c_config_bank.vhd | Configuration register bank |
| i2c_status_bank.vhd | Status register bank |
| i2c_slave_cont.vhd | Main I2C slave controller |
| i2c_slave.vhd | Top-level block |
| i2c_slave_file_reader.vhd | Reads the I2C bus signals from a text file |
| i2c_slave_bench.vhd | Top-level test bench |

## *Functional Testing*

An example VHDL test bench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. i2c_inbuf.vhd
2. i2c_iobuf.vhd
3. i2c_config_bank.vhd
4. i2c_status_bank.vhd
5. i2c_slave_cont.vhd
6. i2c_slave.vhd
7. i2c_slave_file_reader.vhd
8. i2c_slave_bench.vhd

The VHDL test bench instantiates the i2c_slave component together with a file-reader module that reads the I2C bus signals from a text file. The I2C slave address may be modified by changing the generic setting *slave_address* on the slave controller component. The I2C clock frequency can be modified by changing the *t_period* generic on the file reader component. The value *t_period* is a clock divider setting that specifies the system clock to SCL clock ratio.

The input text file is called *i2_slave_stim.txt* and should be put in the current top-level VHDL simulation directory.

The I2C bus signalling is split into 4 phases on 4 consecutive lines. Each line is comprised of two bits in the format 'A B' where 'A' specifies the state of the SCL line and 'B' is the state of the SDA line. The values 'A' and 'B' can either be specified as '0' (logic 0), '1' (logic '1') or 3 (tristate).

As an example, in order to send a start command, the text file should read:

```
0 1   # SCL = 0, SDA = 1
1 1   # SCL = 1, SDA = 1
1 0   # SCL = 1, SDA = 0
1 0   # SCL = 1, SDA = 0
```

To send a single bit (in this case a '1') the text file may be written as:

```
0 1   # SCL = 0, SDA = 1
0 1   # SCL = 0, SDA = 1
1 1   # SCL = 1, SDA = 1
1 1   # SCL = 1, SDA = 1
```

In addition to setting up the input stimulus file with the desired I2C commands, the user may also modify the generic parameters on the I2C slave component as required.

In the default set up, the simulation must be run for around 10 ms during which time the file-reader module will drive the I2C bus with the input commands required to write and read all the config registers followed by a read of all status registers.

The simulation generates the text file *i2c_slave_out.txt* which samples the state of the SDA line on the rising SCL clock edge. The contents of this file may be examined to verify the operation of the I2C slave controller.