

Introduction

There are thousands of I²C peripherals on the market today, ranging from data converters to video processors. The I²C bus is a good choice for designs that need to communicate with low speed peripherals due to its simplicity and low cost. Although there are other I²C bus specifications, the Philips Semiconductor I²C bus specification is the most commonly accepted I²C standard.

This reference design is intended to demonstrate how a fast, highly flexible I²C Master Controller can be constructed and utilized in an ispMACH™ device. The I²C Master Controller design was created to be a general purpose design offering a viable solution to controlling an I²C bus. It also provides a designer a foundation from which the user can customize the I²C Master Controller to meet a particular design requirement.

The I²C Master Controller was implemented in VHDL, synthesized and fitted using Lattice design tools. The design was simulated at both the Register Transfer Level (RTL) and gate level using Model Technology's ModelSim PE/Plus 5.2e. The design requires 153 macrocells and 18 I/O pins. Using an M4A-192/96-55VC or an LC4256B-5T100C yields an approximate operating frequency of 94MHz.

Applicable Documents

- Lattice Semiconductor Data Book CD-ROM
- Philips I²C Bus Specification version 1.0
- Targeting MACH[®] using Synplicity's Synplify v5.0.8 with DesignDirect-CPLD v1.0 Application Note
- Targeting MACH using Exemplar's Leonardo Spectrum v1998.2d with DesignDirect-CPLD v1.0 Application Note

Design Goals and Limitations

The following goals were considered during development of this reference design:

- I²C bus speeds of 100kbits/sec and 400kbits/sec
- I²C 7-bit addressing
- Multiple I²C masters on one I²C bus
- Up to 256-byte I²C transactions
- Interrupt mode or polling mode
- Generic microprocessor interface that supports MPU speeds up to 150MHz
- Programmable configuration registers
- Strive for compliance with the Philips 1.0 specification
- Implementation in the latest ispMACH 4A and ispMACH 4000 devices
- Hierarchical HDL design for simple user modification
- Fully automated, self-checking HDL test bench for ease of verification.

The I²C Bus Master Controller does not support the following features:

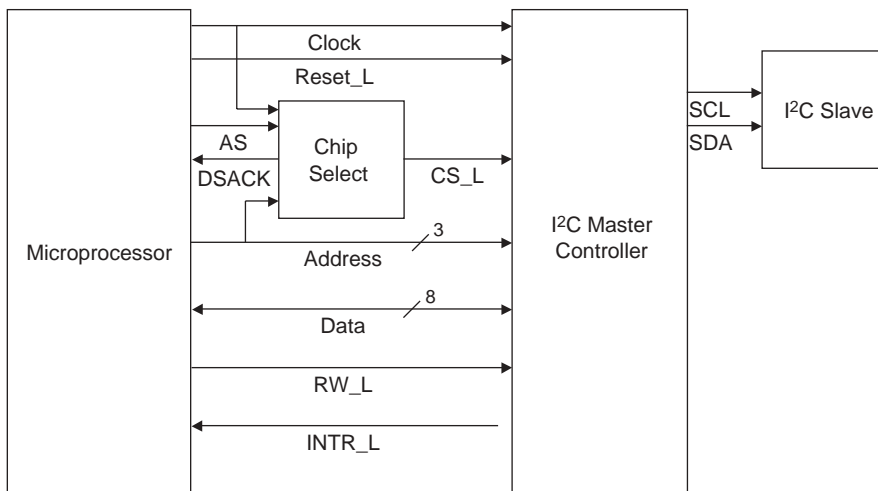
- High speed mode 3.4Mbit/sec
- Mixed speed modes on one bus
- 10-bit addressing
- Master Controller cannot be used as an I²C slave device

Theory of Operation

Overview

The I²C Master Controller is designed to interface with up to 127 different I²C slave devices. In order to accomplish this task, the I²C Master Controller requires several components to make a complete I²C bus interface system. The components required are a microprocessor, chip select unit and I²C slave devices (Figure 1). The microprocessor initiates and configures all I²C bus transactions. Next, the chip select unit assures that the microprocessor's bus cycles adhere to the I²C Master Controller requirements. Finally, the I²C slave device can be any I²C slave device operating within the specification created by Philips Semiconductor.

Figure 1. I²C Master Controller I/O Interface



Functional Description

The I²C Master Controller accepts commands from a microprocessor. These commands are decoded into I²C slave device read/write cycle transactions. The I²C bus transactions can be configured to be 1 to 256 bytes in length. Furthermore, the I²C Master Controller can operate in interrupt or polling mode. This means that the microprocessor programmer can choose to poll the I²C Master for a change in status at periodic intervals or wait to be interrupted by the I²C Master Controller when data needs to be read or written.

Table 1. I²C Master Controller Signal Descriptions

Signal Name	Signal Direction	Active State	Name - Definition
Clock	Input	N/A	Microprocessor Clock
Reset_L	Input	Active Low	System Reset
CS_L	Input	Active Low	Chip Select – Microprocessor must keep data valid on the bus as long as chip select is asserted. Data must be valid for at least three clocks.
Address	Input	N/A	Address bits reading and writing to configuration and data registers.
Data	Bi-Dir	N/A	Data bus
RW_L	Input	1 = RD 0 = WR	Indicates whether the microprocessor is writing to or reading from I ² C Bus Master Controller registers.
INTR_L	Output	Active Low	Interrupt request signal
SDA	Bi-Dir	N/A	I ² C data bus line
SCL	Bi-Dir	N/A	I ² C clock line

Microprocessor Interface Design Requirements

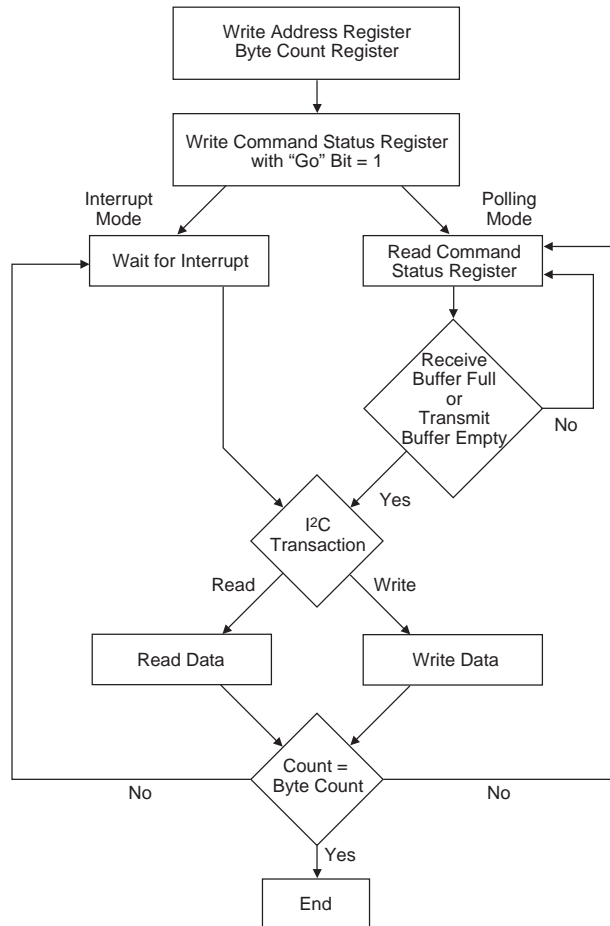
The following list contains requirements that the microprocessor must follow to ensure proper operation of the system:

- The Chip Select must be synchronized to the microprocessor clock frequency.
- Address and data must be valid the entire time chip select is asserted during a write cycle.
- Data is latched into the appropriate I²C Master Controller registers on the rising edge of the third microprocessor clock.
- Data strobe acknowledge is controlled externally to the I²C Master Controller. It is up to the designer to insert the appropriate wait states to achieve the requirements above.

Interfacing to the I²C Master Controller from a Microprocessor

There are several operations a user must undertake to interface to the I²C Master Controller. The first involves loading the appropriate I²C Master Controller configuration registers and issuing a “go” command. The configuration registers can be loaded in any order with one exception, the “go” bit should be asserted last. The registers that need to be configured are the Address Data, Byte Count and Command Status. The “go” bit may be asserted at the same time that the Command Status Register is written to. Next, the user may be required to write to the lack register. If the user is operating in interrupt mode, writing to the lack register after every interrupt will be necessary. Alternately, if the user is operating in polling mode, it will be necessary to read the Command Status Register. This is necessary in polling mode because this is how the microprocessor knows when to read or write to the I²C Master Controller. Finally, the user may be required to read from or write to the data bus, depending on the type of I²C transaction. A summary of these steps is shown in Figure 2.

Figure 2. Programming Flow for the I²C Master Controller



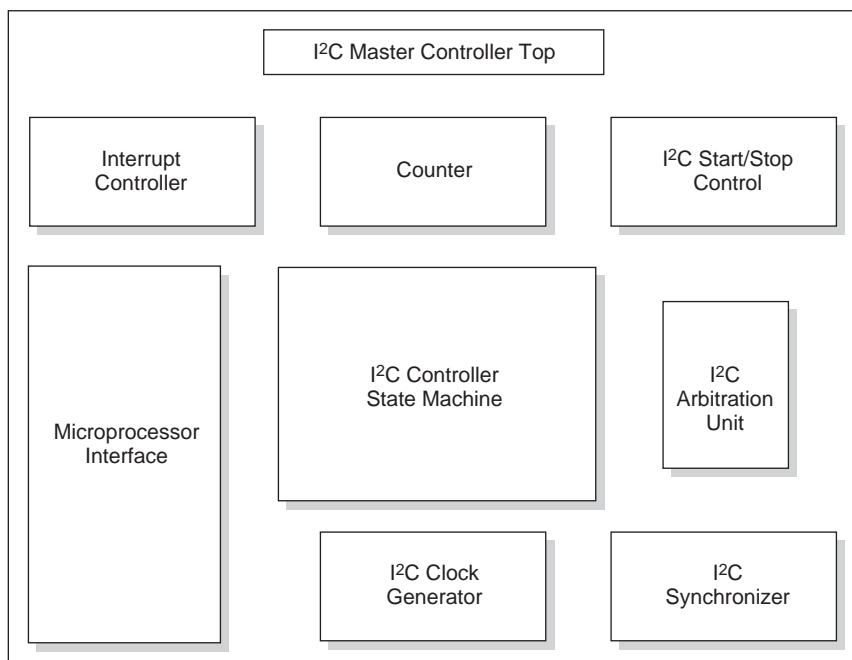
Register Transfer Level (RTL) Implementation of the I²C Master Controller Reference Design

Functional Blocks

The HDL code for the I²C Master Controller reference design contains the following functional blocks:

- I²C master top level
- Microprocessor
- Interrupt controller
- Counter
- I²C start/stop control
- I²C clock generator
- I²C synchronizer
- I²C arbiter
- I²C controller state machine

Figure 3. The I²C Master Controller Block Diagram



Register Definitions

The I²C Master Controller uses the following registers:

Data Buffer

Bit	MPU Direction	Name	Description
7-0	Write/read	Data_Buffer	Stores I ² C read/write data

Address Data Register

Bit	MPU Direction	Name	Description
7-1	Write	Low_Address_Reg	Holds the first seven address bits of the I ² C slave device
0	Write	12C_RW_Bit	I ² C read/write bit

Command/Status Register (when written to MPU)

Bit	MPU Direction	Name	Description
7	Write	Go	Starts the I ² C transaction. This is asserted after the MPU has completed writing to all configuration registers.
6	Write	Abort	Stops a current I ² C transaction.
5	—	Reserved	Reserved
4	—	Reserved	Reserved
3	Write	I2C_Mode	0 = standard mode, 100kbits/sec, 1 = fast mode, 400 kbits/sec
2	—	Reserved	—
1	Write	Trans_IE	Interrupt enable for transmission
0	Write	Receive_IE	Interrupt enable for receive

Command/Status Register (when read from MPU)

Bit	MPU Direction	Name	Description
7	Read	I2C_Bus_Busy	I ² C bus busy
6	Read	Error	I ² C transaction error
5	Read	Abort_Ack	Abort completed by I ² C controller
4	Read	Lost_Arb	Lost arbitration bit
3	Read	Done	Transaction done
2	Read	Reserved	Reserved
1	Read	Trans_Buffer_Empty	Transmit buffer empty
0	Read	Receive_Buffer_Full	Receive buffer full

Byte Count

Bit	MPU Direction	Name	Description
7-0	Write	Byte_Count	Number of data bytes for the current transaction

IACK

Bit	MPU Direction	Name	Description
0	Write	Iack	Interrupt Acknowledge

I²C Master Controller Top

This is the top level HDL block of the I²C Master Controller reference design. This block was created to instantiate all modules of the design and control the tristate drivers of the SDA and SCL signals.

Microprocessor

This module is the main interface between the microprocessor and the I²C Master Controller. This block has three main functions.

- Accepts write data from the microprocessor and latches the data into the appropriate registers.
- Controls data being sent to the microprocessor from the I²C Master Controller.
- Resets the Interrupt Acknowledge, Transmit Buffer Empty and Read Buffer Full flags at the appropriate time.

Interrupt Controller

This block generates an interrupt signal when either Transmit Buffer Empty flag or Receive Buffer Full flag are set (assuming the appropriate configuration bits are set).

Counter

The Counter module contains two counters. The first is a three-bit counter used for counting each bit of an eight-bit packet transmission or reception during an I²C bus packet transaction. The second counter is an eight-bit counter that keeps track of the number of bytes that have been written or read during the I²C transaction. The second counter increments after each byte has been written to or read from the I²C slave device. The count is then compared with the Byte Count Register. If the value is a match, the I²C Master Controller considers the transaction complete, issues a stop signal on the I²C bus, asserts the “done” flag and waits for the next transaction to be initiated from the microprocessor.

I²C Start/Stop Control

The Start/Stop Control block generates and detects start and stop events on the I²C bus. The detection of start and stop events is necessary to determine whether or not the I²C bus is in use by another master on the bus when the primary master gets a go signal from the microprocessor. Furthermore, the start detection is necessary for the primary I²C Master Controller because it cannot proceed with a transaction until the start condition has been accepted by the I²C bus. As for the start and stop generation block, the I²C Master Controller uses those blocks that generate start and stop events at the appropriate times during the I²C transaction.

I²C Clock Generator

This block generates an I²C clock signal. The I²C Master Controller can be programmed to generate a I²C clock that will run in either standard mode(100kbits/sec) or fast mode(400kbits/sec). The mode that the clock runs in depends on what the value of the mode bit in the command register. Due to the nature of the I²C bus, the actual SCL clock that is seen by all devices on the bus may not be running at the same frequency that the master requests or generates. This is because the I²C SCL clock line will run at the speed of the slowest clock being generated by a master. Please refer to the Philips I²C specification noted in the Applicable Documents section for further information regarding the I²C bus.

I²C Synchronizer

This block synchronizes the SCL and SDA signals with the system clock and protects against metastability. Normally, asynchronous signals can cause a synchronous system to become unstable by violating setup and hold times. This reference design avoids this by sending the SCL and SDA signals through two registers before using the signals.

I²C Arbiter

This reference design supports the multi master feature of the I²C bus specification. The Arbiter block contains the code necessary to perform arbitration on the I²C bus.

I²C Main State Machine

The Main State Machine block controls most of the other blocks and serves as the interface between the I²C Master Controller and the I²C bus. This block contains three state machines and many control signals. The three state machines are: main, read and write. The main state machine controls the other two and is where the I²C transactions begin. The read and write state machines shift I²C data into or out of the appropriate registers. All three state machines are one hot encoded and are therefore more efficient at driving the various output control signals.

Address Map

The microprocessor can control the I²C Master Controller by writing to and reading from configuration registers. The I²C Master Controller configuration registers are located at the following addresses:

A2A1A0	Description
000	Data buffer
001	Address data register
010	Reserved
011	Reserved
100	Command status register
101	Byte Count
110	Interrupt Acknowledge

HDL Verification

Overview

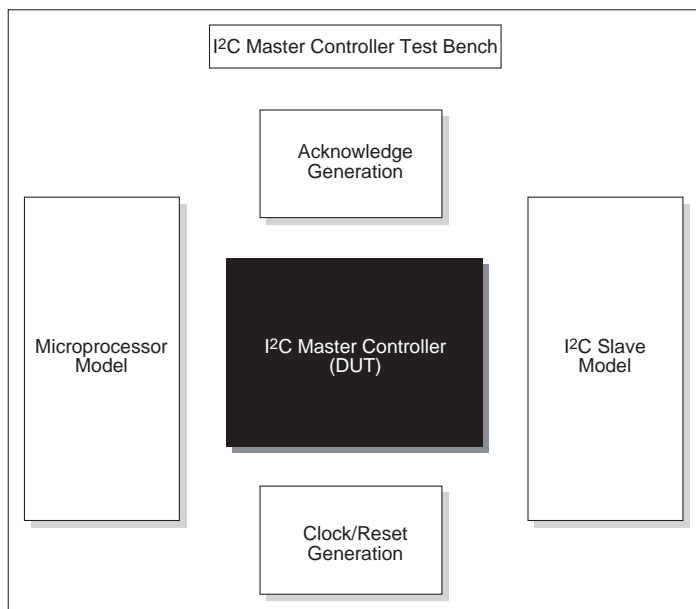
This test suite provides a way of verifying the functionality of the I²C Master Controller reference design at the Register Transfer Level (RTL) as well as the gate level. The test suite was designed to test one- and two-byte I²C read and writes utilizing both interrupt and polling modes. This is accomplished by calling the appropriate functions in the microprocessor model. The microprocessor model initiates all I²C transactions. The I²C Slave model emulates an I²C serial EPROM and responds to the requests of the I²C Master Controller.

Functional Blocks of the HDL Test Suite

The HDL code for the I²C Master Controller reference design test suite contains the following functional blocks:

- I²C Master Controller test bench
- Microprocessor model
- I²C slave model
- Clock/reset generation block
- Acknowledge generation block
- The device-under-test (I²C Master Controller)

Figure 4. The I²C Master Controller Test Suite



I²C Master Controller Top Level Test Bench

This is the module that instantiates all the other modules of the test suite. The HDL Test Bench is a fully automated and self-checking test environment.

The Device-Under-Test (The I²C Master Controller)

The Device-Under-Test (DUT) is the I²C Master Controller reference design. The DUT may have slightly different port names for gate level simulation and therefore the top level test bench may need to be modified to reflect this. The test suite comes with scripts to run Model Technology's ModelSim in either the RTL or gate level.

Microprocessor Model

The microprocessor model is the part of the test suite that simulates the microprocessor in an I²C Master Controller system and is where the designer can issue commands to the I²C Master Controller. This module was used to initiate a two-byte read or a two-byte write I²C transaction using both interrupt and polling modes. The module is made up of several tasks that can be easily reconfigured to stimulate a variety of other I²C bus transactions.

I²C Slave Model

The I²C slave model simulates an I²C Slave serial EPROM. This part of the test suite provides the I²C Master Controller with an I²C slave device with which to communicate.

Clock/Reset Generation Block

The Clock/Reset block generates the system clock. It also generates an initial reset when the simulation starts. The module has different clock parameters that can easily be interchanged to generate different system clock rates.

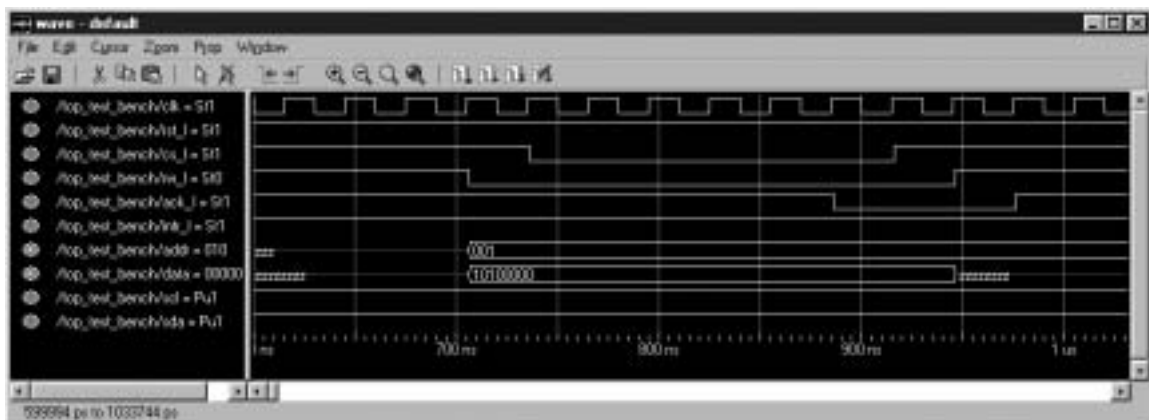
Acknowledge Generation Block

The acknowledge block is used to generate and send an acknowledge signal back to the microprocessor model four clock cycles after every microprocessor transaction. This is necessary because without this block, the microprocessor may not be able to terminate its bus cycle. The functionality of this module may need to be modified or duplicated in the Chip Select unit of the I²C Master Controller system to ensure proper timing and operation of a particular microprocessor.

I²C Master Controller Transaction Waveforms

The following waveforms illustrate the typical operation of the I²C Master Controller reference design implemented in an M4A-192/96-55VC. All the waveforms were taken from RTL Simulations using Model Technology's ModelSim Simulator. Lattice design tools were used to export a gate level HDL netlist and an SDF file. These files were then used to simulate the I²C Master Controller at the gate level. The gate level waveforms are not shown because of symbol renaming.

Figure 5. Microprocessor Write Cycle to Low Address Register



Technical Support Assistance

Hotline: 1-800-LATTICE (Domestic)
1-408-826-6002 (International)
e-mail: techsupport@latticesemi.com