DESIGN AND IMPLEMENTATION OF I2C BUS CONTROLLER USING VERILOG

1 MR. J. J PATEL, 2 PROF B. H. SONI
1 P.G. Student, E.C. Dept., 2 Faculty, E.C. Dept.,
1, 2 U.V. Patel College of Engineering, kherva
jatinpatel01@gmail.com

ABSTRACT: The I2C protocol was given by Philips Semiconductors in order to allow faster devices to communicate with slower devices and also allow devices to communicate with each other over a serial data bus without data loss. I2C enabled microcontrollers like PIC18F452 from Atmel and TMS470 from Texas Instruments requires a lot of programming and knowledge of the register structures for configuring it. Hence they are not portable. We here present a model of I2C bus controller. The I2C controller is designed using verilog in xilinx 11.1 and targeted in virtex 4 and virtex 5.

Index Terms — I2C Bus, master, slave, verilog, Finite state machine

I. INTRODUCTION
The EEPROM, ADC and RTC will require an interface for communication between them. So I2C bus is used as an interface between them. It is used to minimize system-level interconnect. This simplifies the system level design and the design of the motherboard and associated chip-boards. Moreover transmitting information over the I2C bus will improve system performance since the transmission of digital data is much less susceptible to interference from environmental noise sources.

II. I2C BUS SPECIFICATIONS
The I2C Controller Bus is a two-wire, bi-directional serial bus that provides a simple and efficient method of data transmission over a short distance between many devices. I2C provides good support for communication with various slow, on-board peripheral devices that are accessed intermittently, while being extremely modest in its hardware resource needs. It is a simple, low-bandwidth, short-distance protocol. I2C is easy to use to link multiple devices together since it has a built-in address [1][2]. The two I2C signals are serial data (SDA) and serial Clock (SCL) as shown in Figure 1. The device that initiates a transaction on the I2C bus is termed the master. The master normally controls the clock signal. A device being addressed by the master is called a slave [1][2].

The I2C protocol supports multiple masters, but most system designs include only one. There may be one or more slaves on the bus. Both masters and slaves can receive and transmit data bytes. Standard I2C devices operate up to 100Kbps, while fast-mode devices operate at up to 400Kbps. Most of the I2C devices available today support 400Kbps operation. Higher-speed operation may allow I2C to keep up with the rising demand for bandwidth in multimedia and other applications[2].

III. I2C CHARACTERISTICS
The SDA and SCL lines are bi-directional lines connected to a positive voltage supply through a pull-up resistor. The bus is free when these lines are ‘high’. The data on the SDA line is valid only when the SCL line is ‘high’. Change of data is allowed when SCL line becomes ‘low’. During data transfer, the master generates the START and STOP conditions, which are unique conditions and are shown in Figure[2].

Fig. 1 I2C bus configuration using masters and slaves

Fig. 3 Data transfer on the I2C bus
IV. I2C BUS CONTROLLER DESIGN

I2C protocol consists of four parts 1) START signal generation, 2) Slave address transfer, 3) Data transfer, 4) STOP signal generation [12].

![I2C Master Controller Block Diagram](image)

Fig. 4 The I2C Master Controller Block Diagram

V. I2C BUS ARCHITECTURE

![I2C bus architecture](image)

Fig. 5.1 I2C bus architecture

It consists of a master and a slave connected using I2C bus. The SDA line is connected to a positive supply voltage through a pull up resistor. The SCL line is not connected to a pull up resistor as there is only one master.

**Designing the I2C master controller**

For designing the I2C master controller is to use a finite state machine (FSM). We can easily implement finite state machine by writing VHDL or verilog code [15].

**Finite State machine**

A finite state machine is a sequential circuit that uses a finite number of states to keep track of its history of operations, and based on this history and its current inputs, determine what to do next.

VI. RESULT ANALYSIS

Here the I2C bus controller is designed using verilog and implemented in virtex 4 xc4vlx15-12sf363 using xilink 11.1. The device utilization summary is given below:

<table>
<thead>
<tr>
<th>Device utilization summary(virtex 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic utilization</td>
</tr>
<tr>
<td>Used</td>
</tr>
<tr>
<td>Available</td>
</tr>
<tr>
<td>Utilization</td>
</tr>
<tr>
<td>Number of slices</td>
</tr>
<tr>
<td>Number of slice flipflops</td>
</tr>
<tr>
<td>Number of 4 input LUTS</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
</tr>
<tr>
<td>Number of GCLKS</td>
</tr>
</tbody>
</table>

Here the I2C bus controller is designed using verilog and implemented in virtex 5 xc4vlx15-12sf363 using xilink 11.1. The device utilization summary is given below:

<table>
<thead>
<tr>
<th>Device utilization summary(virtex 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic utilization</td>
</tr>
<tr>
<td>Used</td>
</tr>
<tr>
<td>Available</td>
</tr>
<tr>
<td>Utilization</td>
</tr>
<tr>
<td>Number of slices</td>
</tr>
<tr>
<td>Number of slice flipflops</td>
</tr>
<tr>
<td>Number of 4 input LUTS</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
</tr>
<tr>
<td>Number of GCLKS</td>
</tr>
</tbody>
</table>
VII. CONCLUSION

The result shows that minimal resources are utilized in virtex 4. The design process is simplified using verilog HDL to design the I2C bus controller. The designer can write his design description without choosing any specific fabrication technology. If a new technology emerges, designers do not need to redesign the circuit. He simply input the design program to the logic synthesis tool and creates a new gate level netlist using the new fabrication technology. The logic synthesis tool will optimize the circuit in area and timing for the new technology.

VIII. REFERENCES:

[3] I2C bus Inter Integrated Circuits bus by Philips Semiconductors