International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 26085-26091

http://www.ripublication.com

Design and Verification of Configurable I^2C Subsystem Bus

Jaladi Chaitanya[1], V. Narasimha Nayak[2], B. Murali Krishna[3]
[1, 2, 3] *Electronics and Communication Engineering, K.L University, Vaddeswaram, Guntur, A.P, India.*
[1] *chaitanyajaladi@gmail.com*

Abstract

Serial communication is used for all long distance communication and most computer networks, where the cable's cost and difficulties in synchronization makes parallel communication impractical. Serial computer buses have become more common even at shorter distances, as signal integrity and transmission speeds improved in newer serial technologies have begun to outweigh the parallel bus's advantage of simplicity. This paper presents serial data communication implementation using I^2C (Inter Integrated Circuit) slave subsystem bus with configurable addressing modes. In this both the addressing modes are configured into a single design. Configurable I^2C subsystem design has done in verilog HDL and verification has done in SV (System Verilog) and UVM (Universal Verification Methodology). Environments of SV and UVM act as masters for I^2C - slave subsystem bus.

Index Terms: - I^2C, Addressing mode, System Verilog and UVM.

Introduction

Coming to the serial communication [2] point of view, there are protocols like SPI (Serial peripheral interface), RS-422, RS-232, RS-485, and FireWire for interfacing high speed and low speed peripherals. Pin requirement in these protocols is more in the IC (Integrated Circuit) for serial data communication implementation, as the physical size of IC have decreased from the past 10 to 20 years, serial data transfer require less amount of pin connection. USB/SPI and mostly UARTS are all just 'one point to one point' data transfer bus systems. These protocols use multiplexing of the data path and forwarding of messages to service multiple devices. To overcome this problem, Phillips introduced the I^2C protocol, where communication requires only two lines with two or more chips and can control a network of device chips with just a two general purpose I/O pins whereas, coming to other bus protocols we require more signals and pins to connect devices.

It is always better to use I^2C in case of if you want to connect more devices to the bus. As we know that with I^2C 7-bit addressing mode we have 128 no. of possible addresses. That means 128 devices can be connected to single system at a time. 10-bit addressing expands the number of possible addresses. By combining this two addressing modes and configuring to a single design we can use any addressing mode as per the requirement of devices that are connected.

In this paper configurable I^2C subsystem acts as slave and masters are SV and UVM environments. It is proved to be the best bus in the case of control applications, like devices may have to be added or removed from the system. We can also use I^2C Protocol for communication between multiple circuit boards in equipments with or without using a shielded cable depending on the distance and speed of data transfer.

The I^2C-bus protocol:

I^2C is a two wire[7][2], bidirectional serial bus that acts as data communication between two devices. I^2C bus can interface many devices where each device contains a unique address which acts as slave. SCL and reset are the input lines which are used to initiate the bus controller process. The R/WB signal is given as an input to indicate whether transmitter is master or slave in the data transmission.

The physical I^2C bus consists of only two wires, SCL and SDA. SCL is the serial clock line; it is used to synchronize all data transfers over the I^2C bus. SDA is the serial data line; the SCL and SDA lines are connected to all devices for communication on the I^2C bus. The I^2C bus is idle when both SCL and SDA are at logic 1 level. The bus will be busy after the START condition. Once the start condition has been sent, a byte will be transmitted by the MASTER to the SLAVE through SDA and SCL lines.

This first byte after a start condition [9] will identify the slave on the bus (address) and last bit of byte will select the mode of operation. The meaning of all following bytes depends on the slave. The slave device whose address matches the address that is being sent out by the master will respond with an acknowledgement bit on the SDA line by pulling the SDA line low. Data transformation is done in packets of 8 bits. The bits transmitted on the SDA line starts with the MSB (Most Significant Bit). If read or write data transfer direction bit is low then slave will receive data form master which is write operation to slave. For every byte transferred, the slave device receiving the data from master sends back an acknowledge bit (ackout), so there are actually 9 SCL clock pulses to transfer each byte of data. To complete the transfer master will send not acknowledge to the slave. Then the STOP condition will be generated, where the SDA line is driven low while SCL line is high. This indicates the end of the transaction with the master device.

On the other hand while transferring [8] the data (R/WB is high) the slave will send the data to the master and master will respond by an acknowledgement (ackin) signal which is send to the slave. After the START condition (S), slave receives a slave address. This address is 8bit long where last bit represents R/WB which is data direction bit. Active low of R/WB represents write(receiving from the master) operation and active high represents read(transmission to the master) operation. A data transfer is terminated by a STOP condition (P) every time which is generated by

the master. However, if a master still wishes to continue the communication on the bus, it can generate a repeated START condition (Sr) and address another slave without generating a STOP condition after not ack signal. Various read/write combination formats are then possible within such a transfer.

Where as in 10 bit addressing mode two bytes of address decide the slave device. Last bit of the first byte after the Start condition determines the slave access mode ('1' = read / '0' = write). After the transfer of first byte address all the slave devices will send the acknowledgement to the master. Second byte of slave address will decide which slave will be selected.

Proposed approach:

In general I^2C bus protocol means everyone knows 7-bit addressing mode. But if we want to interface more devices a few will use 10-bit addressing mode.

Instead of using two separate designs for two addressing modes of single bus protocol configurable I^2C will combine both of the addressing modes into single design. We can use any of the addressing modes as per the requirement by just selecting addressing mode select signal. If mode select signal is low 7-bit addressing mode will be selected else 10-bit will be selected. The FSM in fig.1 explains the operation of configurable I^2C design.

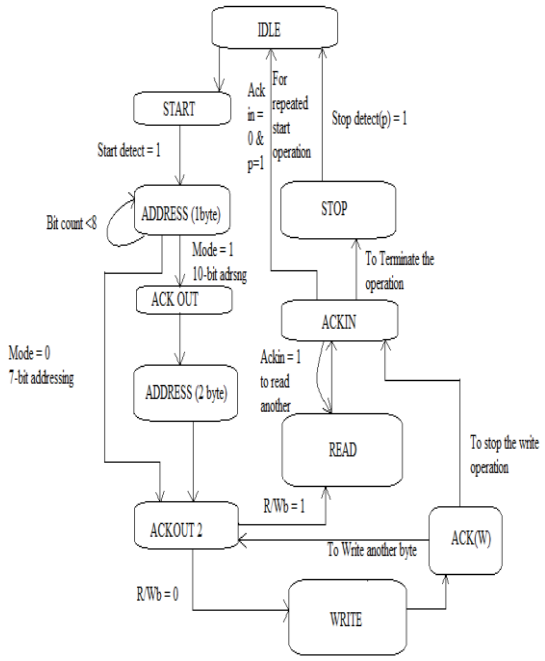

Figure 1: FSM for Configurable I^2C subsystem bus

Test Bench for I^2C Subsystem bus in System Verilog (SV):

SV[6] contains separate environment structure unlike verilog HDL contains test bench module to test the DUT.

Components in SV:

1. Transaction class
2. Generator class
3. Interface
4. Driver class
5. Monitor class
6. Score board class
7. Environment class and
8. Top module

Transaction class:
In transaction class we will declare variables. Like addressing mode, address and data to randomize

Generator class:
Generates the random data for the variables declared in the transaction class. This randomized stimulus will be sent to driver class.

Driver class:
This is also called bus function model. Here the values sent by the generated class will be driven to the DUT (Design Under Test). The generator and driver are communicated through mail box. Driver will drive the inputs to the DUT using virtual interface because classes are dynamic in nature whereas DUT is static.

Monitor class:
Monitor class will collect the outputs from the DUT. These two are also connected through virtual interface.

Score board class:
In driver and monitor we will collect the expected outputs and outputs collected from DUT respectively and compare them in score board.

Environment class:
Environment class will contains all the classes we discussed above.

Top module:
Top module will contain environment class and DUT instantiation.

I^2C Subsystem bus Test Bench in UVM (Universal Verification Methodology):
The **Universal Verification Methodology** (UVM) is a standardized methodology for verifying integrated circuit designs[5]. Uvm components, uvm env and uvm test are the three main building blocks of a test bench in uvm based verification. Fig 2 shows the environment of UVM test bench

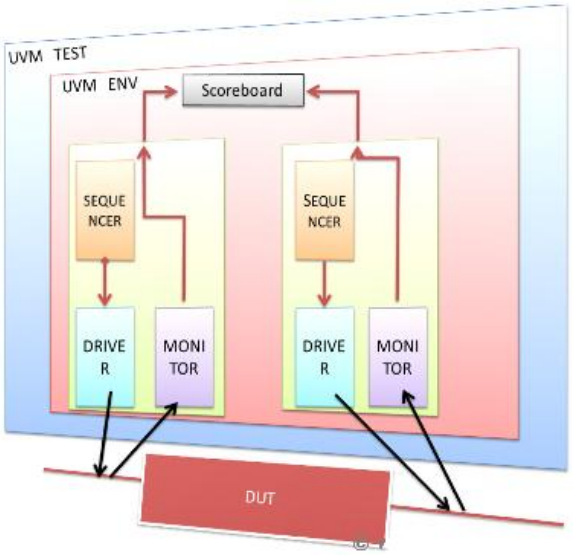

Figure 2: UVM Environment

In uvm sequencer[1], driver and monitor are main components. These three are combined to form an agent. Two main components in uvm are active and passive agents. Sequencer, driver and monitor are active in active agent. Where as in passive agent only monitor will be active and remaining two will be idle.

Sequencer will have randomized data. Driver . Driver will drive data to DUT and monitor will collect the inputs of driver if it is active agent.

In passive agent monitor will collect the outputs from DUT. Driver and sequencer are passive.

Scoreboard will compare both active and passive agent's monitors.

Results:

Coverage report results:

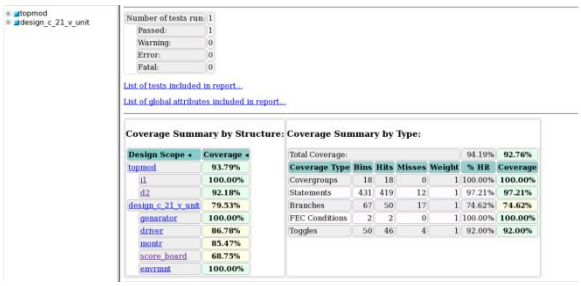

Figure 3: Functional coverage report of configurable I^2C subsystem bus

Wave forms:

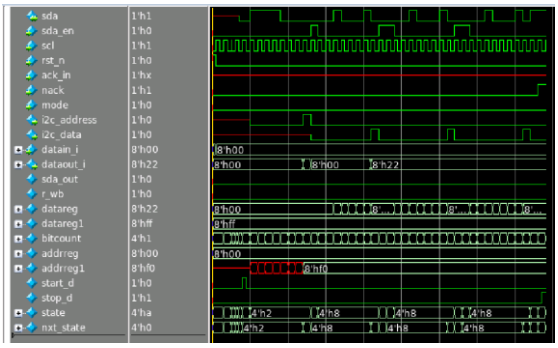

Figure 4: 7-bit write operation

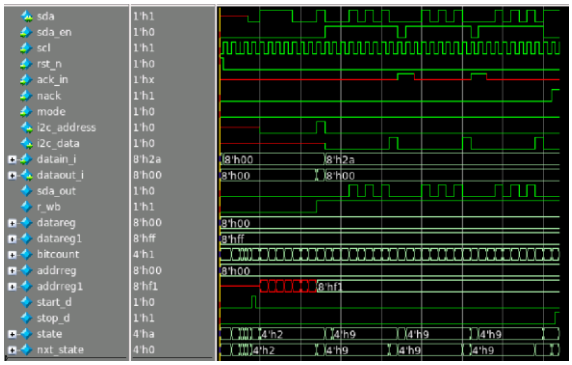

Figure 4: 7-bit read operation

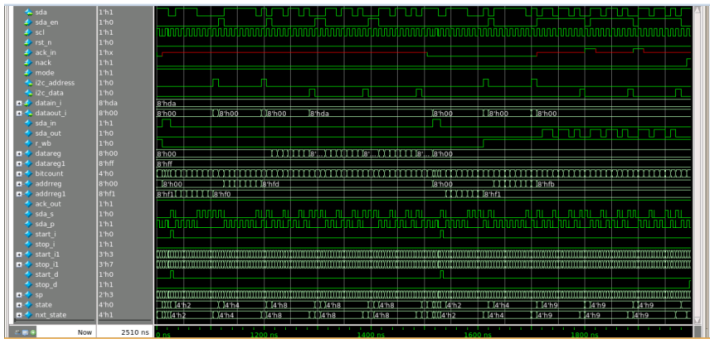

Figure 5: 10-bit write and read operations

Comparison Table

7-Bit addressing mode	10-Bit addressing mode
128 devices can be interfaced at a time	1024 devices can be interfaced at a time
Speed of the bus is more	Speed of the bus is low compared to 7-Bit

Conclusion

This paper demonstrates how configurable I^2C slave subsystem is designed by combining both 7-bit and 10-bit addressing modes using mode select signal and how writing into it and reading from it has done. So that we can increase no of peripherals devices that are interfaced using I^2C bus protocol as slave.

References

[1] Juan Francesconi, J. Agustin Rodriguez, Pedro M. Juli´ "UVM Based Test bench Architecture for Unit Verification" Argentine School of Micro-Nanoelectronics, Technology and Applications IEEE 2014.

[2] Bollam Eswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh "Implementation of I^2C Master Bus Controller on FPGA", International conference on Communication and Signal Processing, IEEE April 3-5, 2013.

[3] Antonio José Salazar Escobar, José Machado da Silva, Miguel Correia "An I2C Based Mixed-Signal Test and Measurement Infrastructure" IEEE 2014.

[4] MilyJashank, Sureshchandra J Gupta, Jayalakshmi M Nair "Analysis of Environmental Temperature Variation as a Precursor of Earthquake Prediction Using I2C" IEEE 2014.

[5] Accellera, UVM 1.1 User Guide, 2011.

[6] C. Spear, SystemVerilog for Verification, Second Edition: A Guide to Learning the Testbench Language Features, 2nd ed. Springer Publishing Company, Incorporated, 2008.

[7] NXP Semiconductors, I^2C-bus specification and user manual, 2012.

[8] *I^2C Bus Specification*, Philips Semiconductor, version 2.1, January 2000.

[9] http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/general-introduction/i2c-bus-protocol.html