Research Paper Engineering

Systemc Model Based I2c Testing Using Socket Programming

Sudhir Rajput	Department of TE, P.G. Scholar, R.V. College of Engg, Mysore Road, Bengaluru560029, India
Rathina Balan Thalaiappan	Senior Staff Engineer, Intel Mobile Communications,Bengaluru 560066, India
Tamal Saha	Software Development Engineer, Intel Mobile Communications,Bengaluru 560066, India
T.P. Mithun	Department of TE, Asst. Professor, R.V. College of Engg, Mysore Road, Bengaluru560029, India

ABSTRACT

In this paper we propose a method to test a device driver even before the fully functional virtual platform is available for testing. We setup an Intel modem baseband chip build environment and using socket programming test the SystemC model of I2C. SystemC model gives the same behavior as the hardware. All the target test cases are directly executed on the SystemC model. This speeds up the driver development process and maximum possible verification of driver is done even before the hardware is ready. I2C is a multi-master, multi-slave, single-ended serial communication bus, it is used for connecting peripherals to processors in embedded systems. Socket programming is used for the communication between the I2C driver and the SystemC model. SystemC is a set of C++ classes and macros which are used for hardware simulation.

KEYWORDS I2C, Device driver, SystemC, Socket programming

INTRODUCTION

A smart phone has two processors, one for the GSM protocol stack called baseband processor (BP) and another general purpose processor for the user interface and application, this processor is also known as application processor (AP) [1]. Internal architecture of a smart phone is shown in figure 1.

The baseband processor of almost all modern smart phones is a System-on-Chip (SoC). The baseband processor cores have the typical set of peripherals, such as RTC, UARTs for RS232 and IRDA, SPI, I2C, SD/MMC card controller, keypad scan controller, USB device etc.

Inter-Integrated Circuit (I2C) is a serial data transfer protocol [2]. I2C is a simple, bi-directional half duplex bus. It provides a protocol that allows devices to communicate to each other via two wires. One line is used for data transfer (SDA) and other for clock synchronization (SCL).

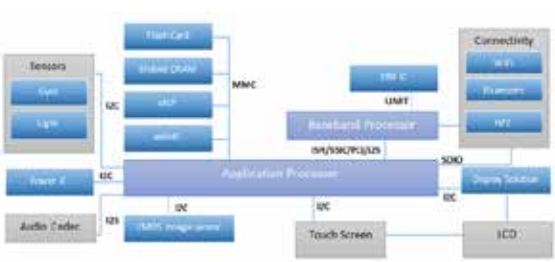

Figure 1: Internal architecture of a smartphone.

A SystemC model of I2C is developed to test the functionality of the I2C driver software used in the smart phone. This SystemC model replaces the device emulator which was used previously for host testing. The communication between the SystemC model of I2C and test cases is done using socket programming.

The important specifications of I2C bus are described in Section II and specifications of socket programming are described in Section III. The process methodology is described in Section IV and simulation results are described in section V. Section VI contains conclusion and future scope.

I2C BUS SPECIFICATIONS

The I2C bus is built around a two-wire serial bus, serial data (SDA) and serial clock (SCL), these two lines carry information between the devices connected to the bus. Each device connected to I2C is recognized by a unique address and can operate in master mode or in slave mode [3]. Master initiates a transfer and generates the clock for the same and the device addressed by the master is called as slave. If two or more devices tries to transmit data at the same time then the winning master is recognized using arbitration process.

- Features of I2C

- All the devices connected to the bus have unique addresses and a simple master/slave relationship always exits; master can operate as master transmitter or as master receiver.
- Only two lines SDA and SCL are required.
- I2C bus has collision detection and arbitration logic to prevent data corruption if two or more devices initiate data transfer simultaneously.
- It supports 3 modes of operation: Standard mode with data transfer up to 100 kbits/s, Fast mode with data transfer up to 400 kbits/s and High-speed mode with data transfer up to 3.4 Mbit/s.
- To preserve data integrity on chip filtering is done to reject the spikes on data bus.
- The maximum numbers of ICs that are connected to the bus are limited by the maximum bus capacitance of 400 pF.

- I2C Characteristics

The SDA and SCL lines are connected to a positive voltage supply using pull-up resistors. The bus is free when SCL and SDA lines are 'high'. Master generates a start and stop condition during data transfer as shown in the fig 2.

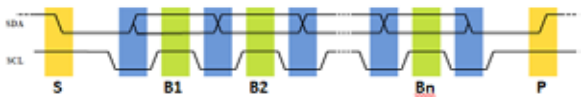

Figure 2:The start and stop condition.
Start condition: A high to low transition on SDA when SCL remain high, represented by 'S' in fig.2

Stop condition: A low to high transition on SDA when SCL is high, represented by 'P' in Fig.2

Data on SDA is valid only if SCL is high. B1, B2…Bn represents the valid bits.

- Data Transfer

Data is transferred after the start condition, the transmission is byte oriented and every data is placed on SDA line. Master frees the SDA line after transmission. The complete data transfer is shown in Fig3. Bytes are transferred MSB first.

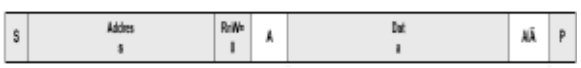

Figure 3: The complete data transfer operation.

After start condition address of the slave is sent and after slave address RnW bit is sent which determines the direction of data. Slave send an acknowledge bit 'A' to master, after this data is sent or received depending upon the RnW bit. If RnW bit is 1 then master reads data from slave and if 0 then master writes data to the slave.

- **Arbitration**

When two or more masters may generate a defined start condition simultaneously then arbitration logic is used to allow multi master mode. Arbitration takes place on SDA line when SCL line is high. When a master transmits a low level and another transmits a high level then due to wired AND functionality low level dominates the high level, as a result device that was transmitting a high level switches off.

SOCKET PROGRAMMING

Sockets are used for inter-process communication across a computer network. Network sockets are used and controlled by socket application program interface (API). Berkeley sockets standard is used in most of the internet sockets [4]. A socket address is formed by combining IP address and port number. Socket address is used to deliver the packet to the appropriate application process or thread.

- Types of sockets
- Datagram socket: It is connectionless network socket. Packet using datagram socket are individually addressed and routed.
- Stream socket: It is a connection oriented network socket. The data is received in a sequence. It uses a well-defined mechanism for creating and destroying connection. It is implemented on TCP so that application can run on networks using TCP/IP protocol [5].
- Raw socket: It is an internet socket. It allows direct sending and receiving of data without any protocol specific transport layer formatting.

- Connection of the Client / Server system

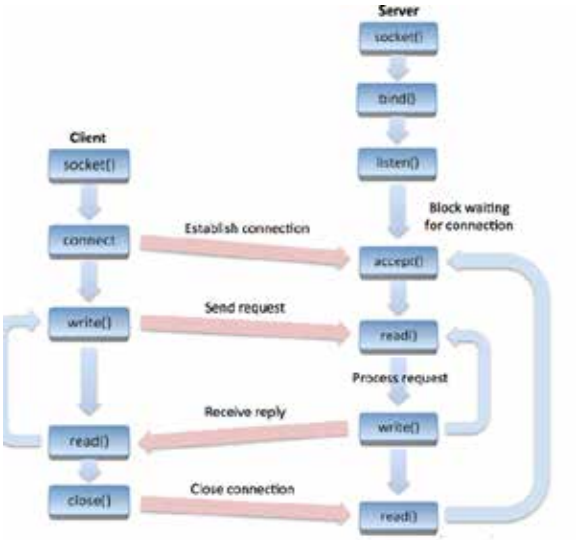

Figure 4:Connection establishment steps

Server and client have to open the sockets with same socket address. Server should bind to a particular socket address, and start listening for connection from client. Client socket uses connect() function to connect to the server and server uses accept() function to accept the connection from the client. Read() and Write() functions are used for data transfer between the server and client. In this paper we use stream sockets to establish a connection between client and server. Connection establishment steps are shown in Fig. 4.

PROCESS METHODOLOGY

A device emulator is used for host testing, but it does not support full functionality of the device driver. In the proposed method we replace the device emulator with the SystemC model.SystemC model behaves as the server and client socket is integrated with the source code of the driver. Using SystemC in place of device emulator reduces the difference between host and target test cases.

Architectural block diagram of the implemented communication between the SystemC model of I2C and test cases is shown in the Fig.5

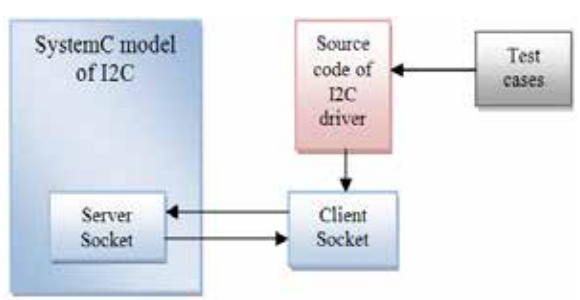

Figure 5:Communication between SystemC model and test cases.

The test cases of I2C driver are executed using source code. The test cases are loaded to Generic module test suit (GMTS) so that they can be executed in an order. A client socket is written to establish connection to server socket (SystemC model) using socket programming.

The steps for making a connection with server socket are shown in the Fig.6.

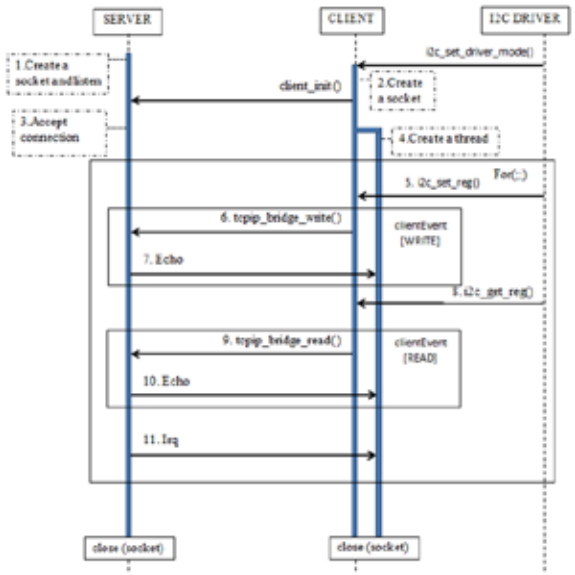

Figure 6: Connection establishment between test suit and SystemC model

The steps for making connection between SystemC model and test cases are:

- The first step is to call the socket() function at the server, specifying the type of communication protocol (TCP based on IPv4), and call listen() function to listen for an incoming connection
- ♦ Socket type - Stream socket.
- ♦ IP address "127.0.0.1"
- ♦ Port number "1234"

- A socket is created at the client side andconnect() function is called to establish a connection with a TCP server.
- Accept () function is called at the server side to accept the pending connections on socket. It then creates and returns a handle to the new socket.
- A thread is created at the client side to continuously receive data simultaneously with main thread. An infinite for() loop is used to receive data.
- I2C driver initiates write operation using i2c_set_reg() function, it provides the value and the address of the memory location.
- tcpip_bridge_write() function uses the information provided in previous step to write value to the specified memory location.
- An echo of the message received at server, is sent back to the client, this echo message is received by the client using receive() function. The message is compared with "Read", "Write" and "irq", and the write_event is signaled.
- I2C driver initiates read operation using i2c_set_reg() function, it provides the address of the memory location from where value is to be read.
- tcpip_bridge_read( ) function uses the information provided in previous step to read value from the specified memory location.
- An echo of the message received at server, is sent back to the client, this echo message is received by the client using receive( ) function. The message is compared with "Read", "Write" and "irq", and the read_event is signaled.
- An interrupt may occur at the server side any time, this is received at client side and respective interrupt service routine is run.

RESULTS

Fig. 7 shows the server socket is bind to the IP address '127.0.0.1' and port number '1234'. The server socket is waiting for a connection request from the client side.

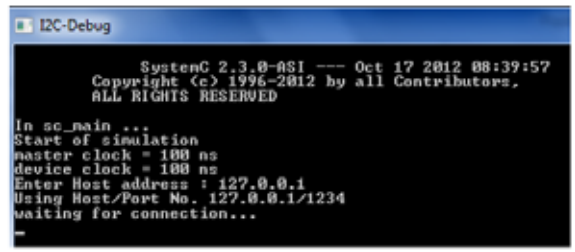

Figure: 7 Server socket

Server socket is connected to the clientsocket and read write operations are happening this is shown in Fig. 8.

Interrupts from the server side are injected to the client side as shown in the Fig. 8.

The test cases are executed using GMTS, we can see the test cases which are passing and which test are failing. A report of all the test cases can be generated using GMTS. Fig. 9 shows the GMTS.

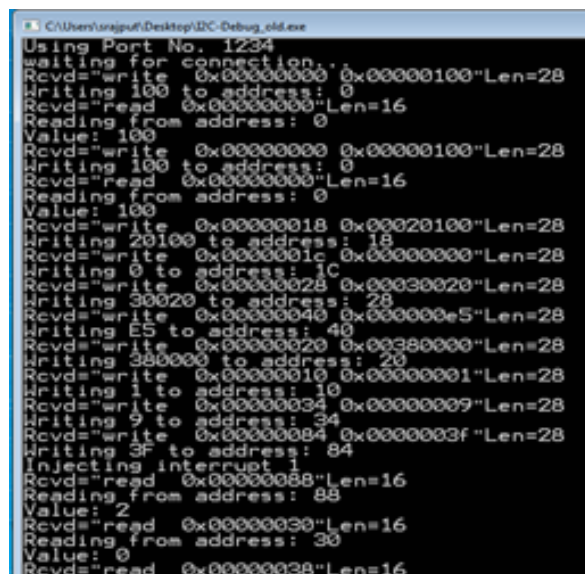

Figure 8: Server socket connected to client socket

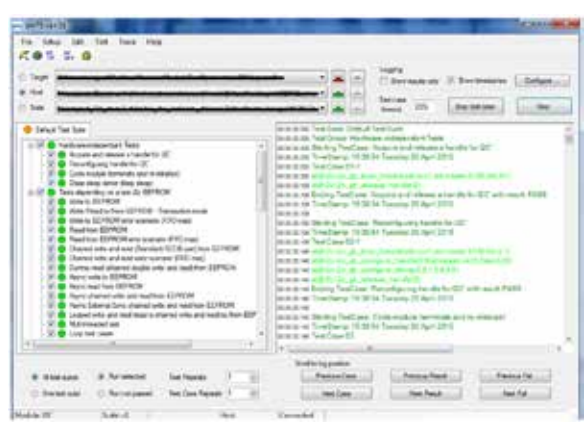

Figure 9: Test cases in GMTS

CONCLUSION AND FUTURE SCOPE

A communication between SystemC model of I2C and the test cases using socket programming is implemented in this paper. This method gives an efficient way to test the I2C driver even before the fully functional virtual prototype and hardware is ready. Currently test cases are executed only for

I2C master in future SystemC model I2C slave can be implemented and test cases for I2C slave can be executed on it. This method can be used by other device drivers for testing purpose.

REFERENCES

[1]Liviulftode, Cristian Borcea, Nishkam Ravi, Porlin Kang, and Peng Zhou, "Smart Phone: An Embedded System for Universal Interactions" IEEE conference publications May 2004 | [2]Peter Corcoran, "Two Wires and 30 Years - A tribute and introductory tutorial to the I²C two-wire bus" IEEE Consumer Electronics magazine july 2013. | [3]NXP (2000, Jan.) I2C bus specifications, version 2.1.[online].available:http://www.nxp.com/documents/user_manual/Um10204.pdf. | [4]K.L. Eddie Law, Roy Leung, "A Design and Implementation of Active Network, Socket Programming" IEEE conference publications 2002. | [5] Zuquete A., Lisbon Portugal, Guedes P., "Transparent authentication and confidentiality for stream sockets" Micro, IEEE (Volume:16 , Issue: 3) August 2002.