DESIGN AND VERIFICATION OF LOW SPEED PERIPHERAL SUBSYSTEM SUPPORTING PROTOCOLS LIKE SPI, I2C AND UART

Shanthipriya S1 and Lakshmi S2
1Masters of Technology Very Large Scale Integrated, Sathyabama University, Chennai, Tamil Nadu, India
2Faculty of Electricals and Electronics, Sathyabama University, Chennai, Tamil Nadu, India
Email: shanmugam.shanthipriya@gmail.com

ABSTRACT
In this paper we will discuss about Low speed peripheral architecture, which involves multiple I2C, SPI, UART instances, DMA engine. Given the limited number of general purpose IO ports are SOC boundary, these protocol blocks are grouped into a single subsystem while sharing all the resources like DMA, channels and GPIO’s while interfacing with external chips. The design has been done using verilog we will be analyzing the performance of the subsystem by using System verilog & Universal verification method based testbench environment to generate various use case scenarios involving different number of I2C, SPI and UART instances. And this project is simulated using Questasim 10.2c.

Keywords: I2c, SPI, UART, DMA, verilog, system verilog, UVM, GPIO.

1. INTRODUCTION
In recent days microcontroller embedded system are often use many modules in the designs in large area of products from industries, medical, and educational domains.

The communication between modules are very important, each and every system includes some microcontroller core intelligence control etc. For example RAM EEPROM (or) data converters. Now a day’s interface is the heart of the system performance, on the cases of Intellectual property macro cell is the key source of design mode, and can able to produce chip. In order to attain an error free communication, each and every soc must linked properly in an efficient manner. Generally SOC uses peripheral subsystem in order to connect with external chips, the protocols which used to form subsystem are categorized into two types one is high speed communication protocol and other is low speed communication protocols. Here we are using low speed communication protocols, because low speed communication protocols leads to low power consumption. The SPI, I2C and UART are low speed communication protocols. They all together forms Low speed peripheral sub system in order to connect with external chips. In this work, design and verification of low speed peripheral subsystem is presented.

2. BUS PRINCIPLES
A. SPI bus principle
SPI is a synchronous data bus, which has perfect synchronization of data and clock line. Though clock is an oscillating signal it knows commands the receiver that when to sample the data line. This could be sample when clock signal goes from low to high (or) vice versa, which depends on data sheet. The next bit of the data will be detected when the receiver detects the edge of the signal. The device will generally operate at top speed, where it can work almost. The four main signals of SPI are SCK, MOSI, MISO, SS. [1] The SPI will always have only one master but have many slaves. The data which comes out from the master are called Master Out/ Slave In, when slave needs to response back to the master it uses MISO (i.e.) Master In /Slave Out. SPI have separate transmit and receive line, though it is full duplex we can transmit and receive data simultaneously. [1] The other important signal in SPI is SS (Slave select). Which tells when to Transmit (or) Receive data? Usually the SS line will held high, which disconnects the slave device from SPI bus. The line is brought low when the data have to transmit, which will activates the slave. [1] [3]

Figure-1. Spi module.

SPI simulation output is as follows , it works according to the above spi protocol behaviour
B. I²C bus principle

I²C is an inter-integrated protocol which is often used for short communication within a single device. They can communicate multiple slave and one (or) more master chips. It requires only two signal wires to transmit and receive information. The two important signals of I²C are SCL and SDA. Where SCL is called a clock signal and SDA is called a data signal. In order to send more data from the master, sometimes, the slave devices free the clock low. I²C are “open drain” where it can make the signal line low but cannot drive high. The communication of I²C is divided into two frames: one is address frame and other is data frame. The I²C protocol has 8 bit of data messages which transfer from Master to slave (or) vice versa [9], [10]. The communication of protocol is as follows:

Start condition: when I²C starts to transfer data it pull SCL line high and pull SDA line low. This makes the slave device alert and notice that a transmission is above to start. When two master devices take a lead at the same time, whichever pull SDA low first will control the bus.

Address frame: In I²C communication, the new communication sequences starts by a 7 bit of address frame (or) clocking the MSB bit after the read/write happens. The 9th bit is NACK/ACK, when SDA line does not go low before 9th clock pulse, then the receiving device will not receive the message. After the 8th bit of data or address are passed the receiving device will given control over SDA.

Data frame: The data is placed on the SDA line; initially address is passed after that data will be transmitted. The master will generate clock pulse at a regular interval and data will be placed on SDA by master or slave, which depends on R/W operation.

Stop condition: The master generates stop condition when all data frames have been sent. The stop condition is define by SDA line goes low to high after a SCL goes from low to high. I²C simulation output is as follows, it works according to the above I²C protocol behaviour [7], [6].
C. UART principle

UART is a universal asynchronous protocol, which is used to transmit and receive the data. The communication of UART is similarly as like that of SPI, they communicate directly with each other. Usually it converts parallel data to serial (or) vice versa. UART requires only two pins, transmitting TX pin and receiving RX pin. As UART is asynchronous protocol which means that there is no clock signal to synchronize the bit from transmitting UART to receiving UART. Instead of clock it has start and stop bit so that receiving UART knows when to start and stop the data. [2], [1]. The UART communication is as follows:

![UART module](image)

**Figure-5. UART module.**

**Start bit:** To start the data transfer initially the UART pull the transmission line from high to low. When the receiver UART detects this transition it start to read the data.

**Data frame:** There can be 5 to 8 bit of data if parity bit is used, if not it can be upto 9 bit. In most cases the data will be sent in least significant bit.

**Parity bit:** The parity bit will tell whether it is a odd or even parity. The transmission in UART is based on baud rate. Baud rate measures the speed of data transfer which express in bits per second (bps). The receiving UART checks for odd or even parity when it read the data frame. When it is “0” it is even parity. When it is “1” it is odd parity.

The transmission is said to error free transmission when the parity bit matches the data.

Stop condition: when the receiving UART detect from LOW to high transition the data packet ends. [2] [5]

![UART communication pattern](image)

**Figure-6. UART communication pattern.**

UART simulation output is as follows, it works according to the above UART protocol behaviour.
3. Low speed peripheral subsystem

A. LSPSS design principles

Low speed peripheral subsystem interfaces with processor on one side, other chips on other sides. Processor writes data in to memory, also programs descriptors in to DMA engine. DMA engine executes these descriptors one by one. While executing these, it fetches data from memory and does the data transfer on I2C, SPI and UART as per protocol behavior. Same concept is used while collecting data on receive interface of I2C/SPI/UART. The design has been done using verilog language. And the verification is done using system verilog and UVM methodology. The various test cases have been done using this methodology.

B. System verilog and UVM methodology

System verilog is mainly used for verification process like checking, coverage collection and stimulus generation, that cause some aspects of verifying a digital design and then from this methodologies are started [4]. In that UVM is one of the methodology used to automate verification. In order to create an efficient verification environment in the platform of system verilog, UVM came into the picture with collection of ADI and PR oven verification. This helps the verification engineer to perform efficiently. This also helps the design to reuse the test from previous projects and can able to modify the components of their needs. This can be improved by without modifying the original code. [3]

4. SIMULATION RESULTS

The design of low speed peripheral subsystem is designed using verilog and verification of LSPSS is done using system verilog and UVM methodology. The various
testcases have been done using this methodology. The testcases of low speed peripheral subsystem is as follows:

a) SPI & UART:

Figure-9. SPI & UART design output.

b) SPI & I2C:

Figure-10. SPI & I2C design output.

c) UART & I2C:

Figure-11. UART & I2C design output.

d) SPI, UART & I2C

Figure-12. SPI I2C & UART design output.

5. CONCLUSION & FUTURE SCOPE

In this paper a low speed peripheral subsystem is designed by using verilog and verified using system verilog & UVM methodology. By this design and verification method we can able to transfer maximum amount data in a given cycles and the main advantage of this work is the entire hardware is optimized so that low cost is achieved and port reduction is also done. The various combinations of testcases have done in order to check whether it works in all combinations. In future it can be further extend by adding more number of SPI I2C and UART protocol.

REFERENCES


[9] Sansar chand sankhya. 2014. Design and implementation of I2C mastercontroller interfaced with RAM using VHDL IJERA.