Design and analysis of microcontroller system using AMBA-Lite bus

Wang Hang Suan[1,], and Asral Bahari Jambek[1]*

[1]School of Microelectronic Engineering, Universiti Malaysia Perlis, Perlis, Malaysia

Abstract. Advanced Microcontroller Bus Architecture (AMBA) is one of the well-designed on chip communication system. It is designed for right first-time development with many processor and peripherals. In this paper, the different family of AMBA architecture such as AXI, APB, AHB are reviewed. In this work, the AMBA-Lite is used and implemented with a few peripherals and an ARM processor. The work is simulated using Synopsys and demonstrated on the Digilent Nexys4 DDR board and the software use to synthesis the design is Vivado 2016.2.

1 Introduction

The demand of microcontroller is increasing rapidly in this few decades. It can be found in many daily application such as washing machine, refrigerator, and automobile. The technology for miniaturising the system is now a major concern by the designer. The silicon technology has been developed from a few transistor to hundreds of million transistors since the invention of microprocessor. Today, the technology allows the integration of different part of component into a single chip which also known as System on Chip (SoC) [1].

One of the main advantage of SoC is that the intellectual property (IP) can be reuse in other design. The Advanced Microcontroller Bus Architecture (AMBA) developed by ARM Ltd. is also designed for this purpose. Therefore, AMBA architecture is most widely used in communication interface [2]. The AMBA-Lite is a simplified version of the AMBA bus family, but it can only support a single master. This cause some of the signals for multi-master system is removed like HGRANT, HBUSREQ.

Besides, the AMBA-Lite also support high-bandwidth operation. This include burst transfer, single clock edge operation, non-tristate implementation and so on [3]. The bus consists of one address decoder and one multiplexor to select the correct slave. The master starts giving the address and control signals. The write data then moves data from master to slave or vice versa. All slaves will be able to extend the data phase by making a request to master and send a response to the master.

In this work, an ARM soft processor and the AMBA-Lite are implemented on the Digilent Nexys4 DDR board. Section II discusses other research work on the different AMBA architecture. Next, section III show the methodology to implement a simple microcontroller using IP developed by ARM. In section IV, the result is shown on the board and on the monitor and the result is discussed and analysed.

2 Literature review

In paper [4], an extensible interface bus (AXI) was used in the design Input Output (IO) system. The AXI is similar to other Advanced Microcontroller Bus Architecture (AMBA) which are a high performance on chip communication buses. The slaves and master interface were developed with AXI on chip communication standard interface. The master was an Arithmetic Logic Unit (ALU) which can perform 22 operations, while the slave peripherals were SRAM, ROM, and FIFO. The AXI act as a medium between the master and the slaves, where the master will start to initiate the address. The round robin arbiter was used to select the slave where this type of arbiter depends upon the request from master. Fig. 1 shows the block diagram of the AMBA IO system.

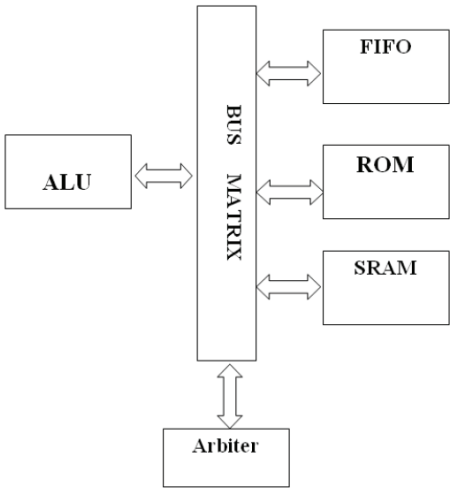

Fig. 1. AMBA IO block diagram. [4]

*wanghs92@gmail.com

In paper [5], a system had been designed with a processor that will send the address to the synthesized memory and read the data. First, a .bin file had been generated using Keil software. Then the file was implemented into a synthesized VHDL module. A Divider Clock Manager (DCM) was used to generate 10 MHz. The Core Generator tool was set to 4KB RAM and using the 32bit of data access which is shown in Fig 2. The result is shown by turning on or turning off the Light Emitted Diode (LED), and the maximum clock speed was 40MHz.

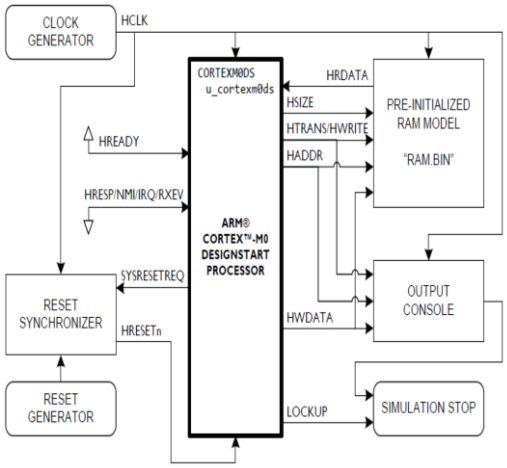

Fig. 2. Block diagram of the system. [5]

In paper [6], a design with an arbiter was used in the system. Fig 3 shows the AMBA AHB interconnect matrix for multichannel technique. The arbiter will decide which master can access the bus, then the master only can initiate read and write. This system used a combination of round robin and dynamic arbiter. The master will be chosen if any of the master request and will reduce one priority. The system is designed with a few master, an arbiter, slave and decoder. The result show that the output data was successfully obtained after the simulation.

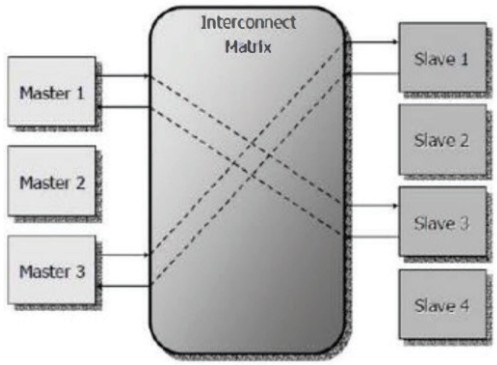

Fig. 3. Block diagram of interconnect matrix.[6]

In paper [7], an Advanced Peripheral Bridge (APB) bridge was used. The system was designed to minimise the clock skew by adding a 3 bits ripple down counter. The flip flop was clocked sequentially, so that the data output of source flip flop can only toggle the clock of sink flip flop. The bridge act as a single master for the connected peripherals which perform read and write transfers. The bridge consists of a state machine, a reset controller, and the ripple down counter sub-module. The undesired clock skew can be avoided by using this approach. The result shown that by using the ripple down counter, it can also reduce the power on chip and total clock domain.

In paper [8], the Advanced System Bus (ASB) was implemented. The operation start from the master contact with the bus. Then, the arbiter will determine its status before the master start transferring the data. The decoder will then select the slave and a response will be sent back to the master. The bridge acts as a slave to the master so it continues putting wait state on the master only when they are needed. The bridge consisted of a state machine and decoding logic to select signals. The state machine of the system is shown in Fig 4 which the decoding cycles start before the slave response. The system bus used was 50MHz with 32 bits of address and data buses. Besides, the system was assumed to connect with 4 peripherals.

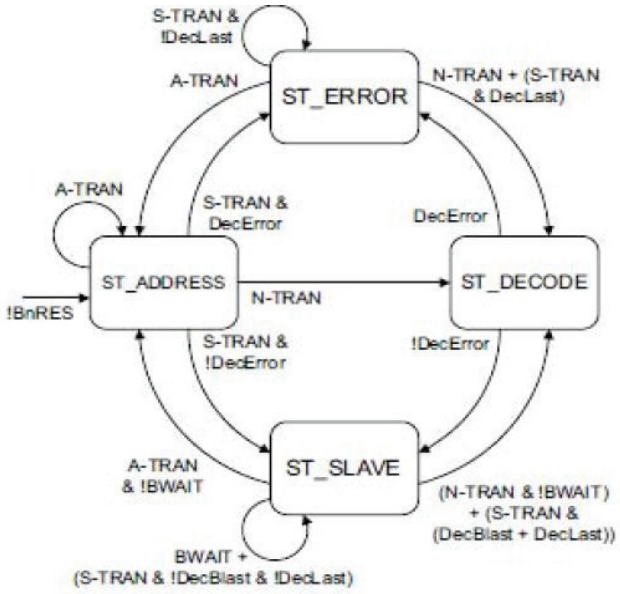

Fig. 4. State machine of the APB bridge.[8]

The summary of different bus architecture made by others is shown in table 1. The purpose of paper [7] and [8] is more to reduce power consumption while others are focusing on the functional of the system. The usage of arbiter depends on the application; it is used when more than one master is implemented. Besides, the bridge needs to avoid any clock skew or timing violation when designing the system.

Table 1. Summary of the methods by others paper.

	[4]	[5]	[6]	[7]	[8]
Master	ALU	Cortex M0 DSK	-	-	-
Bus Architecture	AXI	AHB-Lite	AHB	APB	ASB
Slave	SRAM ROM FIFO	LED Memory	-	APB bridge	APB bridge
Arbiter	Round Robin arbiter	-	Round robin and dynamic	-	-
Total power	-	-	-	0.57mW	0.66mW

3 Methodology

In this section, the design of the top module and the individual module and how it work will be explained. The program of the microcontroller is stored inside the memory which is written in Verilog code. The size of the memory can be defined inside the code. In this system, 4KB of memory is sufficient to store the program. Since it is an internal memory, the program and the architecture is implemented together into the board. In this research, an ARM Cortex-M0 Design Start Processor, an Advanced High-performance bus (AHB) lite, an internal memory, an Universal Asynchronous Receiver Transmitter (UART) peripheral, a Video Graphics Array (VGA) peripheral, a timer peripheral, a General Purpose Input Output (GPIO) peripheral, a Light Emitting Diode (LED) peripheral, and a 7-segment peripheral are used. All these module are written in the form of Verilog code and can be reusable in other design. Fig 5 shows the system block diagram with each has the interface to interact with the AMBA-Lite. The most important aspect to integration is the address mapping. The address need to be correct so that the bus is able decode it to the right IP. Besides, inside the application program, the programmer need to know where is the IP address in order call the respective IP.

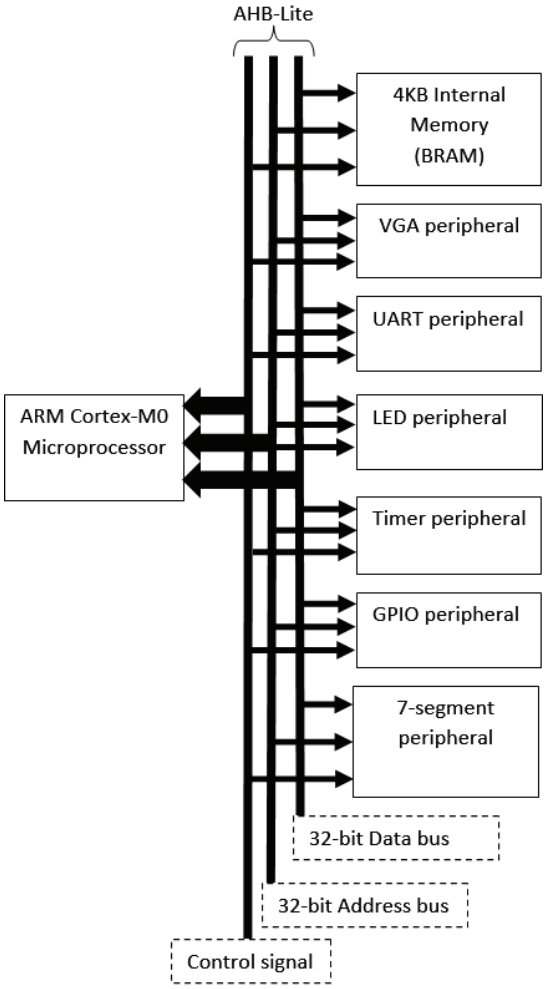

Fig. 5. The peripheral used in the system.

First, the program which the instruction to be executed by the processor is compiled from assembly code into machine code using Keil µVision before targeting into the board as mention above. The LED peripheral is a simple block to interact with the 8 LED on the board for other application purposes. In this paper, the LED is connected to GPIO, when the GPIO receive the signal from switch, it will turn on the LED. The GPIO has a data direction register to indicate either it is input or output signal.

On the other hand, the VGA is a more complex module which consists of a few sub-module inside. One of the sub-module is VGA interface, it is used to generate synchronization signals to the VGA port and transfer the colour to the pin on the board. Besides, it has image buffer and text console, which are used to store the colour information in the image region and displaying text respectively. The text console is implemented on the hardware logic so that it can save the valuable on-chip memory. The VGA connector is coming along the function to convert the digital output to analogue using resistor divider. The hierarchy of the VGA module is as depicted in Fig 6.

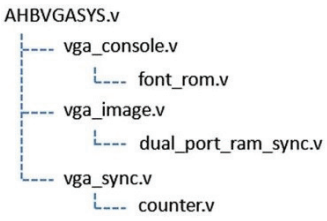

Fig. 6. Hierarchy of the VGA module. [9]

Next, the timer module is a software timer to act like a clock to count the event and the result is displayed on the 7-segment display and it is programmed to reset when it reaches zero. It is based on the hardware timer but when it compared to a hardware timer, it has a lower level of precision. In this module, it has a 32-bit counter, which start from 'FFFF'.

The module has a programmable register to control the frequency of the counter and the load value. The timer prescale value can be set as 1 bit, 16 bits or 256 bits. Besides, there are two modes available for interrupt to occur, which are free running mode and periodic mode. These types of interrupt enable different program can be run on the same time and without modifying the timer architecture.

In this paper, the application will not use any interrupt, therefore, the four digit of 7-segment display will be used to show the result. In order to use the 7-segment display, a decoder is needed. It is designef based on the four registers to display the 32-bit counter with a 1 kHz looping frequency. The peripheral is set for common anode display since the 7-segment display of Digilent Nexys4 DDR board is an anode type.

Besides, this architecture also integrate the UART module which can receive or send signal through serial port. The UART also consists of some sub-module, such as module to generate a fixed transmission baud rate, a First In First Out (FIFO) register to buffer the data to be sent or received. The UART module is designed with a start bit and a stop bit to indicate finishing transfer.

In this paper, the baud rate is set to be 19200 bps, and an open source software called Tera Term is used to communicate between the laptop and the board. The Table 2 shows the memory map of each peripheral in this design. The peripherals module used are all available on the ARM university program website [9].

Table 2. Memory map design.

Module	Base address	End address
BRAM	0x0000_0000	0x4FFF_FFFF
VGA	0x5000_0000	0x50FF_FFFF
UART	0x5100_0000	0x51FF_FFFF
Timer	0x5200_0000	0x52FF_FFFF
GPIO	0x5300_0000	0x53FF_FFFF
7-seg	0x5400_0000	0x54FF_FFFF
LED	0x5500_0000	0x55FF_FFFF

4 Result and discussion

In this paper, the simulation was carried out using Synopsys. The result shown is similar as intended and carried out in different part based on IP. Based on Fig 7, an input act as a switch is added to test the GPIO IP. The LED then has the same output based on the input (sw). Fig 8 shows the UART IP sending out a signal from the system. The word "TEST" is stored in the memory and the application program is transmitted out from RsTx. It can be interpreted using the ASICII character. For example, from the ASICII, the "T" is 1010100 as shown in figure, with a zero start bit and a one stop bit.

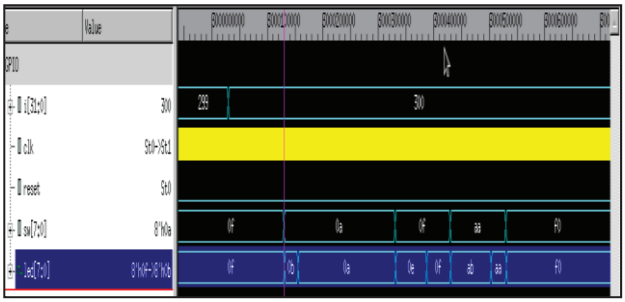

Fig. 7. The simulation on GPIO and LED IP.

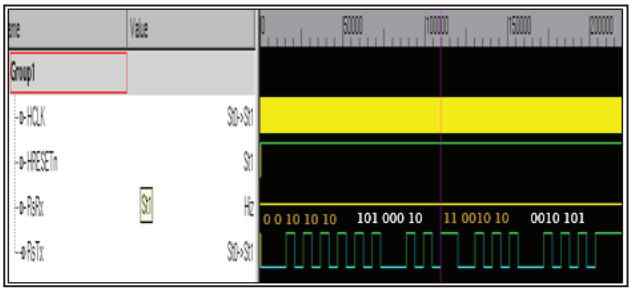

Fig. 8. The content of memory is send out on UART IP.

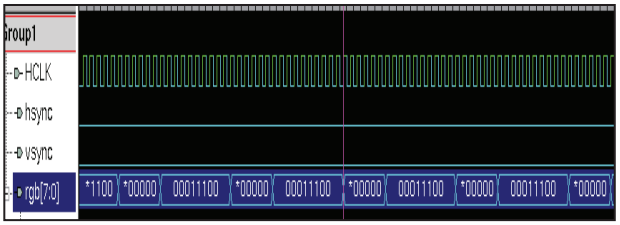

Fig. 9. The content of memory is send out on RGB IP.

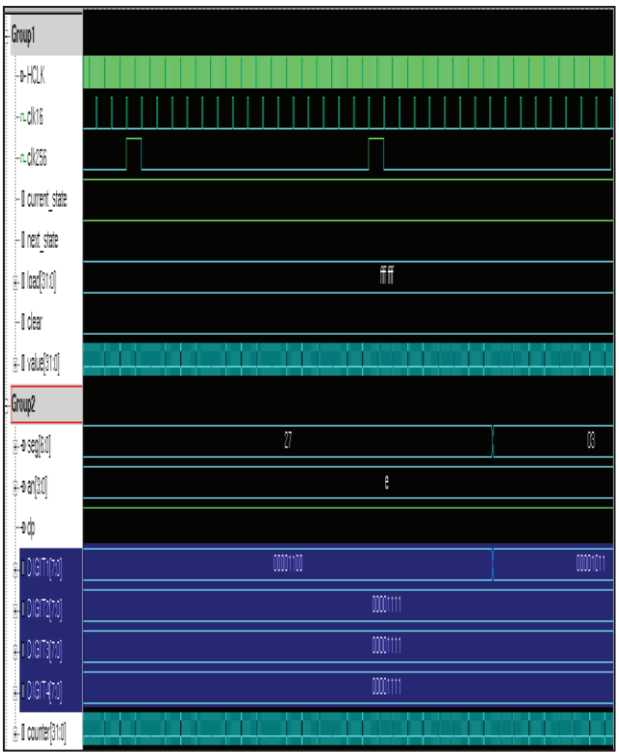

Fig. 10. The timer IP is count down and shown on 7segdec IP.

Fig 9 shows the content stored in memory is displayed on RGB and the colour is set as green which is 00011100. The word “HELLO” is displayed which is also demonstrated in FPGA.

Fig 10 shows the timer is counting down and the 7 segment decoder decoding the time. Four digits are displayed and for the first digit is 00001100 and the next state is 00001011. The figure for the timer is used to debug and to check for the state is running from one state to another. The timer has a prescalar that can be set the clock for an extra 16 count or 256 count which can be seen as clk16 and clk256.

Meanwhile, the Digilent Nexys4 DDR FPGA board is used to run the ARM IP architecture using Vivado 2016.2. The UART communicates with the laptop using Tera Term with a baud rate 19200bps. Fig 7 shows the word ‘HELLO’ is stored in the memory is printed out on the screen. In this figure, the result shown has two IP used. The word ‘HELLO’ which previously stored in the hex file is shown on the first line of another monitor using VGA IP. Another IP used is the UART IP and the result is transferred from the board to the laptop screen via UART terminal.

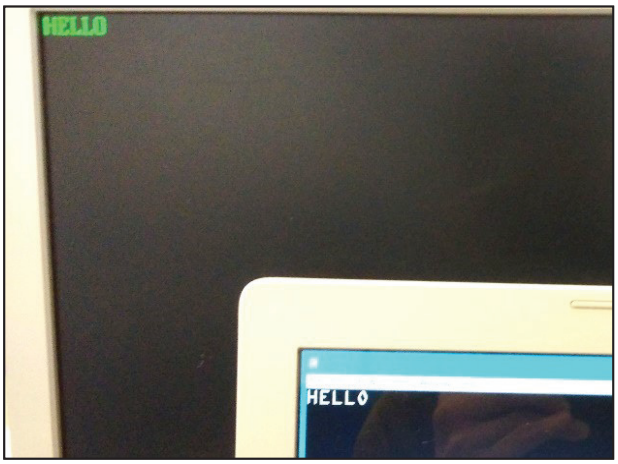

Fig. 11. The content of memory is printed out on screen.

In Fig 8, the result shows that the timer is decreasing in the 7-segment display from FFFF to 0000, with a 16-bit prescaler. The prescaler can be set using the program to determine the speed of the count down. There is no interrupt function in this case, but the program can be added to this IP that is developed by ARM [10]. Besides, the figure also show that the LED is turned on according to the respective switch below by using GPIO module.

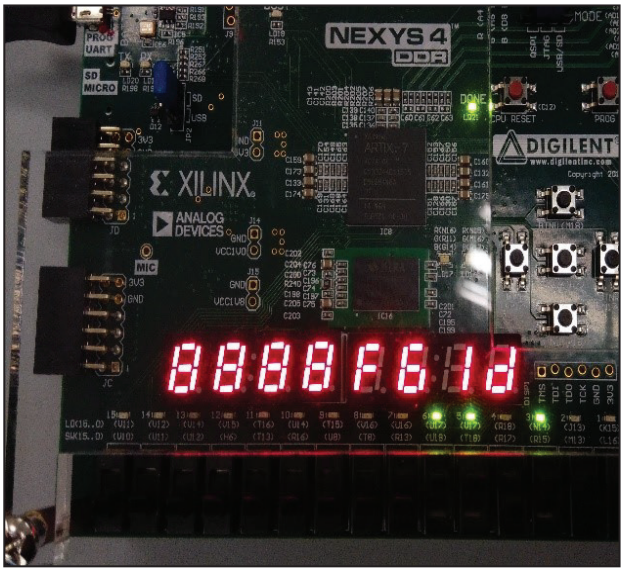

Fig. 12. The timer is counting down.

4 Conclusion

In this paper, an AMBA-Lite with a few peripherals was implemented on to the FPGA board and the result showed was similar as expected. The result is verified using Synopsys and implemented on the FPGA Digilent Nexys4 DDR board. The ARM Cortex-M0 Design Start was used as the processor to run the program inside the memory where the application was compiled using ARM tool chain.

This research was supported by TalentCorp Malaysia.

References

1. K. S. Kumar, P. Deepthi, IJERA **3**, 1005-1010 (2013)
2. A. Laurentiu, B. Angel, WCE **I,** 459-464 (2015)
3. "AMBA 3 AHB-Lite Protocol," **1.0 ed**: ARM Ltd. (2006)
4. K Swetha, G. Ramakrishna, IJEERT **2**, 51-59 (2014)
5. M. J. Oviedo, P. A. Ferreyra, VII DF 87-93 (2011)
6. S. Divekar, A. Tiwari, ICCSP, 1854-1858 (2014)
7. K. Rawat, K. Sahni, S. Pandey, J. Rawat, and S. Tripathi, IC4, 1-4 (2015)
8. K. Rawat, K. Sahni, and S. Pandey, SPIN, 927-930 (2015)
9. "DesignStart - ARM," **1.0 ed**: ARM Ltd.
10. "Cortex-M0 Technical Reference Manual Copyright," ed: ARM Ltd., (2009)