Synthesizing a Requester Device Using VHDL:
A Basic Review supporting Ethernet communication platform

Rinju Mariam Rolly[1], Teena Rajan[2],
Deartment of Electronics & Communication Engineering[1, 2,]
P G Scholar, Mar Baselios College of Engineering & Technology[1,]
Assistant Professor, Mar Baselios College of Engineering & Technology[2]
Email: rinju_mariam@yahoo.com[1] ,

Abstract- A web-based user client can serve as an interface to control the applications executing on a platform, but this would require a web interface on the multiprocessor platform under investigation. The main objective of this project is to provide network connectivity to the electronic devices. By doing so, automatic updating can be achieved. This helps to keep control of the performance of device. A requester device is simulated using VHDL.

Index Terms- Ethernet; FIFO; GPIO ports; Requester device

1. INTRODUCTION

As technology is varying or updating day by day, in this modern internet world, we are interested in real-time performance evaluation of multiple applications executing concurrently on a multiprocessor platform. Ethernet communication platform [1-2] provides a way to control programmable logic devices, as well as data communication using a standard network. The implementation of the communication platform depends on the requester device, because the GPIO ports must be configured to allow the connection of diverse such devices (and architectures). The research work in [1] deals with a communication platform which helps in automation of the electronic devices. Here, the work is about the implementation of requester device that is a part of above specified communication platform. For the complete operation as a communication platform, the network layers should be defined. Reference [3] describes about the communication protocols for a bidirectional data transmission. The requester device simulated is done by VHDL programming [4]. [5-6] explains about multi tasking platforms operating on multi-processor systems.

2. SYSTEM DESIGN

The device that is synthesized in the FPGA is called requester device. The requester device is represented as a forwarder device. The device should transmit data to far away centre, where the received data from the device is analyzed and processed for further applications and system development. The device should not only transmit, but also it should receive signals or commands or data from an external transmission centre. Any device that takes part in communication needs to be a bi-directional in nature. A communication is success only if it conveys its message error free and they both (sender and receiver) exchanged their information and communicated each other.

As shown in figure 1, the requester device consists of three parts: GPIO to FIFO block, the forwarder device and FIFO to GPIO section. GPIO to FIFO is used to receive data transmitted by a remote computer or by any other transmitter. The data is received through the Ethernet physical layer and for perfect transaction that received data block should be read from the GPIO pins to the Data_In of requester device.

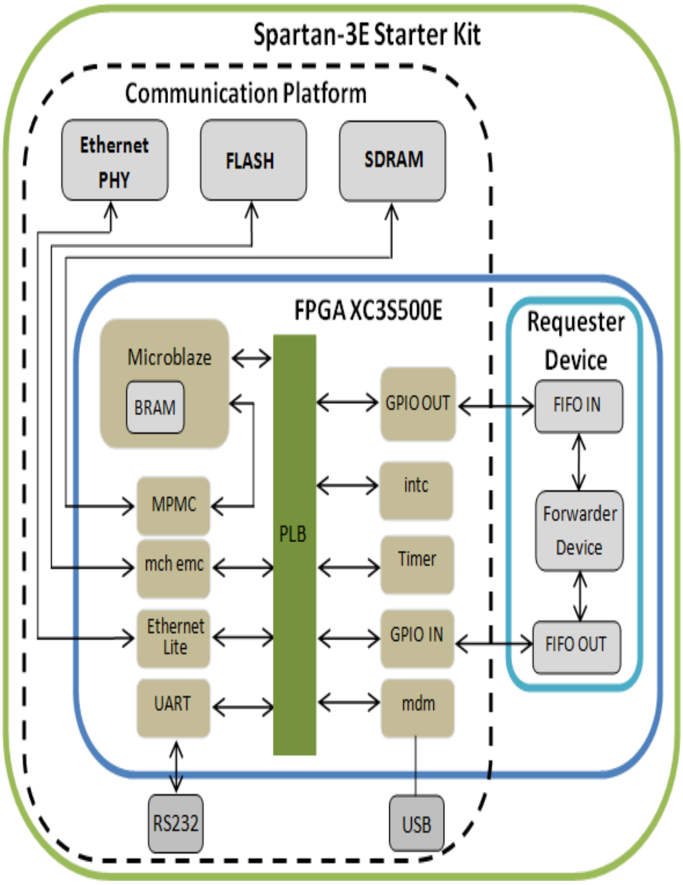


Figure 1: Embedded system with requester device [1]

WE: Write Enable
RE: Read Enable

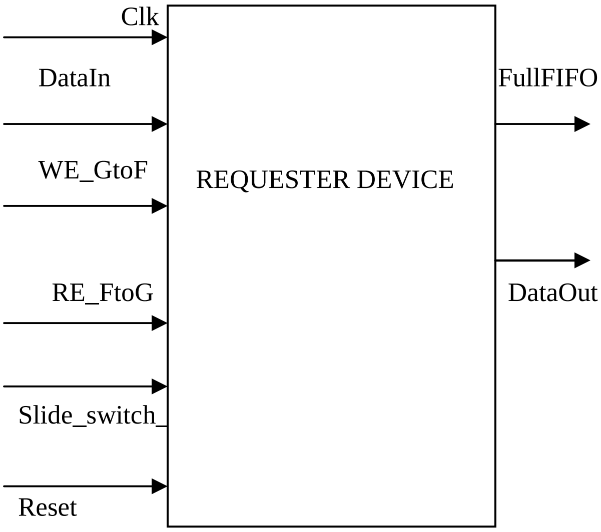


Figure 2: Requester device

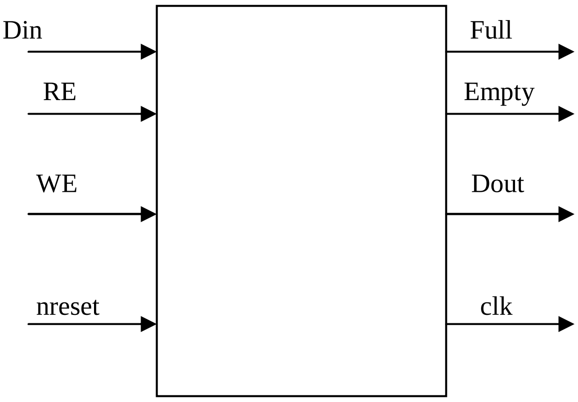


Figure 3: Fifo1 Port Diagram

Figure 3 shows the pin diagram for fifo1, i.e., GPIO_to_FIFO data transfer block. The data out from GPIO pins is read and writes to Din.

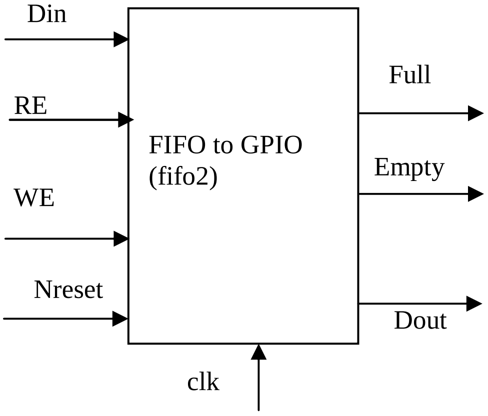


Figure 4: fifo2 Port Diagram

Din: 8-bit Data Input
Dout: 8-bit data output
nreset: reset signal
Full: Shows ‘1’, when the FIFO is full
Empty: Shows ‘1’, when the FIFO is empty

Figure 4 shows the reverse action for figure3. It shows the data from FIFO is given to the GPIO(General Purpose Input Output) pins. Dout, Full and Empty are output signals, whereas the other five signals shown in figure 3 are inputs to the Requester FIFO IN. It describes the port diagram of FIFO_to_GPIO block. Here, the data is given back to the GPIO IN pin from the FIFO OUT. The signals have the same operational nature as for the signals in figure 3. The FIFO’s are enabled by the clocking pulse, which is of picoseconds range. The FPGA has a 50 MHz clock oscillator in built in it.

3. RESULT & DISCUSSIONS

The proposed device is simulated with the help of ModelSim software. The chapter describes the simulation in modelsim simulation and debugging software.

Select New ⟶ Project ⟶ Define the project name(rin) ⟶ Add new file

Add files to existing project ⟶ Save the files with .vhd extension.

All the files saved will get attach to the project. The files are named GPIOtoFIFO, FIFOtoGPIO and Requester_1.The first step for the simulation is:

- Select the VHDL files and compile

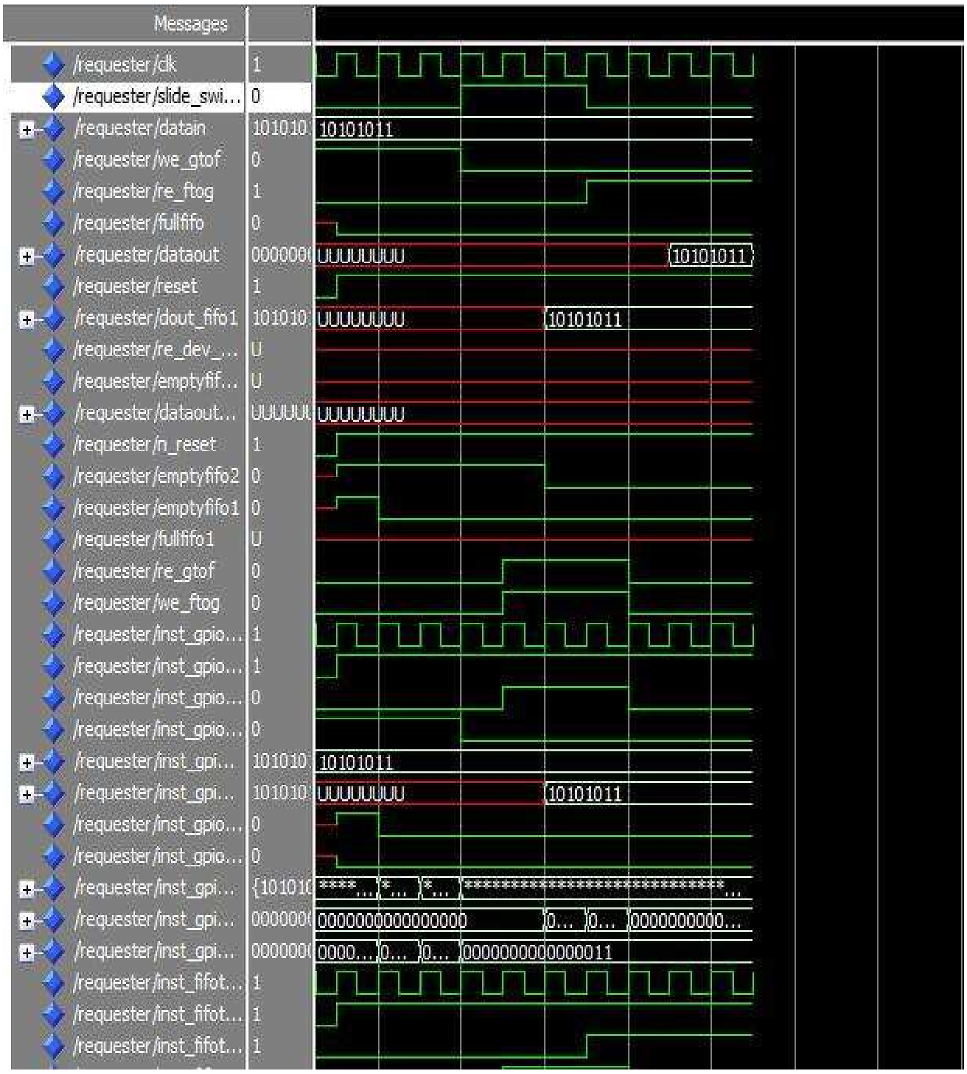
Select compile selected from the compile option. This will cause the compilation of the selected files from the project created. If the files are correct (error free), then the compilation will comes out successfully. The files GPIOtoFIFO and FIFOtoGPIO are defined inside the file Requester_1. Hence the Requester_1 file will compile first, during its compilation, the other two files are called within it. This completes the compilation of the project. Next, we should go for simulation. Compilation is always followed by simulation.

- Simulation

Click on start simulation. A new pop-up window opens as shown in below. From design option, select work library. Click on (+) sign. Fifo1, fifo2 and requester are entities used. Entities are saved in the work directory. Only the requester entity is selected and simulated. Since the other two defined entities are a part of this aforementioned entity, the fifo1 and fifo2 will be called inside the requester and simulates them. So for that, click on requester and select OK. This operation completes the simulation step of project. All the aforementioned steps are done to configure the GPIO pins or otherwise, it is to define the requester device.

- Output

The values are given to the signals manually in ModelSim simulation software, whereas in ISE (Integrated simulation environment) the signals are called automatically. So for that, the values each variable should posses are programmed in ISE bench.



Messages		
/requester/clk	1	
/requester/slide_swi...	0	
/requester/datain	1010101	10101011
/requester/we_gtof	0	
/requester/re_ftog	1	
/requester/fullfifo	0	
/requester/dataout	0000000	UUUUUUUU 10101011
/requester/reset	1	
/requester/dout_fifo1	1010101	UUUUUUUU 10101011
/requester/re_dev_...	U	
/requester/emptyfif...	U	
/requester/dataout...	UUUUUU	UUUUUUUU
/requester/n_reset	1	
/requester/emptyfifo2	0	
/requester/emptyfifo1	0	
/requester/fullfifo1	U	
/requester/re_gtof	0	
/requester/we_ftog	0	
/requester/inst_gpio...	1	
/requester/inst_gpio...	1	
/requester/inst_gpio...	0	
/requester/inst_gpio...	0	
/requester/inst_gpi...	1010101	10101011
/requester/inst_gpi...	1010101	UUUUUUUU 10101011
/requester/inst_gpio...	0	
/requester/inst_gpio...	0	
/requester/inst_gpi...	{101010	****... *... *... *****************************...
/requester/inst_gpi...	0000000	0000000000000000 0... 0... 0000000000...
/requester/inst_gpi...	0000000	0000... 0... 0... 0000000000000011
/requester/inst_fifot...	1	
/requester/inst_fifot...	1	
/requester/inst_fifot...	1	

Figure 5: Simulation result

4. CONCLUSION

A simple requester device is simulated with the help of ModelSim simulation and debugging software. The result shows the data given to the GPIO pin is read by FIFO and it is written back to GPIO out pin. The simulation illustrates a forwarding device and it finds application as a mediator between two devices for data transfer and is base for Ethernet communication platform.

Acknowlegements

I thank my Lord almighty for the strength he blessed me with. I would like to extend my gratitude to my guide, faculties, friends, to my family and to all who supported me in this work.

REFERENCES

[1] Rodrigo Neri de Souza, Andr´e Vaz da Silva Fidalgo, Daiana Nascimento Muniz, "*Ethernet Communication Platform for synthesized devices in Xilinx FPGA*", IEEE Conference, September 2011

[2] Rodrigo Neri de Souza, Plataforma de Comunicac¸ ˜ao Ethernet para dispositivos embarcados em FPGAs da Xilinx, Undergraduate thesis, IFSC, Brazil, 2010.

[3] Douglas L. Perry, VHDL programming by example, 4ed. Tata McGraw Hill 2002

[4] Tanenbaum, A.S., Computer Networks, 4 ed.,

[5] M.J. Rooijakkers, "Web server controlled multi-tasking on a FPGA based multiprocessor platform",

[6] Akash Kumar, Shakith Fernando, Yajun Ha, Bart Mesman, and Henk Corporaal, "Multi-Processor System-Level Synthesis for Multiple Applications on Platform FPGA", *IEEE Conference Publications, August 2007*