International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 3, No. 3, November 2014, pp. 85~97
ISSN: 2089-4864

Design and Implementation of an Ethernet MAC IP Core for Embedded Applications

Kautilya Singh Parmar*, Sanket Dessai*, S.G. Shiva Prasad Yadav, Atul Chauhan*****
**Department of Computer Science and Engineering, Faculty of Engineering and Technology, M.S. Ramaiah School of Advanced Studies, Bangalore India
** Department of Telecommunication Engineering, M.S. Ramaiah Institute of Technology, Bangalore, India
***Hardware Engineer, NVIDIA Graphics Co.Bangalore, India

Article Info

Article history:

Received May 11, 2014
Revised Sep 20, 2014
Accepted Oct 12, 2014

Keyword:

ASIC
CRC-32
Ethernet MAC
FPGA
LAN

ABSTRACT

An IP (intellectual property) core is a block of logic or data that is used in making a field programmable gate array (FPGA) or application-specific integrated circuit (ASIC) for a product. As essential elements of design reuse, IP cores are part of the growing electronic design automation (EDA) industry trend towards repeated use of previously designed components. Ethernet continues to be one of the most popular LAN technologies. Due to the robustness resulting from its wide acceptance and deployment, there has been an attempt to build Ethernet-based real-time control networks for manufacturing automation. There is a growing demand for low cost, power efficient MAC IP Core for various embedded applications. In this paper a project is discussed to design an Ethernet MAC IP Core solution for such embedded applications. The proposed 10_100_1000 Mbps tri-mode Ethernet MAC implements a MAC controller conforming to IEEE 802.3 specification. It is designed to use less than 2000 LCs/LEs to implement full function. It will use inferred RAMs and PADs to reduce technology dependence. To increase the flexibility, three optional modules can be added to or removed from the project. A GUI configuration interface, created by Tcl/tk script language, is convenient for configuring optional modules, FIFO depth and verification parameters. Furthermore, a verification system was designed with Tcl/tk user interface, by which the stimulus can be generated automatically and the output packets can be verified with CRC-32 checksum. A solution which would consume a smaller part of the targeted FPGA, and thus giving room for other on-chip peripherals or enable the use of a smaller sized FPGA. To employ a smaller FPGA is desirable since it would reduce power consumption and device price.

Corresponding Author:

Sanket Dessai,
Departement of Computer Engineering, M.S. Ramaiah School of Advanced Studies,
#470-P, Peenya Industrial Area, Peenya 4th Phase, Bangaluru-560058, Karnataka, India.
Email: sanketdessai@gmail.com

1. INTRODUCTION

With the ongoing extension of technology, we are witnessing the emergence of new computing machines, which will bring us far beyond the desktop PC. Devices featuring advanced connectivity and Internet functionality will soon become the standard in computing. In fact, we're on the verge of a revolution that will bring us a wave of smart, electronic devices that can be controlled, gather information, and distribute data via the web.Virtually every embedded designer is looking to use Internet, to enhance or expand the reach of embedded systems. This need for connecting devices directly into Internet has lead many great manufacturers to implement ASICs (Application Specific Integrated Circuits) or reusable libraries for microcontrollers, specially designed for this purpose. As essential elements of design reuse, IP cores are part

of the growing electronic design automation (EDA) industry trend towards repeated use of previously designed components. Ethernet continues to be one of the most popular LAN technologies.

2. THE IEEE 802.3 ARCHITECTURE MODEL

The architecture of IEEE 802.3 corresponds closely to the two lowest layers of the OSI/BR model as shown in Figure 1. The Data Link layer in the OSI/BR model is partitioned into three sublayers in the architecture in order to obtain maximum flexibility within the family of IEEE 802 standards. By doing this, various media access methods are allowed since the LLC sublayer is the same for all of them. Each sublayer in the architectural model provides a set of services that the nearest implemented higher sublayer uses. Service is the gathering name for function, procedure and variable that is made public and used by other parts of a system but the part providing them.

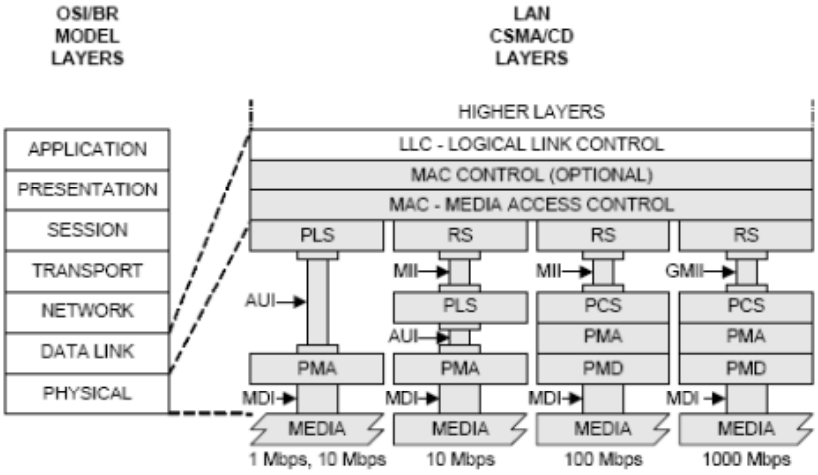

Figure 1. Mapping of Triangle Spans [1]

A service is described in its most abstract form by a service primitive. There are two generic types of primitives, REQUEST and INDICATION. The REQUEST primitive is passed from a higher layer to a lower and INDICATION vice versa. The REQUEST primitive requests a service to be initiated while the INDICATION primitive indicates an event. The architecture also defines five important compatibility interfaces (MII, GMII, AUI, MDI and, as shown in figure 1). All interfaces, but MDI, are optional and in this study, only MII and GMII are of interest.When implemented in hardware the typical solution until today has been to implement the Physical layer except Reconciliation sublayer, RS, in one device, often referred to as a PHY device, and the Data Link layer together with RS into another, often referred to as a MAC device. The MAC device also typically incorporates a bus controller suitable for the intended host system, e.g. PCI if implemented for use in a PC. Another solution that has become more common is to implement the Data Link layer and the Physical layer together in a single device in order to save space, power and cut costs.

When a two-device constellation is used, the two devices are connected to each other via the MII and/or GMII. A benefit with separate MAC and PHY devices is that one MAC device can be connected to several PHY devices. By doing that the bandwidth can be increased since the links form a single link as seen by the LLC sublayer. This type of link is referred to as aggregated link. In this work, a PHY device will be used and the FPGA will contain the RS and higher sublayers.

3. DESIGN OF ARCHITECTURE OF THE MAC IP CORE

The Ethernet IP Core consists of five modules (as shown in figure 2):

- ✓ The host interface connects the Ethernet Core to the rest of the system via the WISHBONE (using DMA transfers). Registers are also part of the host interface.
- ✓ The TX Ethernet MAC performs transmit functions.
- ✓ The RX Ethernet MAC performs receive functions.
- ✓ The MAC Control Module performs full duplex flow control functions.

- ✓ The MII Management Module performs PHY control and gathers the status information from it.

The Ethernet IP Core is capable of operating at 10, 100 or 1000 Mbps for Ethernet and Fast Ethernet applications. An external PHY is needed for the complete Ethernet solution.

The Ethernet IP Core can operate in half- or full-duplex mode and is based on the CSMA/CD (Carrier Sense Multiple Access / Collision Detection) protocol. When a station wants to transmit in half-duplex mode, it must observe the activity on the media (Carrier Sense).

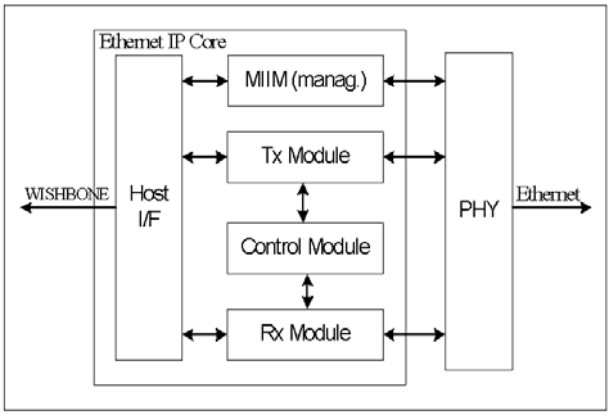

Figure 2. Ethernet MAC IP Core Architecture

As soon as the media is idle (no transmission), any station can start transmitting (Multiple Access). If two or more stations are transmitting at the same time, a collision on the media is detected. All stations stop transmitting and back off for some random time. After the back-off time, the station checks the activity on the media again. If the media is idle, it starts transmitting. All other stations wait for the current transmission to end.In full-duplex mode, the Carrier Sense and the Collision Detect signals are ignored. The MAC Control module takes care of sending and receiving the PAUSE control frame to achieve Flow control (see the TXFLOW and RXFLOW bit description in the CTRLMODER register for more information). The MII Management module provides a media independent interface (MII) to the external PHY. This way, the configuration and status registers of the PHY can be read from/written to.

a. Host Interface

The host interface is connected to the RISC and the memory through the two Wishbone interfaces. The RISC writes the data for the configuration registers directly while the data frames are written to the memory. For writing data to configuration registers, Wishbone slave interface is used. Data in the memory is accesses through the Wishbone master interface.

b. Frame Transmission Algorithm

To transmit the first frame, the RISC must do several things, namely:

- ✓ Store the frame to the memory.
- ✓ Associate the Tx BD in the Ethernet MAC core with the packet written to the memory (length, pad, crc, etc.).
- ✓ Enable the TX part of the Ethernet Core by setting the TXEN bit to 1. As soon as the Ethernet IP Core is enabled, it continuously reads the first BD. Immediately when the descriptor is marked as ready, the core reads the pointer to the memory storing the associated data and starts then reading data to the internal FIFO. At the moment the FIFO is full, transmission begins.

At the end of the transmission, the transmit status is written to the buffer descriptor and an interrupt might be generated (when enabled). Next, two events might occur (according to the WR bit (wrap) in the descriptor):

- ✓ If the WR bit has not been set, the BD address is incremented, the next descriptor is loaded, and the process starts all over again (if next BD is marked as ready).
- ✓ If the WR bit has been set, the first BD address (base) is loaded again. As soon as the BD is marked as ready, transmission will start.

c. Frame Reception Algorithm

Grid size plays an important role in both convergence and To receive the first frame, the RISC must do several things, namely:

- ✓ Set the receive buffer descriptor to be associated with the received packet and mark it as empty.
- ✓ Enable the Ethernet receive function by setting the RECEN bit to 1.

The Ethernet IP Core reads the Rx BD. If it is marked as empty, it starts receiving frames. The Ethernet receive function receives an incoming frame nibble per nibble. After the whole frame has been received and stored to the memory, the receive status is written to the BD. An interrupt might be generated (if enabled). Then the BD address is incremented and the next BD loaded. If the new BD is marked as empty, another frame can be received; otherwise the operation stops. Only frames with length greater than 4 bytes are received without an error. Smaller frames are received with a CRC error (CRC is 4-bytes long).

d. TX Ethernet MAC

The TX Ethernet MAC generates 10BASE-T/100BASE-TX transmit MII nibble data streams in response to the byte streams the transmit logic (host) supplies. It performs the required deferral and back-off algorithms, takes care of the inter-packet gap (IPG), computes the checksum (FCS), and monitors the physical media (by monitoring Carrier Sense and collision signals). The TX Ethernet MAC is divided into several modules that provide the following functionality:

- ✓ Generation of the signals connected to the Ethernet PHY during the transmission process
- ✓ Generation of the status signals the host uses to track the transmission process
- ✓ Random time generation used in the back-off process after a collision has been detected
- ✓ CRC generation and checking
- ✓ Pad generation
- ✓ Data nibble generation

e. 32 BIT Checking Algorithm

The frame check sequence field provides a mechanism for error detection. Each transmitter computes a cyclic redundancy check (CRC) that covers the address fields, the type and the data field. The transmitter then places the computed CRC in the four bytes CRC field. As the CRC field is the remainder when M(X) is divided by the following polynomial:

$$G(X) = X^{32}+ X^{26}+ X^{23}+ X^{22}+ X^{16}+ X^{12}+ X^{11}+ X^{10}+ X^{8}+ X^{7}+ X^{5}+ X^{4}+ X^{2}+ X+1,$$

where M(X) is a polynomial that covers the data bits.

In general, digital logic does not implement efficiently the division of very large number. Consequently, binary information must be converted into a more appropriate form before the CRC is used. The strings of bits to be verified is represented as the coefficients of a large polynomial, rather than as a large binary number, as shown in the following example:

$$1,1000,0000,0000,0101 = X^{16}+X^{15}+X^{2}+1$$

Typically, CRC calculations are implemented with linear-feedback shift registers(LFSRs). LFSRs use a method that yields the same results as subtraction and shift division process when the subtraction is performed without carry by the XOR function. To affect subtraction and shift division one bit at a time, you can shift through and examine each bit in the original frame of data. For the first bit of value 1, the divisor high-ordered bit is subtracted (XOR) from the dividend. That dividend bit, which is unnecessary and is not generated, is set to zero by the subtraction. The lower order bits of the divisor cannot be subtracted yet, because the corresponding divisor bits have not been shifted in. As shown in the following figure, for the simple case of the CRC-16, the algorithm is implemented by shifting the data stream into a 16-bit shift register. Register Bit(0) receives an XOR of the incoming data and the output of Bit(15). Bit(2) receives an XOR of the input to Bit(0) and the output of Bit(1). Bit(15) receives an XOR of the input to Bit(0) and the output of Bit(14). In the case of CRC-32, which is used in Ethernet header, we use 14 XOR gates, one for each coefficient of polynomial G(X). As data is shifted into the CRC circuitry, a CRC calculation accumulates in the registers. When the CRC value is loaded into the CRC calculation register, the ending CRC checksum is loaded into the CRC Register. The value loaded into the CRC Register should be zero; otherwise, the configuration failed CRC check.

In addition, as the data is 8-bit wide we take advantage of the Verilog variables in order to process 8 bit data each clock cycle. In order to calculate the CRC field, as soon as the first bit is processed, we save this

register and we immediately begin the process of the second bit. This way we achieve the serial processing of 8 bits for the CRC calculation in only one cycle.

f. RX Ethernet MAC

The RX Ethernet MAC transmits the data streams to the host in response to the 10/100/1000 received MII nibbles. The module is divided into several sub-modules providing the following functionality:

- ✓ Preamble removal
- ✓ Data assembly (from input nibble to output byte)
- ✓ CRC checking for all incoming packets
- ✓ Generation of the signal that can be used for address recognition (in the hash table)
- ✓ Generation of the status signals the host uses to track the reception process

g. MAC Control Module

The MAC Control Module performs a real-time flow control function for the full-duplex operation. The control opcode PAUSE is used for stopping the station transmitting the packets. The receive buffer (FIFO) starts filling up when the upper layer cannot continue accepting the incoming packets. Before an overflow happens, the upper layer sends a PAUSE control frame to the transmitting station. This control frame inhibits the transmission of the data frames for a specified period of time. When the MAC Control module receives a PAUSE control frame, it loads the pause timer with the received value into the pause timer value field. The Tx MAC is stopped (paused) from transmitting the data frames for the "pause timer value" slot times. The pause timer decrements by one each time a slot time passes by. When the pause time number equals zero, the MAC transmitter resumes the transmit operation.

The MAC Control Module has the following functionality:

- ✓ Control frame detection
- ✓ Control frame generation
- ✓ TX/RX MAC Interface
- ✓ PAUSE Timer
- ✓ Slot Timer

h. MII Management Module

The MII Management Module is a simple two-wire interface between the host and an external PHY device. It is used for configuration and status read of the physical device. The physical interface consists of a management data line MDIO (Management Data Input/Output) and a clock line MDC (Management Data Clock). During the read/write operation, the most significant bit is shifted in/out first from/to the MDIO data signal. On each rising edge of the MDC, a Shift register is shifted to the left and a new value appears on the MDIO.

Internally the interface consists of four signals:

- ✓ MDC
- ✓ MDI
- ✓ MDO
- ✓ MDOEN (Management Data Output Enable)

The unidirectional lines MDI, MDO, and MDOEN are combined to make a bi-directional signal MDIO that is connected to the PHY.

The configuration and status data is written/read to/from the PHY via the MDIO signal. The MDC is a low frequency clock derived from dividing the host clock.

Three commands are supported for controlling the PHY:

- ✓ Write Control Data (writes the control data to the PHY Configuration registers)
- ✓ Read Status (reads the PHY Control and Status register)
- ✓ Scan Status (continuously reads the PHY Status register of one or more PHYs [link fail status]).

The MII Management Module consists of four sub modules:

- ✓ Operation Controller
- ✓ Shift Registers
- ✓ Output Control Module
- ✓ Clock Generator

4. DESIGN OF VERIFICATION TOOL

A GUI configuration interface, created by Tcl/tk script language, is convenient for configuring optional modules, FIFO depth and verification parameters.

a. Configuration Option Module

There are three optional modules can be removed from IP core to reduce area. This GUI configuration tool is used to customize this IP core. The designed GUI window appears on screen as shown in figure 3:

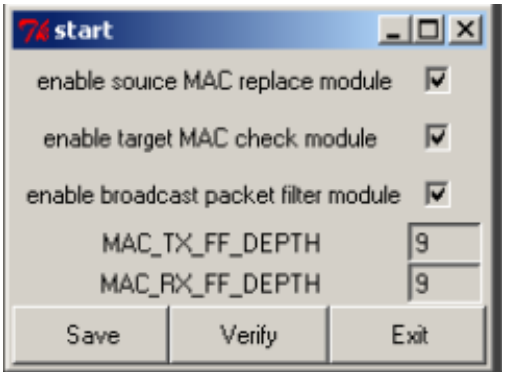

Figure 3. GUI Configuration

Clicking the check button will enable the corresponding module of IP core. The FIFO depth can be set in this window. The default setting of FIFO depth is 9, which means that the FIFO can contain 512 words. Because of the FIFO width of user side is 32bit, the actual capacity of FIFO is 512*4=2K bytes. After changing the setting, it's recommended to save the new configuration.

b. Operation of Verification Tool

Upon clicking the "verify" button of above window, a new window will appear for verification (as shown in figure 4).

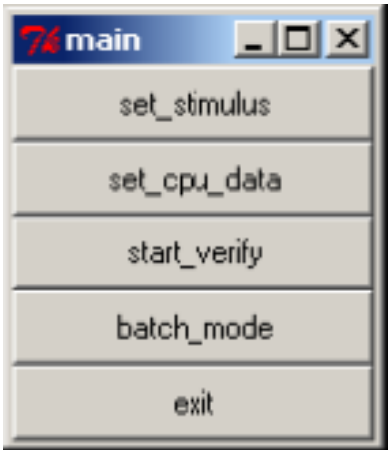

Figure 4. GUI Configuration

The first button "set_stimulus" (please refer figure 5) is used to config the parameters used for automatically generate stimulus for simulation.

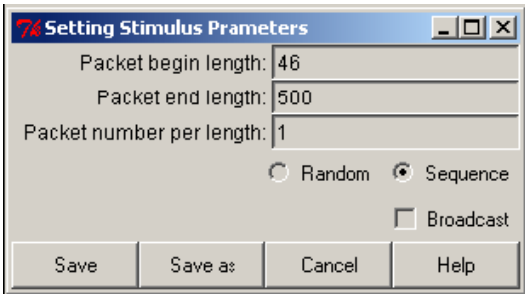

Figure 5. Stimulus Setting Window

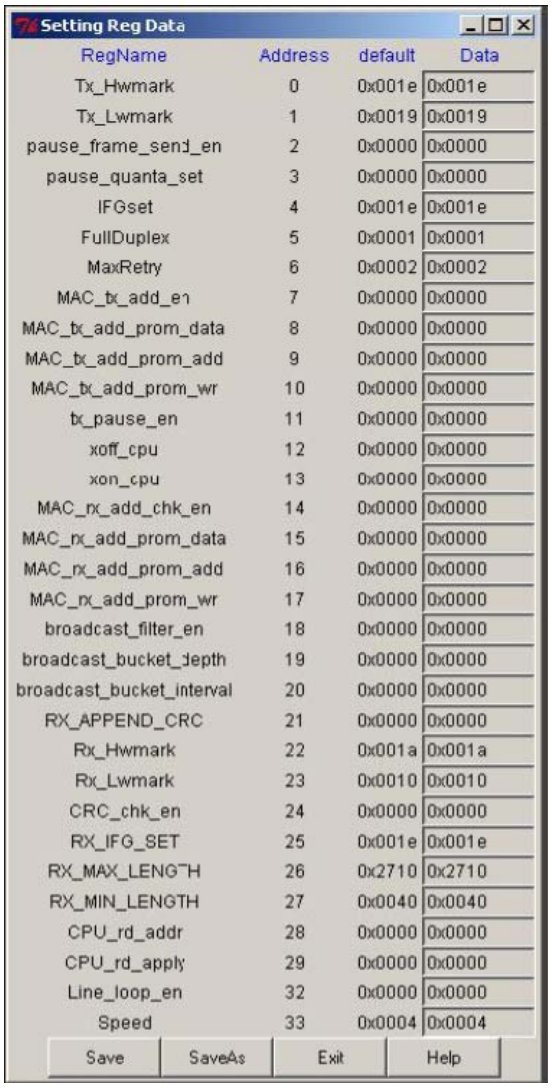

Figure 6. Register Configuration

Upon selecting “Random” mode, the generated packet length will be random in the range of “Packet begin length” and “Packet end length” as shown in figure 6. The generated packet number will be equal to “Total Gen Packet number”.If needed to generate the broadcast packets; it can be done by clicking the “broadcast” select button. The “save” button will save current configuration to “config.ini” file. Furthermore, one can use “save as” button to save the configuration as another file which can be used for “batch_mode” “set_cpu_data” button of main window is used to config internal registers. All the registers will be listed in following forms:

The “save” button will save current configuration to file “CPU.dat”.Also, user can use “save as” button to save register setting to any others files you like. Some complex operation of register such as reading statistic counters need to edit “CPU.dat” file manually. The “start_verify” of main window will start simulation. The compilation and simulation output will be printed in the following windows as shown in figure 7.

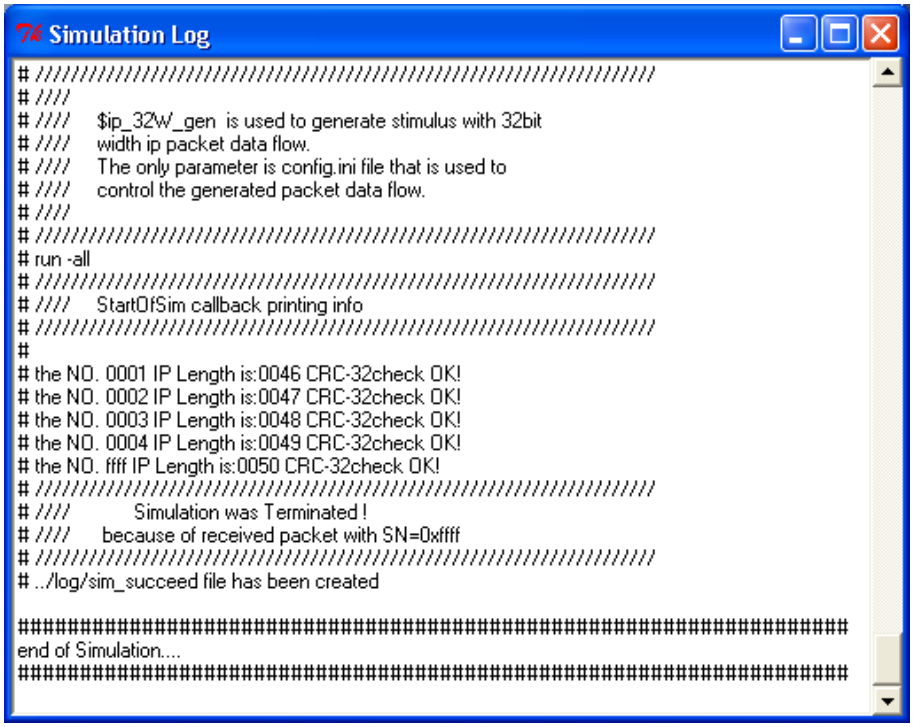

Figure 7. Verification Tool Simulator

At first, a bash script will be invoked to compile the source file .If no any error occurred; the ModelSim-simulator will be invoked to start simulation. When any error packet is received, the simulator will stop and print the data of received error packet. The “batch_mode” button of main windwos will invoke setting register data window as shown in figure 8.

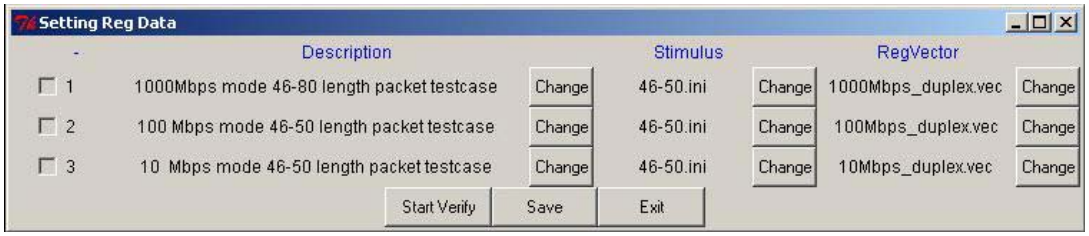

Figure 8. Register Data Setting Window

This window will be used to perform verify the IP core with several test case. In this window, user can change the description, stimulus and reg vector of a test case.

5. RESULTS AND DISCUSSIONS

Before starting to use this IP core, the following working environment needs to be made ready. At first, it is needed to setup a WinXP (recommended) or other stable operating system. In addition, Cygwin is

needed to run some bash scripts. Tcl/tk is also needed for many GUI scripts. Finally, the ModelSim is needed for simulation.

a. 1000 Mbps Full Duplex 46-1500 Length Packet Through Test

There are three optional modules can be removed from IP core to reduce area. This GUI configuration tool is used to customize this IP core. Specifiy the packet length in the setting stimulus as shown in Figure 8. Then select "start_verify" at main frame. The script will call the ModelSim simulation tools to verify the design. The packet sent to PHY will loop back to receiving port .When a "good" packet received; the following message is printed:

the NO. 0001 IP Length is:0046 CRC-32check OK!
the NO. 0002 IP Length is:0047 CRC-32check OK!
the NO. 0003 IP Length is:0048 CRC-32check OK!
the NO. 0004 IP Length is:0049 CRC-32check OK!
the NO. ffff IP Length is:0050 CRC-32check OK!

Press "set_cpu_data" button on main frame to set core to 100Mbps mode as shown in Figure 4. The speed parameter needs to be changed as per the figure 9.

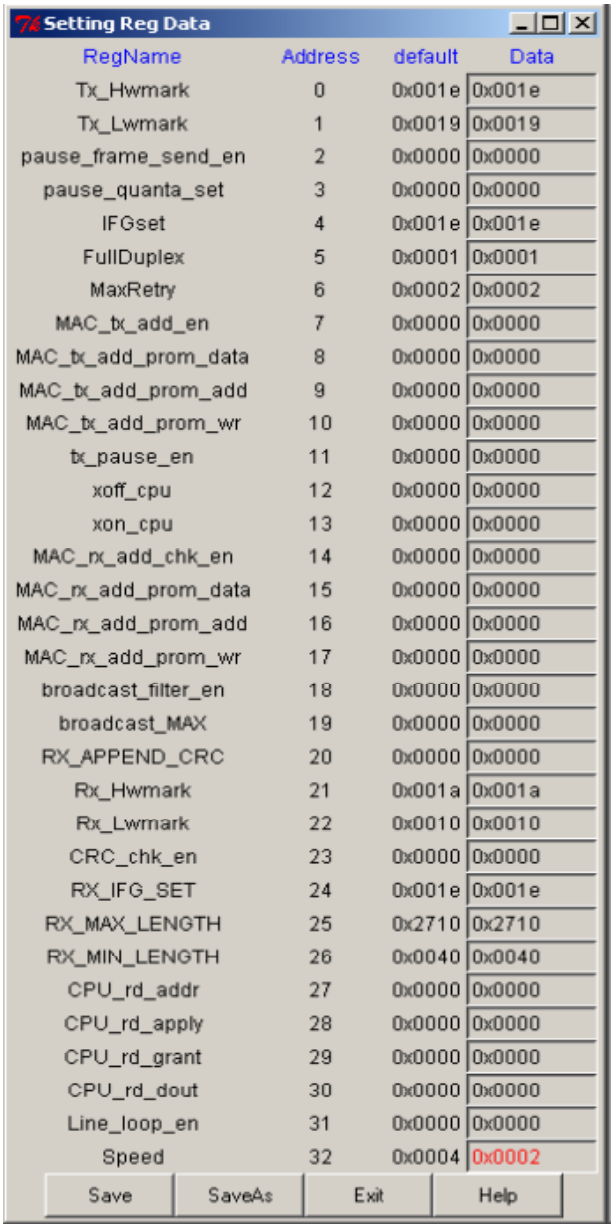

Figure 9. Setting Register for 100Mbps

After selecting the core speed, the verification needs to be started by clicking on verify button of the main window. The register setting needs to be changed as per the flow control test as shown in Figure 10.

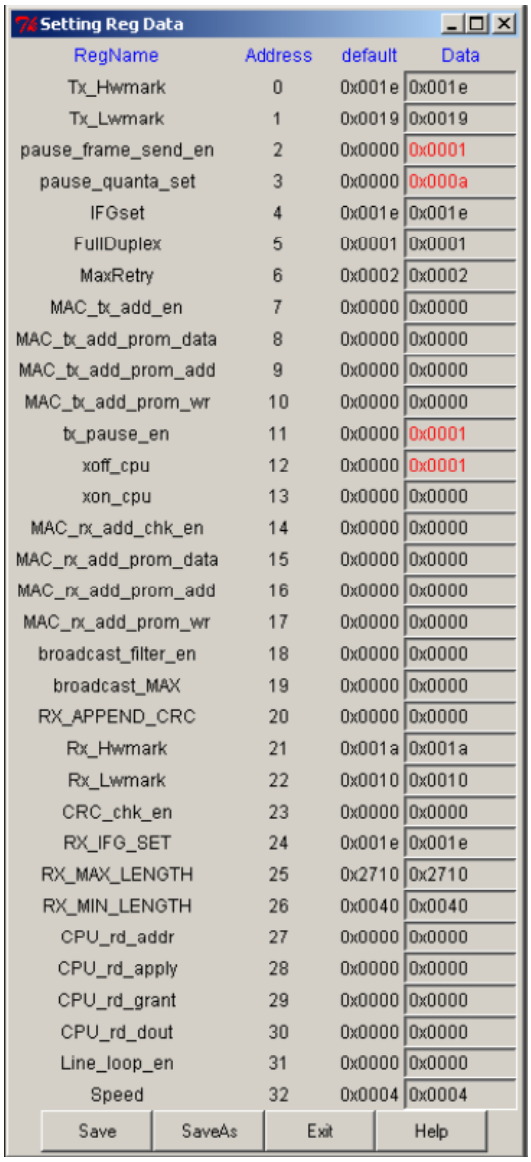

Figure 10. Setting Register for Flow Control

Starting the verify, the simulation will output
Pause frame received:
Received Pause Quanta is: 0x000a

At the same time, the transmit state machine will enter pause mode and delay packet send for 10 slot time.

b. Broadcast Filter Test

Setting Stimulus as following windows as shown in Figure 11.

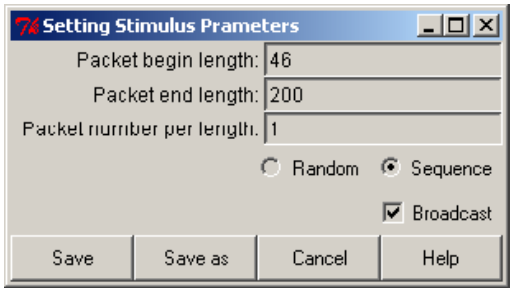

Figure 11. Broadcast Filter Test

Register settings needs to be changed as per Figure 12.

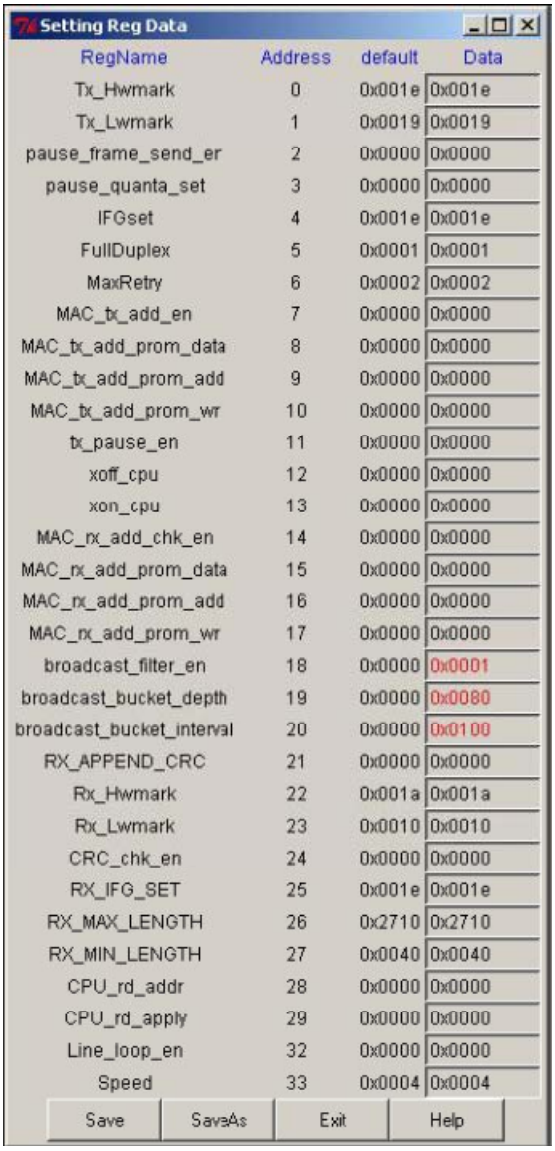

Figure 12. Setting Registers for Broadcast Filter Test

The report of simulator look likes:

```
the NO. 0001 IP Length is:0046 CRC-32check OK!
the NO. 0006 IP Length is:0051 CRC-32check OK!
the NO. 0008 IP Length is:0053 CRC-32check OK!
the NO. 000d IP Length is:0058 CRC-32check OK!
the NO. 000f IP Length is:0060 CRC-32check OK!
the NO. 0011 IP Length is:0062 CRC-32check OK!
the NO. 0025 IP Length is:0082 CRC-32check OK!
the NO. 0027 IP Length is:0084 CRC-32check OK!
the NO. 002e IP Length is:0091 CRC-32check OK!
the NO. 0035 IP Length is:0098 CRC-32check OK!
the NO. 0038 IP Length is:0101 CRC-32check OK!
the NO. 003d IP Length is:0106 CRC-32check OK!
```

Some broadcast packets were dropped, because the broadcast flow exceeds bandwidth limitation. The above experimental results on the receiver of the Ethernet MAC proved that the proposed design is capable of operating at 10/100/1000 Mbps conforming to the IEEE 802.3 Standards.

6. CONCLUSIONS

The continuing advances in the performance of network make it essential for Ethernet MAC Core to provide services that are more sophisticated rather than simple data transferring. Modern network interfaces provide fixed functionality and are optimized for sending and receiving large packets. Previous research has shown that both increased functionality in the network interface and increased bandwidth on small packets can significantly improve the performance of today's network servers. One of the key challenges for networking systems researchers is to find effective ways to investigate novel architectures for these new services and evaluate their performance characteristics in a real network interface platform. The development of such services requires flexible and open systems that can easily be extended to enable new features.The project successfully presents the design of 10_100_1000 Mbps tri-mode Ethernet MAC IP controller conforming to IEEE 802.3 specification. It is designed to use less than 2000 LCs/LEs to implement full function. It uses inferred RAMs and PADs to reduce technology dependence.

To increase the flexibility, three optional modules can be added to or removed from the project. A GUI configuration interface, created by Tcl/tk script language, is convenient for configuring optional modules, FIFO depth and verification parameters. Furthermore, a verification system was designed with Tcl/tk user interface, by which the stimulus can be generated automatically and the output packets can be verified with CRC-32 checksum.This smart solution consumes a smaller part of the targeted FPGA, and thus gives the room for other on-chip peripherals or enables the use of a smaller sized FPGA.

REFERENCES

[1] H.D. Hughes, L. Li, "Simulation model of an Ethernet", *Computer Performance*, vol.3, no.4, pp. 211-217, December 1998.

[2] R. Jaganathan, K. Underwood, and R. Sass, "A Configurable Network Protocol for Cluster Based Communications using Modular Hardware Primitives on an Intelligent NIC", *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2003)*, pp. 286–287, April 2003.

[3] P. Bellows, J. Flidr, T. Lehman, B. Schott, and K. Underwood, "GRIP: A Reconfigurable Architecture for Host-Based Gigabit-Rate Packet Processing", *10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2002)*, pp. 121–130, April 2002.

[4] B. Schott, C. Chen, S. Crago, J. Czarnaski, M. French, I. Hom, T. Tho, and T. Valenti, "Architectures for System-Level Applications of Adaptive Computing", *7th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 1999)*, pp. 270–271, April 1999.

[5] J. Riihijarvi, P. Mahonen, M. Saaranen, J. Roivainen, and J. Soininen, "Providing Network Connectivity for Small Appliances: A Functionally Minimized Embedded Web Server", *IEEE Communications Magazine*, vol. 39, pp. 74–79, October 2001.

[6] H. Fallside, M. John, and S. Smith, "Internet Connected FPGAs", *IEEE Symposiumon Field-Programmable Custom Computing Machines*, pp. 289–290, April 2000.

[7] IEEE Computer Society, ed., "IEEE 802.3: CSMA/CD Access Method", *the Institute of Electrical and Electronics Engineers, (New York, NY)*, pp. 4-16, March 2002.

[8] H. Kalte, D. Langen, E. Vonnahme, A. Brinkmann, and U. Ruckert, "Dynamically Reconfigurable System-on-Programmable-Chip", *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pp. 235–242, January 2002.

[9] J. Lockwood, "Evolvable Internet Hardware Platforms", *Third NASA/DoD workshop on Evolvable Hardware*, pp. 271–279, July 2001.

[10] RobertW. Brodersen Chen Chang, JohnWawrzynek, "BEE2: A High-End Reconfigurable Computing System", *IEEE Design and Test of Computers, vol. 22, no. 2,* April 2005.
[11] J. Addison, W. Cole, "Ethernet rules closed-loop system", *InTech, vol.45, no.6,* pp. 39-42, June 1998.
[12] Michael Barr, "A reconfigurable computing primer", *Multimedia Systems Design,* pp 44-47, September 1998.