Ethernet Controller for Host Communication with Energy-Efficient Microprocessor

Jasmine A. Berry¹, Ruzica Jevtic², and Borivoje Nikolic²

¹Department of Computer Science, Norfolk State University
²Berkeley Wireless Research Center, University of California, Berkeley

Abstract—It is evident that chips are getting bigger in capacity through progressing time, thus increasing the difficulty in testing their functionality. The most common testing approach involves the use of fully programmable gate arrays (FPGAs) to send test data from a host PC station to the chip and return the chip’s reply back to the host. This paper presents several techniques utilized to model a more robust protocol and to also provide faster synchronization of data with the chip via Ethernet connection. The results of such investigations of the circuits optimized reveal minimal logic utilization of registers and LUTs for the advantage of processor and memory performance.

Index Terms—clock synchronization, control register, digital control manager (DCM), Ethernet, frequency

I. INTRODUCTION

In accordance with Moore’s Law, the number of transistors on a integrated circuit chip increases exponentially every couple of years. With an increasing amount of transistors, the magnitude of power and frequency consequently mount as well. In the present technical society, where sustainability is often a principal topic of discussion, there are methods being developed to decrease voltage power supply in digital devices so as to consume less power [8]. This is especially apparent in the operation of embedded systems and mobile devices (i.e. cellular phones, PDAs, media players, tablet computers, etc.).

The focus of this trial is to test a chip built in the latest 28-nm technology using Verilog as the hardware description language and then implementing the test structure on a programmable logic array (FPGA). A fully programmable gate array is a programmable logic chip that provides the user with substantial flexibility in programming it to do a variety of circuit functions [4].

The 28-nm chip of interest contains an energy-efficient multi-core processor based on RISC-V (reduced instruction set computer-five). It is designed for low power operation targeting mobile applications.

There is a growing need for energy-efficient processors to increase the performance of applications such as mobile devices. In most current portable technologies and those forthcoming, on optimal voltage range approximates at 0.4-0.6 volts. Since no high volume processors are viable in this spectrum, enhancing the RISC-V chip to achieve optimal efficiency at a low supply voltage will be advantageous.

Future applications will benefit from this as they will have the need for a greater performance in processing large sets of data. Some examples of such applications include statistical machine learning, human machine interfaces, and even modeling for games and virtual worlds. One reason we are studying the RISC-V chip is because it is prime for running and parallelizing applications. This chip will serve as a great supporter for energy saving devices and their applications as its goal is to extend the life of the battery using it.

Although designed to operate at low power supplies, the final operational frequency of the chip is not formerly known. So the challenge lies in designing a test structure that will provide a range of frequencies to the chip, and at the same time be able to communicate with the host terminal at another different frequency. This will be done through special blocks that exist in FPGAs and are capable of providing a boundary between two different frequencies. In addition, another important feature necessary for all multi-core applications is the robustness of the interface with the outside world. The multi-core microprocessor should be able to accept separate commands for each of its cores since the increased process variability in the upcoming technologies leading to different performances between the cores.

The contributions of this paper include the following:
- A presentation of the host-target interface optimization though vicissitudes made in the on-chip component of the simulated RISC-V architecture
- An implementation of a DCM module to generate a faster frequency of 125 MHz to accommodate communication protocols via Ethernet from the chip.
- A discussion on the synthesis reports and final data quantities as gathered from the ISE testing environment.

II. PRELIMINARIES AND DESIGN TECHNIQUES

Before detailing the actual execution of the proposed modifications, some formal knowledge on the general components responsible for completing the implementation should be featured. The focal points of this research are on the Host-Target Interface, Ethernet protocol, clock synchronization, and finite state machines.
A. Host-Target Interface

The host-target interface (HTIF) on the chip serves as a connection between the FPGA and the chip itself. From here and throughout the rest of the paper, any reference to ‘off-chip’ correlates to the part of the test structure implemented on the FPGA. And ‘on-chip’ will refer to any entity actually located on the microprocessor. The chip as displayed in Fig. 1 comprises of several processors that contain two control registers; ToHost and FromHost. The ToHost register is responsible for storing the memory address of what is to be read from the RAM of the chip. While the FromHost register counteracts to write data to the memory and store the address value. With this, a program can be loaded into memory before the core starts and the program may execute.

Upon testing and simulating RISC-V’s architecture, it is important to note how testing with an Ethernet controller takes place in with the presence of an FPGA, particularly the flow and direction of data exchange. As depicted in Fig. 1, the host, a PC station, functions as the initial point of communication. Using Ethernet as the standard form of networking allows for the user to send test commands to the HTIF off-chip (FPGA) and then commence execution of the commands respectively.

When processing data transferred from the Ethernet through the FPGA, the HTIF sends program instructions to the memory and in the form of packets and then retrieves an ACK (acknowledge) packet in return; signaling that the command has successfully been processed.

Communication with the HTIF is done by means of the Ethernet. In the process of testing, various packets of data are sent to the off-chip then transferred to the on-chip. Having this method of network proves invaluable in achieving successful examination of the HTIF.

B. Ethernet LAN

Ethernet is the most used local area network (LAN) technology. It provides high speed communication for computers and other digital apparatuses. Specified by IEEE 802.3 restrictions, Ethernet is viable to send packets of information in a Unicast or Multicast method. Its data transfer rates range on the scales of 10/100/1000 Mbps [2].

For this study, we will be operating the Ethernet at 100 Mbps, which requires us to generate a specialized frequency of 125 MHz in the clock scheme. In this LAN, packets are sent in increments of bits according to the arrangement in Fig. 2. To adequately complete communication with the off-chip of our design, only the user data segment of the frame will be of most importance. The destination and source address have already been predefined for testing purposes. Now that Ethernet as a whole has been previewed, the protocols should be explained next.

C. Ethernet Protocol and Commands

This section will relay how we modified the standard Ethernet packet to oblige the new format we have set in place for controlled communication with the chip. By means of the payload (with a minimum of 46 bytes), we have redefined a new structure of the protocol to accommodate the commands used for testing the on-chip, listed in Fig. 3. We have established a packet of data sent from the host to the chip to be arranged in 4 words of 64 bits each. The first word comprises of a 16 bit command, 16 bit sequence number, and 32 bit size of payload. The second word relays the address (control register number and core identification number, or memory address), word 3 corresponds to sending data for the control register (or memory; since it is 128 bits wide 2 words of 64 bits are needed), and word 4 takes in bits for the memory specifically.
Before optimizing the HTIF, there were six commands instantiated for use: write memory, write control register, read memory, read control register, start and stop. Accordingly, all dispatch an acknowledge command to signal successful retrieval. As the HTIF on-chip receives a command (Note: For our testing, the on-chip received its communication 4-bits at a time from the off-chip), the appropriate protocol is triggered and handles the bits to perform the chip operation.

Taking note of the available commands, we look to optimize the HTIF by eradicating the start and stop commands. The start and stop functionality will still be a maneuver for the on-chip, as we have created a way to merge said commands into the write control register command (Write CR). Therefore, only 4 commands are left to be defined for the interface. This will be explained in more detail in Section III.

Once the communication boundaries have been set, synchronization of the clocks is needed to bring organization for the instructions to be executed in the circuit.

D. Clock Synchronization

Clocks play an important role in FPGA testing. Clocking for this study aids in several ways when performing tests on the chip. Per logic circuits, clocks are suited for the synchronization of instructions. For synchronous systems such as ours, clock signals impart a timing reference for the traffic of data [7]. Clock signals are vital to the circuitry design because they also are viewed as control signals. In a synchronous system, certain logic will be required to execute on a positive or negative edge of a clock signal. This gives evidence to the capability of clocks to bring order and priority to the execution of digital logic as there can be an issue of race condition. As for the user, clocking is beneficial in the analysis of chip operation. Users of the system are able to use clocks to predict how the chip works, detect and trace changes in circuit behavior, and debug errors if necessary.

Clock generation provides a way for synchronizing inputs and outputs of data. Through the use and observations of customized clock cycles in our test bench, a proficient simulation was made. Several clock signals were created for our design: clk, clk_offchip, clk_n, and clk_p. Between the Ethernet, FPGA, and the chip, these signals have all been coordinated to work simultaneously for the proper exchange of data at the appropriate time. We have one Ethernet clock that functions at a fixed frequency, clk_offchip that is 64 times slower than the on-chip clock, and two differential clock signals (global inputs p & n). Next we will discuss Finite State Machines, which are an essential part of control in digital architecture.

E. Finite State Machines

Finite State Machines (FSM) are mostly used for digital logic and computer programs. FSM’s are practical for modeling the behavior of schemes using a finite number of states. In general, they are a precise aid in debugging, tracing, and giving unique organization to the existing design. There are a total of 3 FSM’s used for our simulation: 2 for the off-chip and 1 for the on-chip. In the off-chip, there is one FSM for the HTIF entity and the other for Ethernet.

FSMs in context to the chip behavior allocates states for the Ethernet, off-chip, or on-chip according to the current state it is in and conditions that are true in order to advance to the next appropriate state. For instance, when a command first arrives the on-chip will be in the read_cmd state, which simply reads in the incoming bits to a data buffer. If the right conditions are met, state progression is made to cmd_decode where the current command is handled to determine the next instructions to perform. Fig. 4 displays the flow of states without transition conditions, specifically in the on-chip. Notice that each of these states has an indicator that points to the error state. Having this in the FSM proves invaluable when there is a need to debug and trace code. In our source files, the FSM’s were instantiated as seen here:

```verilog
parameter [2:0] state_read_cmd = 3'b000, state_cmd_decode = 3'b001, state_read_toend = 3'b100, state_process = 3'b101, state_cpu_req = 3'b100, state_cpu_wait = 3'b101, state_response = 3'b110, state_error = 3'b111;

reg [2:0] state, next_state;
```

*using Verilog HDL [1]
Between all of the aforesaid components, the process of moving towards the developing the design was done with feasibility. The actions taken in doing such are described in the next section.

### III. DESIGN IMPLEMENTATION

The deployment of the design using the techniques that were previously described was targeted in 1) optimizing the HTIF using Verilog code and 2) constructing a scheme to sustain the multiple clock signals incorporated throughout the different components of the module.

#### A. HTIF Optimization

In making the HTIF off-chip and on-chip more optimal, we took the route of minimizing the protocol commands. Essentially we sought to eliminate the start and stop commands from the HTIF modules. The start and stop commands were previously issued by an order of an 8-bit sequence and 32 bit address, and consequently returned an acknowledge command. Although we wanted to completely remove these two commands, we still wanted the functionality of starting and stopping the microprocessor to be present and available to use.

<table>
<thead>
<tr>
<th>Write CR cmd:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>63..........40 39......24 23........8 7......0</td>
<td>7......0</td>
</tr>
</tbody>
</table>

![Figure 5. Write Control Register command format](image)

The approach made in this was to take advantage of the Write CR command, which encompasses 8 bits for the command sequence, 32 bits for the address (register number and core ID) and 32 bits of data, as presented in Fig. 5. In order to issue a START or STOP, the cmd, core ID, and regnum had to be tailored specifically to differentiate the command from the original Write CR. In the RISC-V instruction set manual is a detailed account of the privileged control registers for the chip. For writing to the control register, FromHost is assigned the register number 17, which in 8-bit binary form is represented as 00010001 [3]. To distinguish Write CR from start and stop, a new register number which had not been already occupied had to be established. Hence, we made register number 11(eleven) the initiating and terminating numeral figure.

To commence a start or stop, the Write CR command is issued with the address set to eleven, represented in the form 16b1011 for eleven. And to make the distinction on whether to actually start or stop the control register, the least significant bit of the data segment is used. A ‘1’ bit signal start and ‘0’ bit for stop.

Fig. 6 displays the flow of condition that occurs to distinguish between Write CR, start and stop. First the command is checked. Secondly, the address number is identified. If necessary, the LSB of htif_req_data is checked as well. Htif_req_data was utilized frequently for the start and stop commands as it is responsible for storing data requested from the data buffer initially received from the Ethernet. While addr[31:0] is a bit array that stores the register number and core ID associated with a particular command.

![Figure 6. Condition Flow for Write CR, Start, and Stop commands](image)

The purpose of the exhibiting the above schematic is to provide a clear intent of the code changes and the actual cost of eliminating the commands. As it can be seen, in place of the start and stop commands are simple conditions to check predefined signals for their values to control register activity. After successfully removing these commands, the clock generation became our next objective.

#### B. Clock Generation Scheme

The next approach in this design was to synchronize the various clocks together for the proper exchange of input and output data to the chip. The primary clock signals used were clk, clk_offchip, clk_p, and clk_n. Several components were necessary in successfully ordering the clock behavior and generating a higher frequency of 125 MHz from the standard 100 MHz. Since our design involves having two frequencies, one unknown and the other fixed, a module must be created to adapt these two distinct values together.

In creating a new clock scheme, a new module referred to as the HTIF_joined was developed to transfer data between the off-chip and on-chip. The HTIF_joined was constructed to handle the multiple clocks with the assistance of a global clock buffer and digital control manager, shown in Fig.7. It serves as a function that links incoming and outgoing data together for the purpose of proper synchronization. In addition another component that was vital in creating this module was the FIFO36_72, which structured the reading and writing of data to and from the chip. All of these components are further explained in the following text.
1) \textit{FIFO36\_72}

FIFO is a First In First Out queue that is often used for data buffering in logic systems. It is essential in identifying where and when information should be read or written by means of the buffer. For our implementation, the FIFO36\_72 was used. This version is 72 bits wide with 512 deep width 36 KB FIFO RAM [4]. It utilizes Error Detection and Correction Circuitry (ECC) and subsequently served as a border between the two clocks for reading and writing. Having this FIFO in the design synchronized the access of memory among the clocks. Two FIFO36\_72’s were used for both \textit{eth\_to\_htif} (reading and writing of data from Ethernet to HTIF) and \textit{htif\_to\_eth} (reading and writing of data from Ethernet to HTIF) communication.

For the FIFO in \textit{eth\_to\_htif}, the write clock operates on the clk and the read clock on the clk\_offchip. This assignment is transposed for \textit{htif\_to\_eth}. Our design required a new reset signal to be created because there was an occurrence of errors resulting from the system reset being high (1) when shorter than three clock cycles of RDCLK (read clock) and WRCLK (write clock). For this matter, \textit{rst\_count3} was the variable used for RST in both FIFO. \textit{Rst\_count3} was generated alongside the HTIF join module and signaled to the value 1 after counting 3 clk cycles following the main rst toggling to 0. After defining this FIFO structure, the clock buffer was next to develop.

2) \textit{IBUFGDS}

IBUFGDS is a dedicated clock input buffer for the design. It takes two differential signals as clock input. P and n represent the positive and negative terminals of the clock signal [5]. Its functionality is shown in Fig. 8. Our purpose for having two clock signals is because they have a much lower swing (difference overcome between the signals) that is transmitted faster to the fan-out of the circuit. Ultimately a higher frequency will be generated since hertz is itemized at cycles per second.

3) \textit{Digital Clock Manager}

A very vital component of this design was the formation of the digital clock manager. The DCM allows us to recondition the clock signal, shift phases, and eliminate any clock skew. It is a function that manipulates clock signals and also supports other operations that include Delay Locked Loop, Digital Frequency Synthesizer, and the Digital Phase Shifter, Digital Spread Spectrum [5]. The DCM is capable of taking in values for the clock input, clock feedback, and reset, presented in Fig. 9. And it can output signals of different degrees of phase shifting.

The focus at this point in the design is to spawn a 125 MHz frequency from 100 MHz. Mathematically this is done by multiplying 100 by 5, then dividing by 4 to get the value of 125. Using the DCM functionality, we were able to utilize the \textit{CLKFX\_DIVIDE()} and \textit{CLKFX\_MULTIPLY()} methods to facilitate our frequency enhancements. Our deployment centered around the DCM\_ADV, which is simply an advanced digital clock manager circuit [5].
IV. EXPERIMENTAL RESULTS

A. Tools and Environments

Verilog was used as the hardware description language for the development of this design [1]. The execution of the source files and test benches for the HTIF_join, on-chip, and off-chip was performed in ModelSim SE 6.6a. Here, all instantiated signals were generated and captured for analysis on the logic level. ModelSim also provided waveform depictions of a customized test bench that proved significant in ensuring correct functionality of the HTIF optimization and new clock scheme.

In addition, ISE Project Navigator was utilized in synthesizing the design simulation for the Xilinx Virtex 5 FPGA/ Model XC5VLX5TFFG1136. The following quantitative data measurements were gathered from the ISE environment. RTL schematics, map and place routes, and several syntheses were taken here.

B. Tables and Graphs

i. Synthesis Report of final design using Write CR

<table>
<thead>
<tr>
<th>Slice Logic Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Registers:</td>
</tr>
<tr>
<td>Number of Slice LUTs:</td>
</tr>
<tr>
<td>Number used as Logic:</td>
</tr>
</tbody>
</table>

Table 1. Slice Logic Utilization

<table>
<thead>
<tr>
<th>Special Feature Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of BUFG/BUFGCTRLs:</td>
</tr>
<tr>
<td>Number of DCM_ADV:</td>
</tr>
</tbody>
</table>

Table 2. Special Feature Utilization

A synthesis report was taken in ISE to show how much area of the chip was used from our implementation. As shown in Table 1, approximately 4% of the logic blocks and registers are being used. Using minimum LUT’s (look-up tables) is essential in escalating the performance of the processor and memory. In the special feature utilization chart, Table 2, it is shown that 9% of clock buffers are in use with only a single DCM, which we deployed for frequency enhancements. Having such a minimal value of utilization is an advantage because it leaves a large mass of space for the microprocessor architecture and memory, should these be implemented in FPGA for the purposes of testing the functionality of the chip before its manufacture. The more area these components have to perform their utilities, the more efficient and resourceful the chip itself will be.

To ensure that our test simulation was running properly to gather the results, we had to analyze the signals in waveforms using Modelsim. Our record of analysis is detailed further.

C. Waveform Signals

Upon testing the design in Verilog, the use of waveforms allowed us to track the behavior of the logic we defined in our source code. Fig. 10 shows a subset of the plethora of signals available, namely for the Ethernet and the HTIF in the off-chip unit. From this particular depiction, we were able to observe that our rst_count3 toggles correctly 3 clk cycles rst goes to 0. Notice how significantly slower the clk_offchip is in relation to clk. It is noted that when eth_in_val (signals true when Ethernet bits are incoming) is 1 for a brief time interval, eth_in_bits (actual bits coming from Ethernet) are being transferred in 64 bits at a time. Then after some time htif_in_val (signals true when HTIF bits are incoming) goes to 1 and parses those same bits, but in segments of 4. When all the bits are received and processed correcting, an acknowledge command of bits 0110 is returned in htif_out_bits (actual bits in the output of the HTIF).

Other waves that were analyzed included those from the HTIF_join module. Fig. 11 displays the activity of the clk, clk_n, and clk_p signals. In the on-chip division of this wave window, we are able to visualize the htif_start1 signaling to 1 after the Write CR command for starting the control registered is issued. The states and next states are also visible here, alongside the submission of the command ‘0011’ for the WCR. To show how much more frequent the clock runs, the fastclk_in_val (signals true when values are incoming with the fast clock) and fastclk_in_bits (actual bits incoming on the fast clock) are signaling between the logical 1 and 0 at a more rapid speed of cycles per second. These analyses conclude our gathering of results and data and we are now able to draw a conclusion on the effectiveness of adjustments made to the system.
V. CONCLUSION

The goal of this project was to devise a more efficient HTIF and to correctly generate multiple clocks for proper synchronization of data between the Ethernet, off-chip (FPGA), and on-chip. Success was achieved in manipulating the design by voiding the start and stop commands and merging their functionality with the Write Control register command. A new clock scheme was implemented to produce a frequency of 125 MHz to accommodate the 100 Mbps data transfer rate for the Ethernet controller.

Overall the main objectives of this subdivision of research were achieved. A more robust protocol was created to help explore energy efficiency of the chip. Successful communication, independent of the frequency, was provided between the host and chip, which can work at different magnitudes of voltage and frequency from the chip. It is also important to note here that as the voltage decreases on a device, so does the frequency of clock cycles in execution. The alterations made in the system are now useful to become a part of future implementations for many potential core designs. Also, the design modifications mentioned in this paper will assist in exploring energy consumption and its tradeoffs for the many microprocessors to come.

VI. FUTURE WORK

Future plans for this project are still focused on increasing the efficiency of the RISC-V chip. One of the possible methods to approach in this is to incorporate a shift registers to store the incoming bits. As of now, the bits are housed in a data storage location called buf_ram, which happens to be intricately intertwined within the circuit. But the absence of it will significantly reduce the need for more registers on the chip, consequently increasing chip performance.

ACKNOWLEDGMENT

J.A. Berry would like to thank her Postdoctoral mentor, Ruzica Jevtic, who was instrumental and exercised the utmost patience in every stage of the process in completing this research. Thanks to Bora Nikolic for providing the opportunity to conduct research in the BWRC, to the SUPERB coordinators for organizing this experience, and to the National Science Foundation for funding this research venture.

REFERENCES