QEMU - GSOC: Implementing LASI network card and NCR 710 SCSI Controller Device Models.

Mentor: Helge Deller <deller@gmx.de>
Contributor: Soumyajyotii Ssarkar <soumyajyotisarkar23@gmail.com>

Project Overview

In this project, I aim to enhance the LASI Network Card implementation along with developing and implementing an NCR 710 SCSI Controller model for the HP PA-RISC architecture in QEMU.

LASI Network Card(Intel 82596-based): For LASI Network Card, I plan to debug, test and complete the existing implementation and make it fully functional, ensuring proper orientation to the original hardware with reference to the documentation present.

NCR710 SCSI Controller: I plan to develop a fully functional, clean and proper implementation of NCR53C710 with reference to the documentation, WinUAE implementation by Paul Brook, similar existing LSI53C895A implementation in QEMU, initial port to QEMU by Helge Deller, along with assistance from Mark Cave-Ayland.

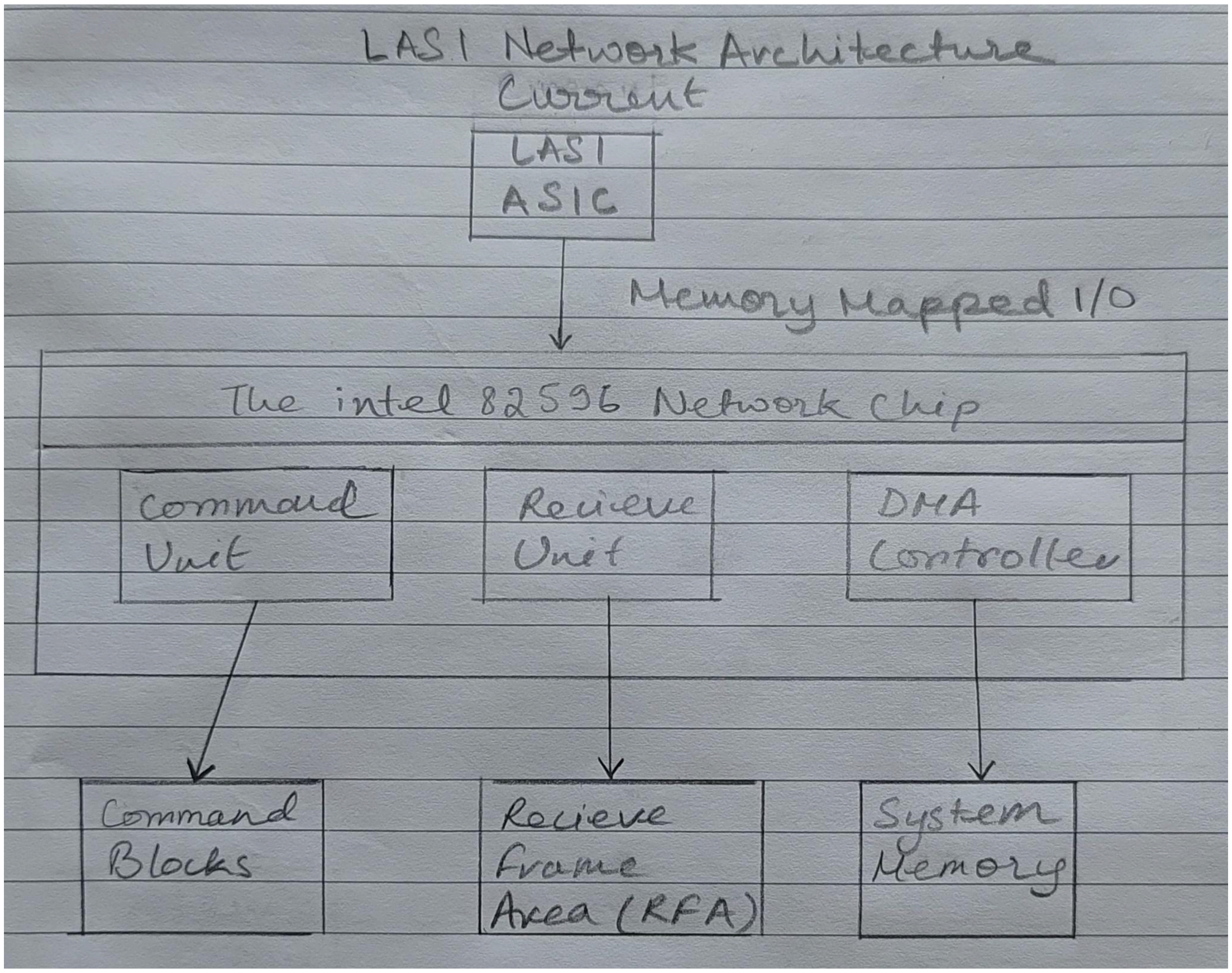
Research status as of 25 March, 2025

For LASI Network Card:
So far I have analysed and traced the current implementation of LASI Network Card in QEMU
I have analysed the Linux driver source code, taken printouts and analysed the datasheet for Intel 82596 CA and am analysing the 712 I/O Subsystem ERS Documentation.
I am in the process of closing loose ends in the current code implementation as of now, such as TODO’s and FIXME.

For NCR710 SCSI Controller:
I have also been analyzing the datasheet and WinUAE implementation to find out any mismatches further below are some mismatches that I found.
I plan to analyse the LSI53C895A implementation in QEMU for further deeper understanding.

Results so far:

For LASI Network Card:

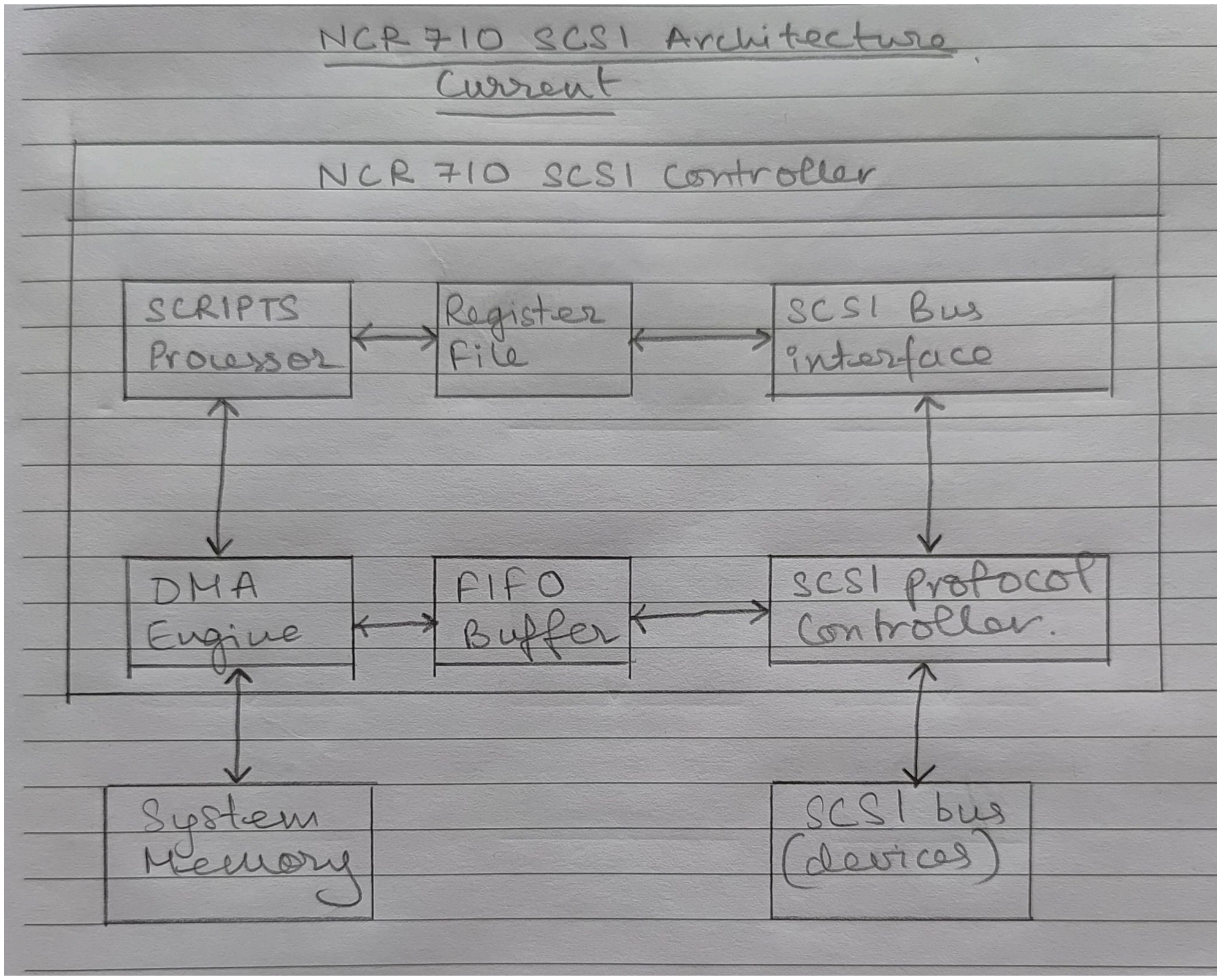


After analysing the current implementation of LASI Network Card, I have already identified few sections that would need improvement:

1. Command Unit (CU) State Machine: The current implementation of LASI Network Card does not correctly model all state transitions in the 82596's command unit, particularly for suspended state and transition between idle/suspended/active states.
2. The current DMA implementation does not properly handle: cross-page boundaries, 32-bit aligned vs unaligned transfers and the bus throttling.
3. Commands Like Diagnose, Time-domain Reflectometry and others are incomplete and not implemented properly.
4. Intel 82596 uses memory structures like SPCs.ISCPs,SCBs, etc. these addressing modes are not handled properly in the current implementation.

5. As I look through the documentation and code even more thoroughly, I will be able to identify more gaps and will adjust the timeline accordingly as necessary.

For NCR710 SCSI Controller:



Similarly after analyzing the current port for NCR710, I have identified a few sections that we should change to get it more closer to the actual NCR710 implementation according to documentation.

1. I think SCRIPTS implementation is lacking.
2. Register layout is off
3. SCSI Phase handling is inconsistent.

TIMELINE

The entire project is divided into 3 phases:

Phase 1: In this phase I plan to get the LASI Network Card working.

Phase 2: In this phase I plan to get the NCR710 SCSI Controller fully functional and working.

Phase 3: In this phase I plan to, complete whatever is left out from previous phases if any, put the finishing touches and update the documentation as necessary for QEMU.

Phase 1

In this phase I plan to focus on developing, implementing and testing LASI Network Card to make it fully functional.

Week 1

- Complete what is left for FIXME and TODO's.
- Do even more through research on LASI Network Card and NCR710 datasheets, architectural mapping to QEMU's existing architecture, and potentially create documentation that will help me in the upcoming weeks along with diagrams and visualisation.
- I will do my best to streamline the workflow for upcoming weeks to have a smoother implementation.
- Refine the setup of the testing environment.
- Discuss with Helge about the plans for upcoming weeks. Here we plan to comb through, scrutinize and modify any sections of the current proposal that might require further adjustments, this should be done to ensure a smooth execution.
- This should set the stage for the upcoming weeks.

Week 2

- I plan to implement the Rx and Tx function working properly.
- Then implement the Command Unit State Machine with proper transition
- I have noticed a few issues with memory structure addressing for both endian modes, so I plan to implement alignment handling and address endianness conversion for PA-RISC.
- Update the QEMU Mailing List with the current progress and code for feedback. (Update: 1/6)

Week 3

- HP-UX loopback testing environment setup and initial testing.
- I will add the **statistical counters** which would be updated to the kernel, this is similar to the approach other NIC takes.
- I will completely add the self test feature which makes use of the dump, diagnose, Time Domain Reflectometry and loopback.
- I also plan to implement diagnostic commands such as dump internal register, and the self-test functionality.
- Then Command Unit State Machine is missing a few states which would be useful in HPUX but not that much in Debian, so once I get the HPUX working I will add them.

Week 4

- I will add the proper global flags that sets up the configure command and implement any miscellaneous use cases defined by the global flags
- I will implement the Time Domain Reflectometry(TDR) which is quite easy for linux, however during my reverse engineering of HPUX driver I noticed its not that simple in HPUX and requires timing it accurately.

- I will get the self test feature working for HPUX 10.20 and HPUX 9 properly.
- And fix the CU and RU state transitions.
- During this period I will also implement the proper Jamming rules as specified in the documentation.
- Update the QEMU Mailing List with the current progress and code for feedback. (Update: 2/6)

Week 5

- In this week, I will implement the
- Linear Mode
- B-stepping mode
- 32 bit segmented mode.
- During my initial research I found out that there is not much support for the 82586 mode for any device drivers in most of the OS LASI supports, so making it functional along with testing would not be possible. Thus I choose to not implement this mode or get back to it on a later date if required.
- I will add proper timer and bus throttle

Week 6

Then for the HP-UX Testing and development phase.

- I will add little endian support for the 82596.
- I will fix the VM state transition variables.
- Then attempt to optimize performance where necessary.
- Post a major update to the QEMU Mailing List with the current progress and code implemented so far for feedback and the projected plan for the next 6 weeks after discussing with Helge. (Update: 3/6)

At this point during the weekend I will commit the code for LASI Network Card which will be tested.

Fair note: testing LASI and NCR710 cross compatibility will be done later during the final phase of this project.

MIDDLE CHECKPOINT

At this point I believe we will be at the middle of the GSOC period, thus I will discuss with my mentor about further plans, analyze the current progress and then sort and plan out whatever implementation is left for the next phase. I would also update the QEMU community about the progress on this project through the mailing list and IRC as necessary.

Phase 2

In this phase we will focus on NCR 710 Development and implementation, during this phase I would also incorporate Mark Cave-Ayland's assistance if I face any design choice and bottleneck during implementation process, I will start with Helge Deller's port of NCR710 as foundation, as it was a hackish implementation of LSI52C895A by Toni Wilen later imported to QEMU by Helge Deller, I will modify it to work as the actual NCR710 Controller.

Well after some discussion with Helge he suggested, and i think it would be the best move to **start from scratch** considering the fact that I have already done some previous implementation of NCR710 before the start of GSOC period. I will stick to getting my implementation working and making it fully functional.

Week 7

During the first week of NCR710 Implementation I plan to refine the existing port as mentioned above, to suite QEMU current codebase, this will be achieved by:

- I already have some code but it is not tested on any OS, so my first step of action would be to get it working on Linux and HPUX by reverse engineering the drivers.
- Next I will copy, paste and modify the
- Clean up, refactor and redo code structures based on the current QEMU's codebase standards.
- Modifying and replacing the debug statements with QEMU's trace framework.
- Then I will fix direct memory access operations to use QEMU's DMA APIs
- And create proper error handling mechanism
- Remove the UAE-specific code paths and implementation details
- Setup initial testing infrastructure.

Week 8

- I will start with implementing the correct register layout for NCR 710.
- Then fix the correct bit definitions and reset values.
- And continue to implement SCRIPTS processor framework which would include an instruction fetch mechanism, an instruction decode logic and execution environment setup.
- Update the QEMU Mailing List with the current progress and code for feedback. (Update: 4/6)

Week 9

This week I plan to implement the SCRIPTS Processor, some preliminary work which I have done already, but I would have to integrate it properly with my build and test it.

- So we will start with the implementation of:
  - Block Move Instructions
  - I/O Instructions
  - Transfer Control Instructions
  - Memory Move Instructions
- Then move on to implement Instruction sequencing with proper cycle timings
- And then implement SCRIPTS debugging facilities
- Finally I plan to add register tracing capabilities.

Week 10

During this week I will focus on SCSI Bus phase handling & DMA

- I will implement accurate SCSI Bus phase Transition

- Then implement DMA FIFO with threshold handling
- And implement DMA Chaining with proper memory alignment.
- Then Implement the Selection/Reselection with proper timing
- And add proper SCSI Message Handling
- Fix SCSI bus signal handling (REQ/ACK handshaking)
- Update the QEMU Mailing List with the current progress and code for feedback. (Update: 5/6)

Week 11

Testing and development for NCR710

- I dont think linux has support for NCR710 so i have to stick with HPUX 9/10.20 using which I have to reverse engineer the drivers to properly implement it.
- Debug device initialisation
- Test with various SCSI Devices
- Test error handling and recovery
- Test with HP-UX
- Debug HP-UX Bootloader interaction with controller
- Test with HP-UX SCSI Utilities
- Fix any OS Specific Issues

Phase 3

Week 12

- Testing the components working together
    - Boot HP-UX and Linux from SCSI Storage
    - Test Network Operations During SCSI I/O
    - And perform stress tests
- Do optimization as necessary
- Address any integration issue that may arise
- Finally update the QEMU Mailing List with the entire progress of this project, outcomes and code for feedback. (Update: 6/6)

Week 13: Documentation and Finalization

- At this point, I will do documentation for the code.
- Document register layouts and bit semantics as necessary
- Document state machines and interactions
- And prepare the patch for submission to QEMU Upstream
- Then write user documentation for configuration options, known limits and testing procedures.

Note: I have intentionally kept the Timeline a bit loose to add additional details as we proceed and moreover do adjustments where required.

Commitments:

I will be fully available throughout the month of May, from 1st June to 14th June, I will have semester end exams during which I will be unavailable. After that, I will have a month-long break from college and will be fully available during that time. For any lost progress during the exam weeks I plan to make up during this period. If any implementation remains incomplete, I will address it in a later phase or independently after GSOC. Regardless of the GSOC timeline, I am committed to completing the project.

The official project size is 350 hours, but I ponder it will require more than that. Regardless, I am prepared to commit to 5-6 hours daily and potentially more on the weekends to ensure steady progress.
Working hours might roughly range from: 10:30AM UTC - 5:30 PM UTC.

About Me:

I am Soumyajyotii Ssarkar, a Sophomore Computer Science student from India. I have been passionate about hardware and software preservation since childhood, being handed down the love for it by my father.
As architectures evolve, it is becoming increasingly difficult to emulate and maintain older systems, thus I have a strong incentive to preserve these retro hardware and software.

Some highlights of my experience with QEMU include GPU passthrough to QEMU for gaming, modifying and adjusting PCI device for passthrough, setting up virtual network topologies with QEMU, Hackintosh, Additionally I have experience with working on and debugging RetroArch codebase.

My experience

I have been coding in C for the past 3.5 years (well focused approach for the last 3 years and as a hobby before that for maybe 1 year so 3.5 years) primarily focusing on low level projects, with some experience in embedded C programming.

Here are some of my relevant projects and achievements:

- I have built a virtual machine from scratch, with its own assembly-like language all implemented in C.
- I have also developed a small scale kernel in C
- Worked on a four months long embedded C project using STM32 Microcontroller.
- Published research on Machine Learning based for System Log Anomaly Detection (with another paper pending publication on Linux System Log dataset analysis for security)
- I have also worked on and contributed to multiple C and C-Rust hybrid Codebases.
- Served as the coordinator for my college's Open Source Labs

I have also spent the past few years experimenting with the Linux kernel and QEMU.
Because of QEMU's capability to emulate such a vast range of systems, it has played a crucial role throughout my teenage years when I was a frequent distro hopper and QEMU was back then, and still is the best tool in my arsenal.