Computer Hardware Design Project
-- BATTLESHIP V1.1 --
ECE 342S

Professor Zaky

Tutor: Ryan Leduc

David Tam 123456789

Albert Chang 123456789

Introduction

The design project allowed us to extend and enhance our hardware experience we had gained during the course of the semester. Our project involved system bus access, DMA transfers, and hardware user interface design. The project proposal can be found in Appendix B.

Objective

The objective of our project was to simulate a battleship game with the use of the Motorola gizmo board. The project involved both hardware and software. The majority of the project relied on PALs and the Gizmo bus. We also used the main memory on the gizmo board to store the ship positions and target shots. The user would be able to play on a 4X4 surface and with a 2X1 size ship. The software would facilitate the communication between the hardware and the users.

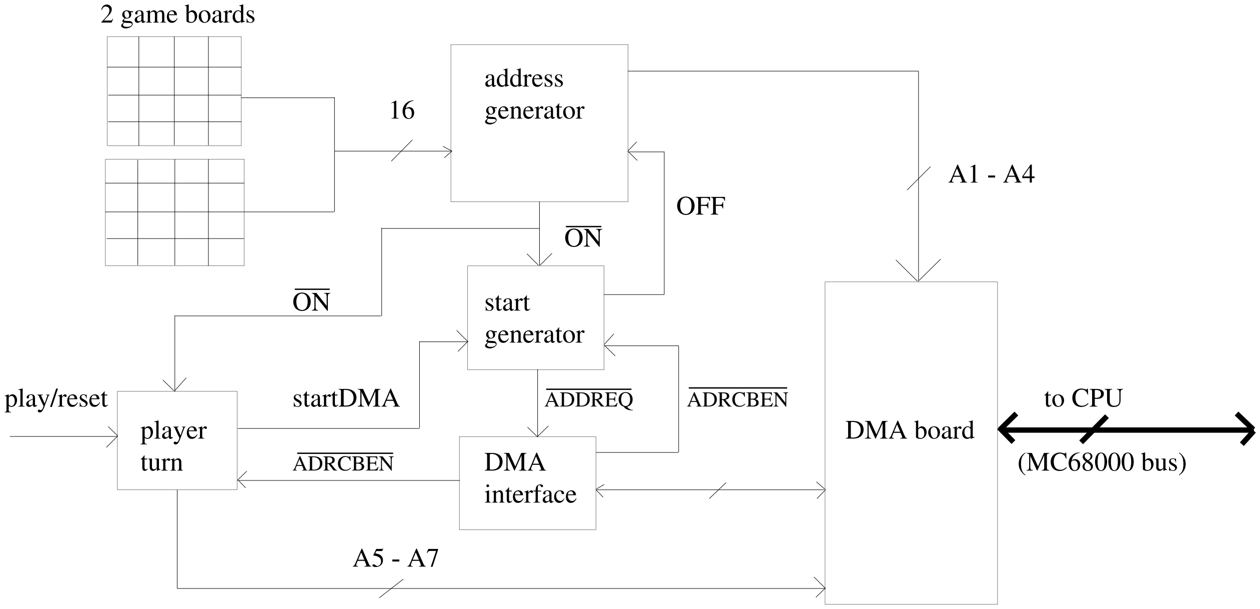
Design

Our approach to the project was divide and conquer. We decided to divide the project into six major components.

1. Game Boards	(2 user interface boards)
2. Address Generator	(4 PALS)
3. Player Turn	(1 PAL)
4. Start Generator	(1 PAL)
5. DMA Interface	(1 PAL)
6. Software	(Assembly Program)

Here I will explain how each individual part works and their relationships with the other components. For a high-level block diagram of the system, see Figure 1.

Figure 1



Component one consists of two boards from which the users may enter their input selections. Each board also includes a pen, that is pulled high (5V). The basic concept used here that the board is divided into 16 squares (made from tin foil) and each square is pulled low through a pull-down resistor. When the pen (+5V) makes contact with a square, the input pin is pulled to 5V.

Component two, the address generator, will generate an address corresponding to the square that was selected by the user. A1-A4 will be the address lines generated by this component. These lines will are connected to the DMA board. A1-A4 lines will allow us to select up to 16 different addresses (representing the board). Appendix A, Figure A1 to Figure A4 contains the Abel programs for the address generator. It consists of 4 PALs. The finite state machine (FSM) is shown in Figure A5.

To differentiate from the two boards we have our next component, the player TURN component.. It also tells our software if the game is in the setup or play mode. It consists of one PAL which generates A5-A7. A5 determines whose turn it is. A6 determines whether the board is in setup mode or play mode. A7 determines whether the board should be cleared or not. Appendix A, Figure A6 contains the Abel program for this component. The FSM is shown in Figure A7.

Component four, the start generator, consists of one PAL. It acts as a clearing house for the DMA requests that are generated by the address generator (4 PALs) and the player turn component (1 PAL). Upon receiving a DMA write request from one of the 5 PALs, this component turns off the other 4 PALs. This needs to be done because the 5 PALs all share address lines A4 to A1. After turning off the other PALs, it sends the ADDREQ signal to the DMA interface component. Appendix A, Figure A8 contains the Abel program for this component. Figure A9 contains the FSM.

The next component is the DMA interface. This consists of one PAL from Lab 3. This DMA interface handles the communication between our hardware, the DMA board, and the MC68000 bus. Appendix A, Figure A10 contains the Abel program for this component. The FSM is shown in Figure A11.

Main memory is divided into 4 sections. The first two sections store the locations of the each player's ship ($20000-$2001F and $20020-$2003F). The following two sections records where the player chose to place a shot against the opponent ($20040-$2005F and $20060-$2007F). A reset signal is sent by writing into location $20080. This where our next and final component comes in to play. The software component continuously checks whether or not a player has won by comparing the different address locations. It also continuously checks location $20080 to determine whether it needs to reset the game. Appendix A, Figure A12 contains the assembly program for this component.

We feel that our design involves both a hardware aspect as well as a software aspect. We feel that our design is very balanced.

Results

Each component was tested thoroughly and found to function as expected. However, when the pieces were connected together, they did not work quite as expected. At first, there seemed to be a number of timing problems and possibly line noise problems. Even though the individual components were found to be logically correct, the individual tests that were conducted did not account for the timing requirements.

At first, the timing problem seemed to have been caused by the lack of debouncing circuitry. Writing to one memory location often resulted in writing into several memory locations. This was not desired. Also, this problem did not occur consistently, which led us to believe that the we had a debouncing problem. However, upon closer analysis with the logic analyzer, it was seen that the hardware problems were caused partially by a bouncing input signal and by another factor. Our hypothesis of the other factor was that the problem was caused by the lack of a proper stopping mechanism in the design of the DMA interface PAL. This mechanism was added but the problem remained. Finally, we realized that some of the address generator PALs may have been interchanged by accident when they were brought back from the PAL programmer. Upon reprogramming the address generator PALs and ensuring they were placed correctly, the second cause of the problem disappeared. At this stage, it was found that the hardware components were functioning completely and correctly, except for the small problem of signal debouncing. A solution was devised but could not be implemented due to time constraints. The solution basically involved having an extra input that the user would tap to cause the hardware to begin a new input cycle.

Since the hardware was found to be fully functional (except for debouncing), the final test involved running the hardware and software components simultaneously. As implied previously, the software component was tested independently and found to be functioning as expected. However, when the hardware was combined with the software, the system did not function as expected. The hardware and software combination functioned did correctly during the initial setup phase and for the first few moves of the game. The following hardware signals were correctly detected by the software:

1. Reset game board
2. Set player one's ship location #1
3. Set player one's ship location #2
4. Set player two's ship location #1
5. Set player one's ship location #2
6. Get player one's guess location
7. Get player two's guess location

After this point in the test, the software began having problems. Upon closer analysis, we found that the problem was now completely in the software domain. The hardware was behaving correctly and writing into the proper memory locations, however the software was having trouble. Due to time constraints, we were unable to debug the software before the end of the last lab period. If we had more time, we would have been able to debug the software, and add the debouncing circuitry. In theory, once these tasks were complete, we would have had a fully functional system.

In conclusion, the overall objective of the project was partially achieved. Since the hardware component was found to be fully functional (except for debouncing), we feel we had achieved the hardware objectives of the project. At the end, the only problem left to debug was software-based. These were minor software problems that could be quickly fixed if given a bit more time. We feel that the software objective was approximately 80% achieved. If more time was available, we feel we would have had a fully functional system. . If we were to do this project again, we would independently test the software component in a more rigorous fashion, testing many possible input scenarios.

Appendix A

-- Abel and MC68000 Assembly Programs --

Appendix B

-- Project Proposal --