

Mini S08

ECE 643 Final Project

Dan Wagner
Kansas State University
College of Engineering
Department of Electrical and Computer Engineering

Dr. John Devore
Professor
Department of Electrical and Computer Engineering

December 8, 2017

Introduction

The Mini S08 project was designed to emulate a subset of instructions from the Freescale S08 microcontroller on an Altera DE2 FPGA development board. The subset was thorough and allowed the Mini S08 to function exactly like the microcontroller except for a few instructions (such as the cmp instruction) in order to keep the project simplistic. The Mini S08 was constrained to be coded in either the Verilog or AHDL languages. The seven segment displays on the board were also required to be used for displaying the Instruction Register, Data Bus, and Address Bus values.

A Floating Point Unit (FPU) was incorporated into the project to perform two operations: division and multiplication. The Mini S08 was able to take two IEEE 754 Floating Point numbers and perform a division and/or multiplication on them. The FPU was built upon the project's previous functionality, as the core framework was completed.

IEEE 754 Standard

The IEEE 754 Floating Point Standard [1] specifies the format of 32-bit floating point numbers. Each number adheres to the following format:

1. The most significant bit is the sign bit which is true when the number is negative.
2. The next eight bits are the biased exponent bits. The bias is 127.
3. The remaining 23 bits are the normalized fractional bits of the number.

When performing arithmetic, the end result may require rounding. Two main methods were investigated: round to even and round to odd. In round to even, the number is rounded as to make the least significant bit a zero (rounds to the nearest even number). If round to odd is used, the least significant bit of the number is made to be a one (rounds to the nearest odd number).

Data Flow

The data flow consists of several components and can be seen in the following figure.

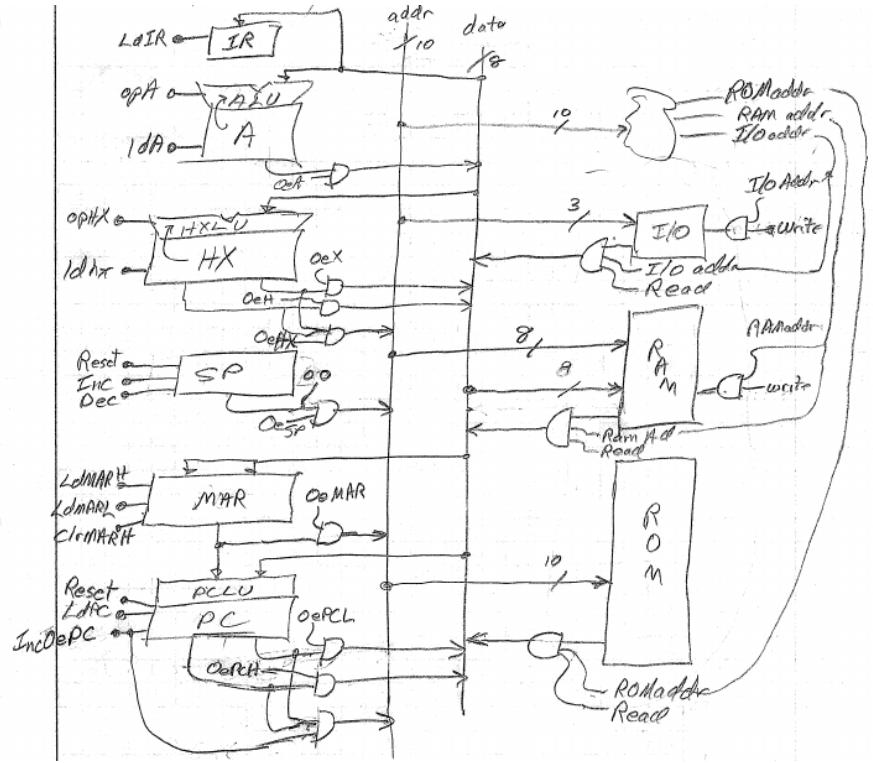


Figure 1: Data Flow of the Mini S08 Project (excluding FPU).

The Mini S08 contains two buses for information: an 8-bit data bus and a 10-bit address bus. The address bus is ten bits to accommodate the ROM memory space size.

Information on the address bus pertains to memory locations where the program or registers need to access in order to retrieve the required information. Data bus accesses are for transporting pieces of data found at these locations, or directly given (in the case of Immediate addressing).

Each register obtains their values from the data bus in different forms. The accumulator (A) is fed its value from the arithmetic and logic unit (ALU), which gets its value from the bus. The H:X register pair obtains its value in a similar fashion, as does the Program Counter (PC). Finally, the Memory Address Register (MAR) pulls its data directly from the bus.

Interaction with the address bus are more complicated. The information placed on this bus depends upon the addressing mode and the specific instruction. In Indexed addressing, the data the instruction needs is located in the memory location denoted by the HX register; thus, the HX register is put onto the address bus. Immediate and Relative addressing access the memory location pointed at by the PC, while Direct accesses the location pointed at by the MAR. Stack addressing and the rts instruction retrieves data from the location pointed at by the Stack Pointer (SP).

Input and output are coordinated between the buses and four modules: FPU, RAM, ROM, and SCI. In Figure 1, the FPU is not shown, and the SCI module is renamed I/O. Data flows to these modules from the address bus (accessing these modules requires addressing each). These modules transfer information to the data bus to be used by the program.

The control signals coordinating these transactions will be discussed in the next section.

Control Units

The Mini S08's operation was modeled as a state machine with five states. These states are shown in Figure 2.

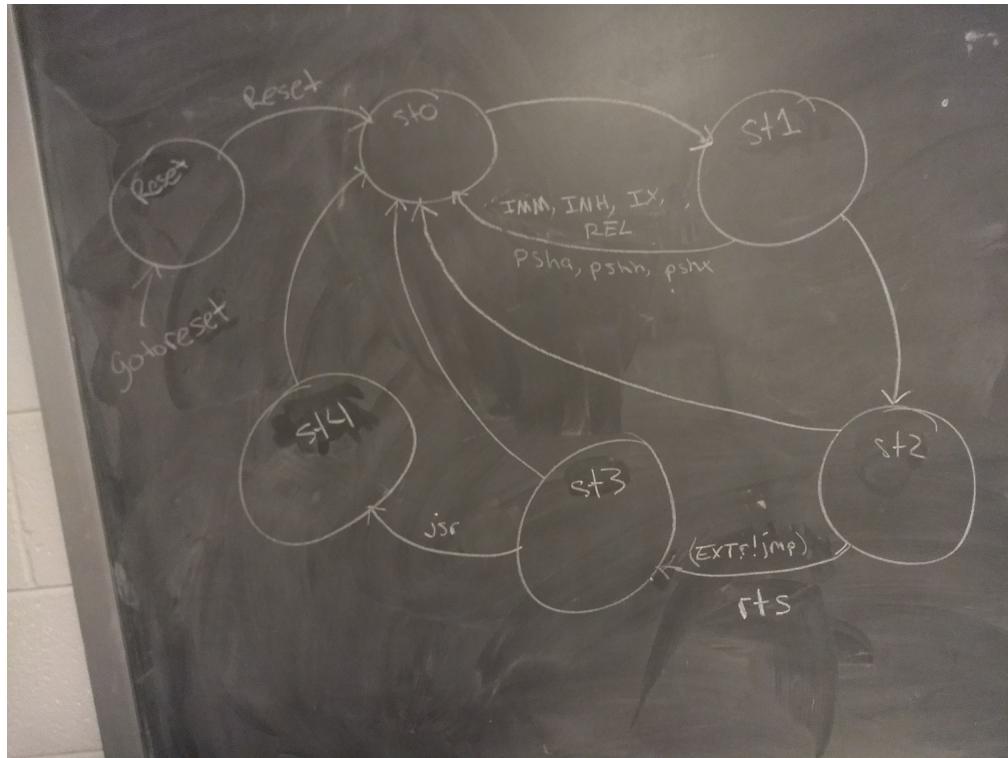


Figure 2: State Machine for the Mini S08 Project.

Reset is the initial state that the MiniS08 is in when programmed. When the reset push button is pressed, the program returns to this state and transitions to state zero. State zero is where the instruction is fetched into the IR. After fetching, the next state is state one.

State one decodes the fetched instruction and transitions to different states depending on the address mode. If the instruction is an Immediate, Inherent, Indexed, Relative, or one of the push instructions, then the instruction is completed in one cycle and the state transitions to the state zero. Otherwise, additional cycles are required for complete the execution and flow transitions to state two.

State two continues executing instructions that failed to take one cycle to complete. If the instruction is an Extended or jump instruction, or an rts instruction, then another state is required to complete it (state three). Otherwise, the instruction finishes and the state transitions back to state zero.

State three, similar to state two, resumes instruction execution that failed to complete in the prior state(s).

State four

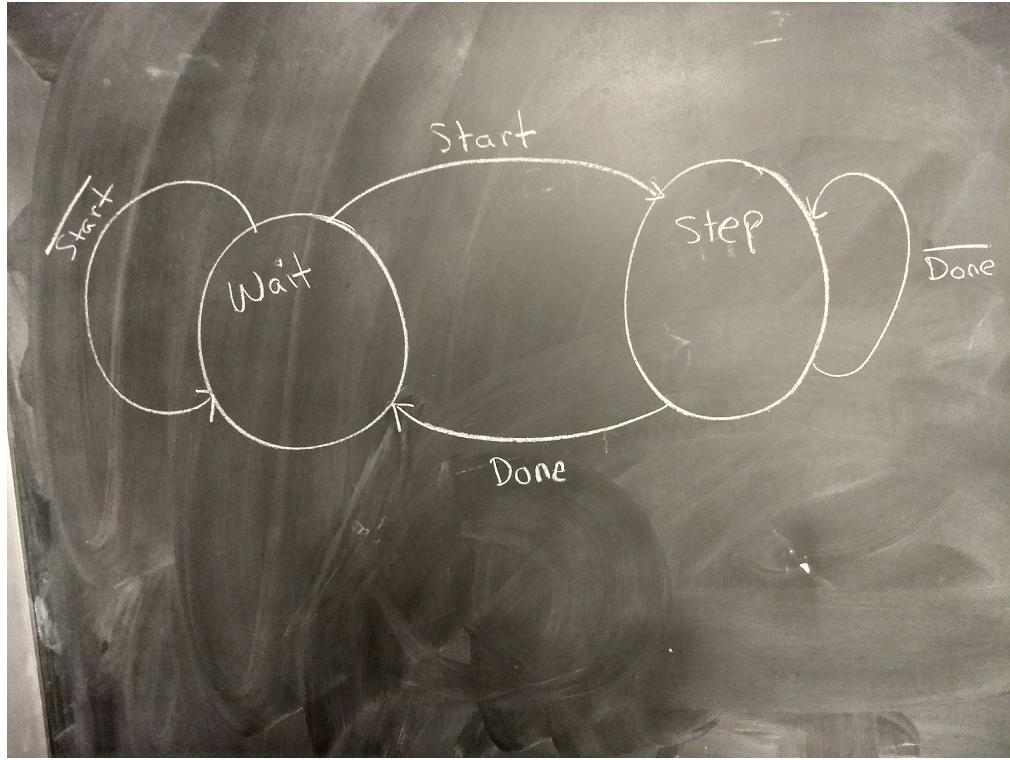


Figure 3: Divide State Machine for the Mini S08 Project.

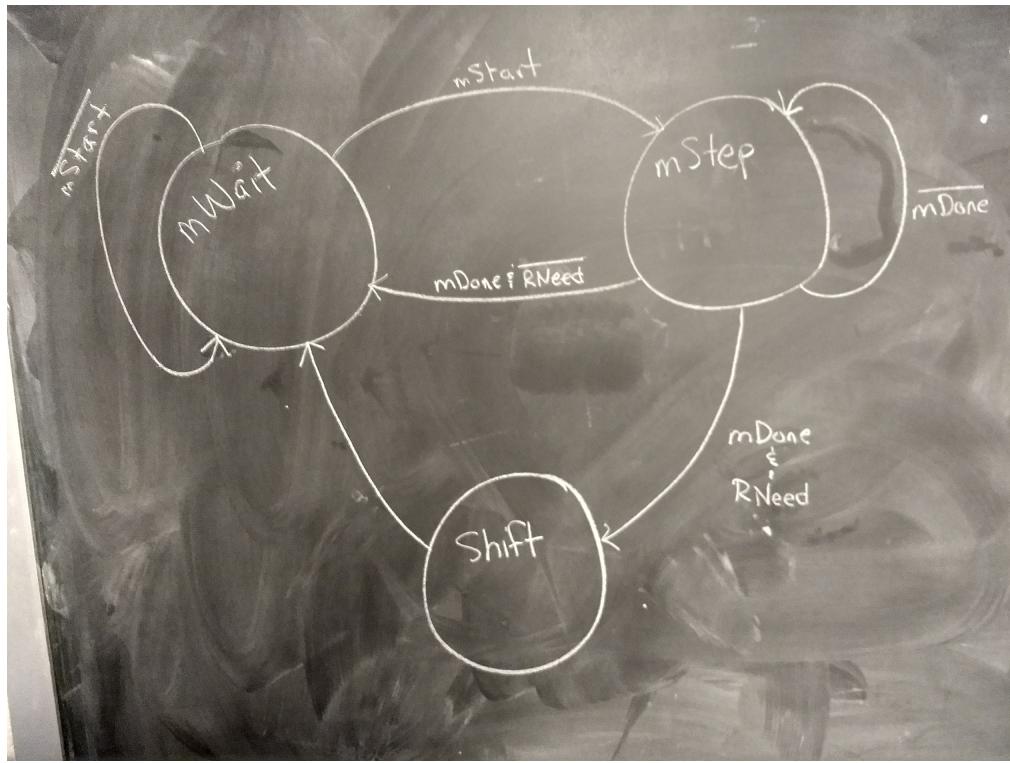


Figure 4: Multiply State Machine for the Mini S08 Project.

Conclusion

References

1. IEEE Standard for Binary Floating-Point Arithmetic IEEE Std. 754-2008, 2008.