Project Report
Motorola MC6809E

Course Number:	ECE 8440
Course Title:	System Design and Modeling
Session:	Fall 1999
Professor:	Dr. Edward L. Hepler

Student:	Flint Weller
Email:	flint.weller@ieee.org
Voice (Day):	(215) 257-6531 x1784
Voice (Evening):	(215) 884-9824

Date Assigned:	Tuesday, September 30, 1999
Due Date:	Tuesday, December 21, 1999

Introduction:

The MC6809E is a second-generation 8-bit microprocessor. It is one of the most powerful 8-bit microprocessors ever produced. This processor is the direct predecessor for the 16-bit MC68000.

Although the MC6809E uses an 8-bit data bus, the internal registers are mostly 16-bit. Figure 1 shows the datapath of the CPU.

The 16-bit registers are two index pointer registers (X and Y), two stack/index pointer registers (S and U), a program counter (PC) and an accumulator (D) that can be addressed as two 8-bit accumulators (A and B).

An 8-bit direct page register (DP) is used to break up the address space into 256 pages of 256 bytes each. An 8-bit condition code register (CC) contains information regarding the output from the ALU and interrupt masks.

There are two hidden, 16-bit registers. One stores a calculated effective address (EA), the other (TEMP) stores temporary data for memory operations.

The two hidden 8-bit registers store the instruction (IR) and postbyte (PB).

The MC6809E uses a general-purpose architecture. However, Most instructions are in the accumulator (or one operand) based instruction format. Like many general-purpose 8-bit processors of the late 1970’s, the CPU control dominated real-estate using approximately 80% of the die area. The machine is a random logic computer with essentially dynamic internal operation. The MC6809E was built on the same technological base as the MC6800 (n-channel, silicone-gate) depletion load redesign.

There are 59 fundamental instructions (Fiqure 3) that are used along with 19 addressing modes, allowing for over 1400 unique software operations.

The addressing modes are:

- Inherent (Includes Accumulator)
- Immediate
- Direct (Direct Page Register + 8-bit offset)
- Extended (16-bit address)
  - Extended Indirect (16-bit addresses)
- Relative
  - Short/Long Relative Branching
  - Program Counter Relative Addressing
- Register
- True Indirect
- Indexed
  - Constant Offset (0, 5, 8 and 16 bit)
  - Accumulator Offset (8 or 16-bit)
  - Auto Postincrement/Predecrement by 1 or 2
  - Indexed Indirect

Results:

The MC6809E VHDL design correctly performs all the addressing modes and 99% of all instructions (SYNC and CWAI are implemented but not tested). The software interrupts are fully operational however, the hardware interrupts are implemented but not tested.

My processor design was originally conceived from studying the Motorola data book instruction cycle-by-cycle performance. This resulted in an overly complex datapath. Besides the program counter, registers S, U, X and Y were provided with an increment and a decrement function built in. All registers were provided with data paths from the other registers. This was necessary to match performance with the data book. After further research, I found that Motorola occasionally sent data asynchronously to the ALU. There are many other caveats to the original 1979 processor that were necessary for it’s performance at that time.

In hindsight, I could have greatly simplified the processor by reducing register-to-register paths, performing a more intelligent instruction decode, allowing more cycles to perform an instruction (the processor clock can be divided for memory use) and replacing the state decoder logic with microcode.

Using the Kawasaki cmos_cba ASIC process, 13,499 gates were required. The longest delay time in the processor has been estimated to be 38.55 nanoseconds (a maximum clock frequency of 25 MHz would be safe). Following my interpretation of the original 1979 processor, I designed the ALU to be able to perform an 8 bit function within one clock cycle, and use two clock cycles for any 16 bit function. Unfortunately, I could not find a way to tell Leonardo about my two-cycle design.

I did not use the schematic editor for joining the multiple processes making up this processor. I found the schematic editor to be too clumsy to use productively. Instead, I wrote a VHDL structural entity to combine the processes.

The state machine diagram, shown in Figure 2, may be a bit misleading. I used discrete states for each cycle of the addressing modes and reset sequences. However, for the instruction operations, I used a generic "Execute #" state. My state decode file "statedecode.vhdl" may be unconventional being that I have a logic decode based on the individual states, but also use a decode based on the instruction which is further based on the actual state. I did this to reduce the amount of discrete states.

The Appendices included in this are:

- A: The Readme.txt file included in the tar archive "flint.tar".
- B: The structural VHDL file.
- C: the synthesis summary.
- D: Example of a software interrupt.
- E: Example of an add instruction using immediate addressing.
- F: Example of a load using indexed indirect addressing.
- G: Example of the multiply instruction.

H: Example of the reset sequence.

Bibliography:

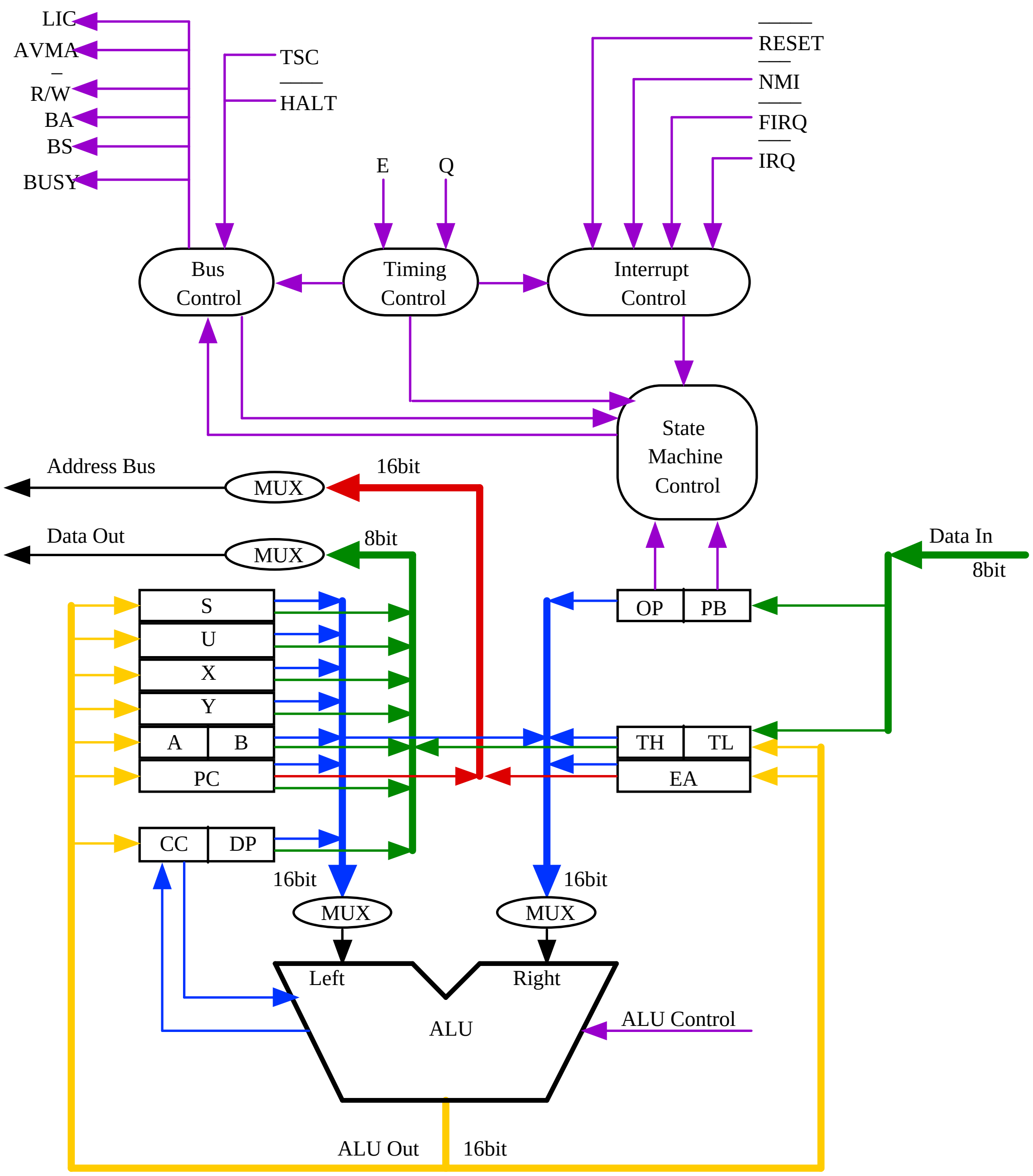
Barden, William, "TRS-80 Color Computer Assembly Language Programming", Tandy Corporation, Fort Worth.

Motorola Technical Information Center, "8-Bit Microprocessor & Peripheral Data", Motorola Inc. MOS Integrated Circuits Group, Austin, 1983.

Ritter, Terry., Joel Boney., "A Microprocessor for the Revolution, the 6809", Byte Magazine, January through March, 1979.

Tepolt, Laurence A., "Assembly Language Programming for the TRS-80 Color Computer", Tepco, Portsmouth, 1985.

Figure 1: DataPath


LIC
AVMA
R/W
BA
BS
BUSY
TSC
HALT
E
Q
RESET
NMI
FIRQ
IRQ
Bus Control
Timing Control
Interrupt Control
State Machine Control
Address Bus
MUX
16bit
Data Out
MUX
8bit
Data In
8bit
S
U
X
Y
A
B
PC
CC
DP
OP
PB
TH
TL
EA
16bit
16bit
MUX
MUX
Left
Right
ALU
ALU Control
ALU Out
16bit