AM9511 Interface for the 6502

Carl Harris
December 2024

The AMD AM9511 Arithmetic Processing Unit, first produced in the late 1970s, provides high performance fixed and floating point arithmetic and a variety of floating point trigonometric and mathematical operations as compared to available microprocessors of that time period. As such, it provides an excellent addition to any retro-computer that will be used to perform fixed-point or floating-point mathematical operations.

This article describes an interface allowing the AM9511A to be directly connected to the bus of the 6502 microprocessor. It is adapted from the work of Kissel and Currie[1] in which they developed an experiment using a 6502 microprocessor to control four AM9511A floating-point arithmetic units, and from Hart's MICROCRUNCH[2] which also used the AM9511A (Intel 8231A) interfaced with a (Rockwell) 6502.

The AM9511 uses an asynchronous interface design – its timing is not derived from the clock of the host microprocessor. This allows the AM9511A to be clocked at 4 MHz while, in the case of the original 6502, the system clock is only 1 MHz. A disadvantage of this approach is that the interface has some strict timing requirements to ensure that the actions of the AM9511 and the microprocessor are effectively coordinated, despite the lack of a shared clock.

The bus signals and timing requirements of the AM9511 are generally easy to satisfy using microprocessors such as the 8080, 8085, and Z80, all of which are designed to use multiple clock cycles for each machine cycle. In such systems, the bus control signals for addressing and read/write control are changed over the course of multiple clock cycles, with addressing (chip selection) happening first, followed by changes to read or write signals in later cycles.

In the 6502, after the falling edge of the clock, addressing and read/write selection are stabilized at effectively the same time. At the next rising clock edge, a read or write operation occurs, ending at the subsequent falling edge at which time the addressing and read/write selection signals are subject to change again.

The AM9511 requires the address input ($C/\overline{D}$) to be stable before the chip select ($\overline{CS}$) and read ($\overline{RD}$) or write ($\overline{WR}$) signals are asserted. The minimum delay between the fall of $\overline{CS}$ and the fall of either $\overline{RD}$ or $\overline{WR}$ is given as zero, which simply means that $\overline{RD}$ or $\overline{WR}$ cannot precede $\overline{CS}$. This is generally easy to satisfy in a 6502-based system.

[1] Kissel, R and Currie, J: Hardware Math for the 6502 Microprocessor; July 1985; NASA TM-86517
[2] Hart, J: MICROCRUNCH: An Ultra-Fast Arithmetic Computing System (part 1); August 1981; MICRO - the 6502/6809 Journal

The first complication involves write operations. The AM9511 requires a minimum positive delay between the rise of the $\overline{WR}$ signal (at the end of a write operation) and the rise of the $\overline{CS}$ signal. Depending on the specific AM9511 part, this delay requirement could be as high as 60 ns. Obviously, this requirement precludes direct use of the $R/\overline{W}$ signal as the input to $\overline{WR}$, since it changes state coincident with the address lines at the falling edge of the system clock – this will not provide a predictable delay.

As suggested by Hart, the solution here uses a monostable multivibrator to effectively control the length of the low pulse on the $\overline{WR}$ signal. As shown in the schematic, the circuit uses a 74HC123 monostable multivibrator. This part is used because it is easy to get a fixed delay in the monostable pulse, without regard to the length of the trigger pulse. The combination of a 68 picofarad capacitor and a 2.4 KΩ resistor provides an output pulse of approximately 75 ns which is sufficient to meet the delay requirement for any AM9511 part, while still being significantly shorter than ½ of the system clock period.

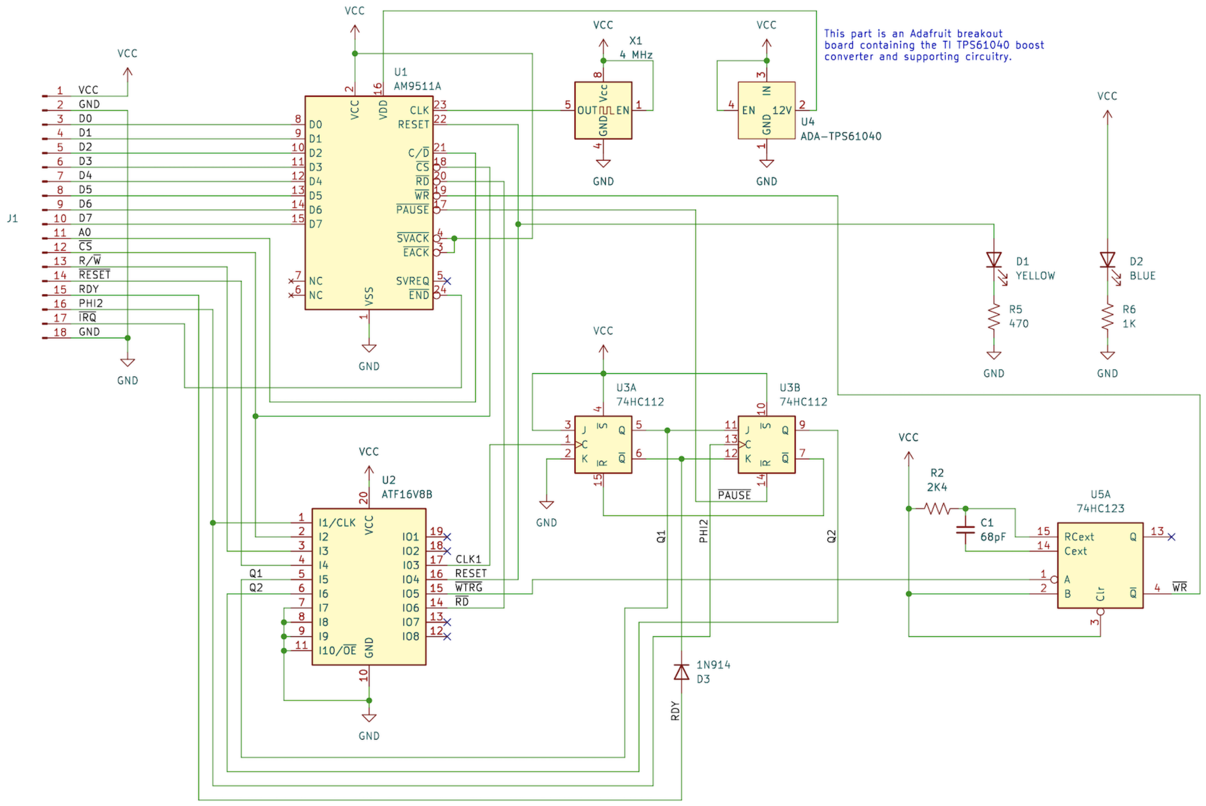

Figure 1: Circuit Schematic

In a read operation, the microprocessor sets $C/\overline{D}$, then asserts $\overline{CS}$ and $\overline{RD}$. The AM9511 then asserts its $\overline{PAUSE}$ output, which remains low until the requested data is available on the $D0..7$ outputs. When the data is available, the AM9511 releases $\overline{PAUSE}$ and the $D0..7$ outputs remain stable until $\overline{RD}$, $\overline{CS}$, or $C/\overline{D}$ changes. The minimum delay between the rise of $\overline{RD}$ and the rise of $\overline{CS}$ is given as zero – in particular, this implies that $\overline{RD}$ cannot rise *after* $\overline{CS}$.

The intent of the $\overline{PAUSE}$ signal in a read operation is to force the processor to wait until the requested data becomes available on the bus. The 6502 has an active high RDY input for this purpose. A complication arises from the fact that the 6502 observes the state of this input on the rising edge of the clock – i.e. at the instant at which it would perform the read operation, if RDY is low it will instead halt the microprocessor until the next rising edge of the clock. After $\overline{RD}$ is asserted, the AM9511 may require as much as 150 ns before $\overline{PAUSE}$ is asserted. In a 65C02-based system with a clock speed of 2 MHz or higher, depending on the time that it takes for the address and $R/\overline{W}$ lines to settle, this could occur *after* the rising edge of the clock cycle in which the read operation was initiated.

Asserting the AM9511's $\overline{RD}$ after the falling edge of the clock is further complicated by the fact that the 6502 can arbitrarily change the state of the address lines and $R/\overline{W}$ immediately at the falling edge of the clock. We must be careful to avoid asserting $\overline{RD}$ spuriously while the 6502's signals are settling – doing so could inadvertently trigger a read operation, unexpectedly disrupting the state of the AM9511's stack.

To solve both of these complications with read operations, the solution here is basically the same as used by Kissell and Currie – a pair of JK flip-flops that latch RDY at a low state at the beginning of an operation, and reset it to the high state after the $\overline{PAUSE}$ signal rises. The state of these flip-flops is also used as an enable for the $\overline{RD}$ signal.

This circuit uses an ATF16V8 programmable logic device to implement the necessary combinatorial logic. This could also be done with discrete 7400 series logic circuits, but the ATF16V8 is readily available and easily programmed using open-source tools Galette and Minipro. Figure 2 shows the logic programmed into the ATF16V8.

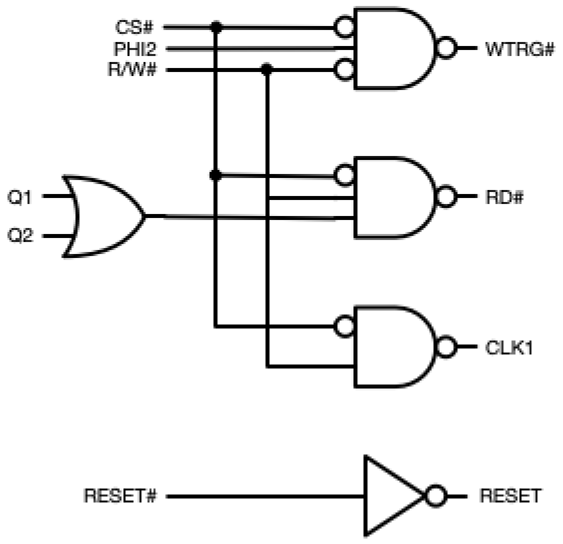

Figure 2: Combinatorial Logic

With this logic, the $\overline{WTRG}$ signal is asserted at the rising edge of the system clock ($PHI2$) if the $\overline{CS}$ signal is asserted and $R/\overline{W}$ is low. This output provides the trigger input used to produce the $\overline{WR}$ pulse for the AM9511.

The $Q1$ and $Q2$ inputs to this logic are the corresponding outputs of the JK flip-flops. These act as an enable signal to prevent $\overline{RD}$ from being asserted spuriously. The $CLK1$ output connects to the clock input of the first flip-flop. Since the 74HC112 has a negative edge trigger, the falling edge of this signal (when $\overline{CS}$ is asserted and $R/\overline{W}$ is high) clocks a logic one into the first flip-flop, and simultaneously drives the $\overline{RD}$ signal low.

The $\overline{Q1}$ output of the first flip-flop is connected to the 6502's RDY input, so a logic one here signals to the 6502 that it should wait at the next rising edge of the clock. The clock input of the second flip-flop is tied to $PHI2$, so at the next falling edge of the clock for which the AM9511's $\overline{PAUSE}$ output is not asserted, the second flip-flop will clock in a logic one. Output $Q2$ will be high, so $\overline{RD}$ will continue to be asserted even though output $\overline{Q2}$ will reset the first flip-flop, signaling to the 6502 that it can continue. The next rising edge of the system clock will complete the read operation, and the subsequent falling edge will cause the second flip-flop to clock in a logic zero (because $J2$ is low while $K2$ is high).

The only other significant elements of the circuit are the 4 MHz clock oscillator and a boost converter needed to get the +12V DC bias input needed by the AM9511. The AM9511A-4DC can run at 4 MHz. Other parts, such as the AM9511A-1DC require a clock speed no greater than 3 MHz. The boost converter is a breakout board made by Adafruit. However, any +5 VDC to +12 VDC boost converter should be fine – only about 100 mA current is needed at +12 VDC.

Observed Timing

The following oscilloscope plots show key aspects of the interface timing as observed using a WDC 65C02 clocked at 1.8432 MHz and the AM9511A clocked at 4 MHz.

Write Operation

This plot shows the timing for a write operation. A write operation targeting the data register versus the command register differs only in the value of the $C/\overline{D}$ signal (shown here as $A0$) – it is high here signifying a write to the command register. Note that the $\overline{WR}$ pulse falls soon after the rising edge of $PHI2$ but rises approximately 75 ns before either $\overline{CS}$ or $A0$ changes state. This satisfies what amounts to the most critical timing constraint of the AM9511A.

I'm

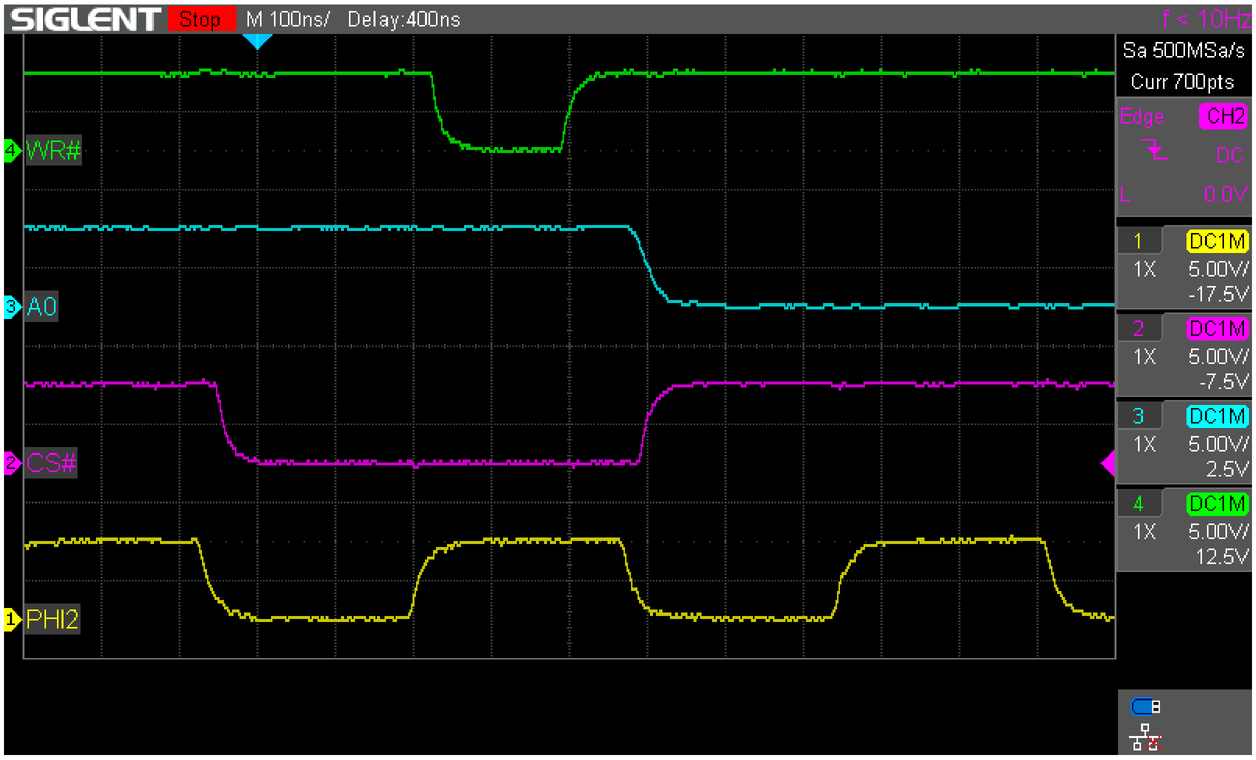

Figure 3: Write Operation Timing

Read Operation

This plot shows the timing of a read operation. The $C/\overline{D}$ input is not shown here since the scope has but four analog channels. However, this signal follows the same timing as the $\overline{CS}$ signal. Here we can see that when $\overline{CS}$ is asserted, the $\overline{RD}$ signal goes low after a short delay. In response to the read request, the AM9511 pulls $\overline{PAUSE}$ low indicating that it is busy accessing the requested data. After the $\overline{PAUSE}$ signal rises, at the next rising edge of $PHI2$, the read operation is completed. By counting the rising edges between the transitions of $\overline{CS}$ from high to low and back to high again, we can see that a total of four cycles were needed to complete the read operation.

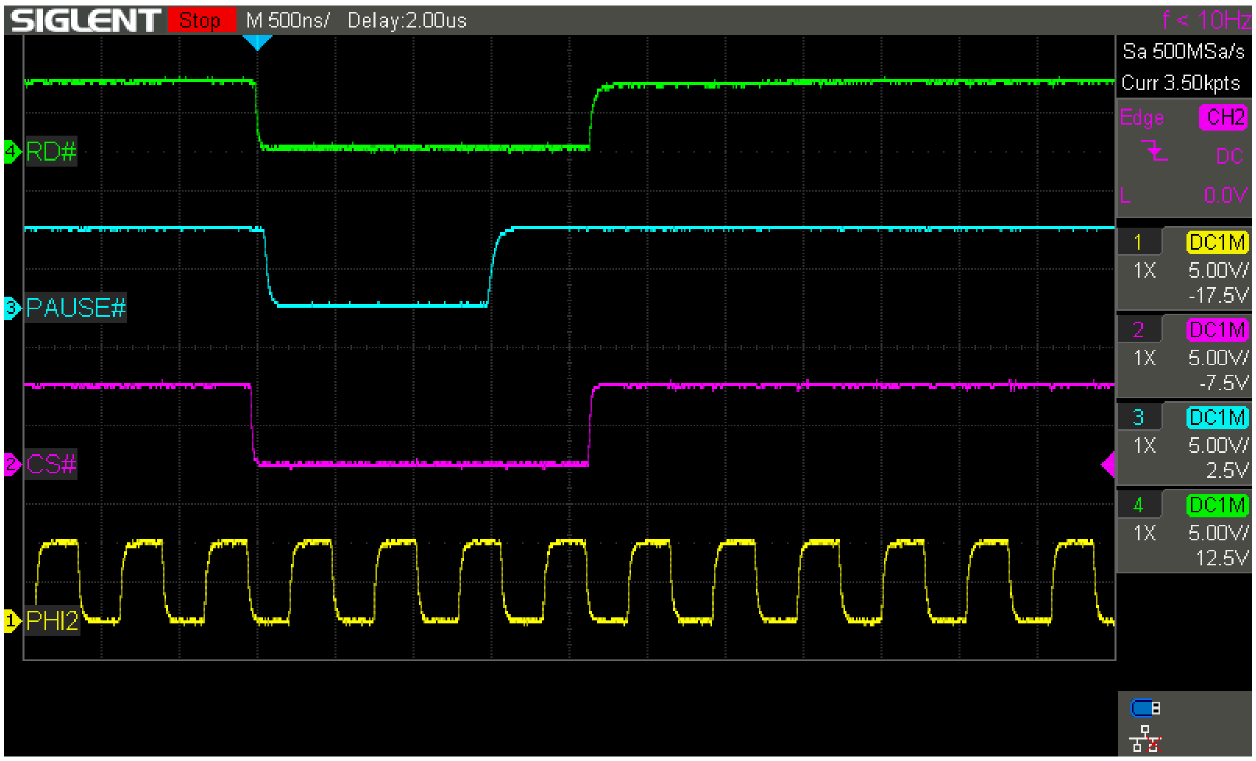

Figure 4: Read Operation Timing

Timing of RDY Signal

In order to see the timing associated with the 6502's RDY signal, in the following plot, $\overline{RD}$ has been replaced with RDY. As shown in the preceding plot, $\overline{RD}$ has essentially the same timing as $\overline{CS}$ so they can be considered equivalent.

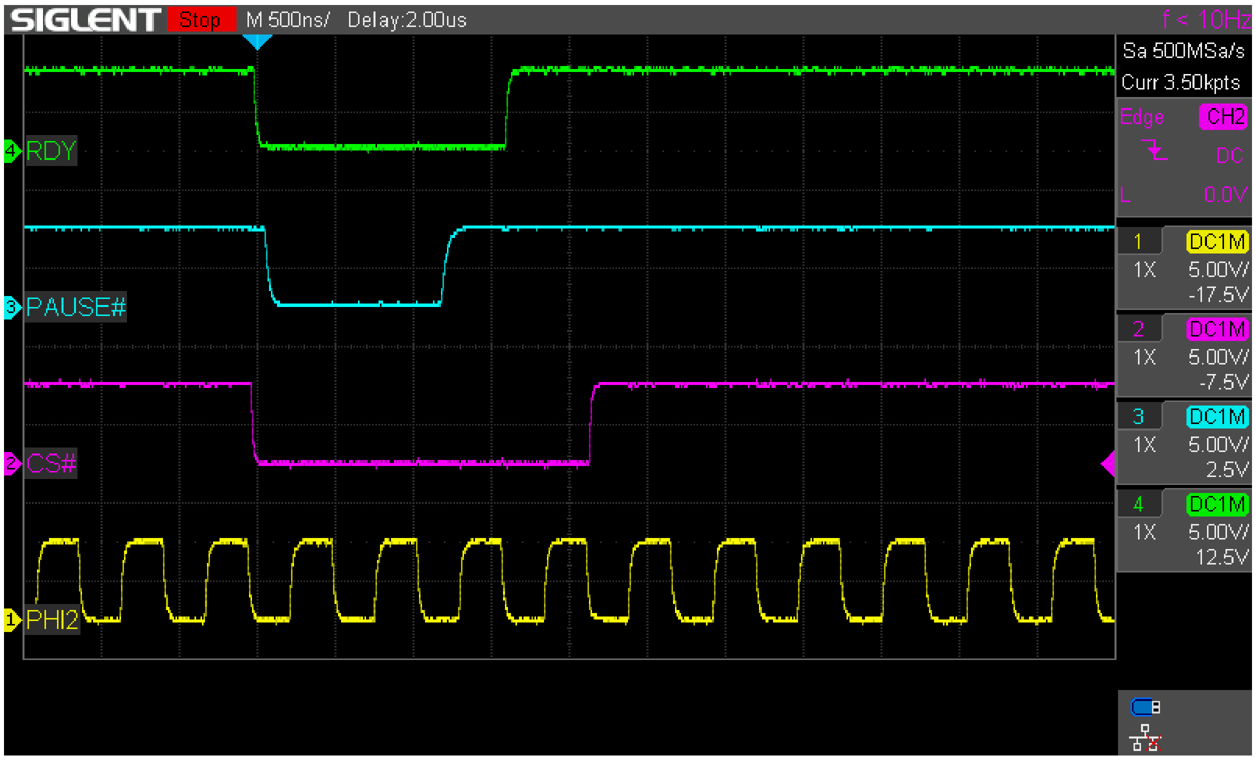

Figure 5: Timing for RDY Signal

Note that RDY falls at virtually the same instant as $\overline{CS}$ as the first flip-flop is clocked to logic one by $\overline{CS}$ whenever the 6502's $R/\overline{W}$ signal is high. Subsequently, the $\overline{PAUSE}$ signal goes low, preventing the second flip-flop from clocking in logic one on subsequent falling edges of $PHI2$.. Before the third falling edge of $PHI2$, $\overline{PAUSE}$ rises, and subsequently the second flip-flop clocks in a logic one, causing the first flip-flop to reset, and asserting RDY. This allows the read operation to be completed at the next rising edge.