Apple IIe in the Alhambra-II FPGA

J. Arias (2026)

1 Introduction

This document presents the design of an Apple-IIe clone for the Alhambra-II FPGA board. A Multimedia shield also have to be attached to that board in order to provide it with a VGA video interface, audio, and a PS2 keyboard. But more importantly, this shield also includes an 8MB PSRAM that is going to be used for most of the memory of the recreated computer. Only the specific details of this port are covered here. For more information about the general details of the Apple-II computers and previous FPGA ports please read:

`https://www.ele.uva.es/~jesus/a2.pdf`

The target computer of this recreation has the following features:

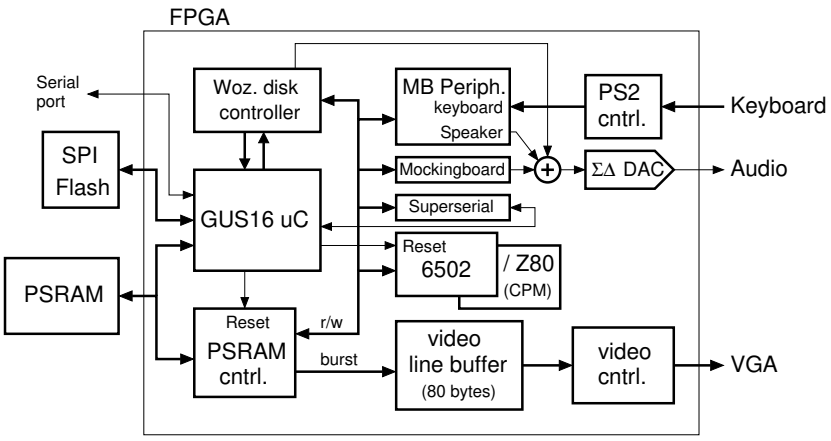
- Apple-IIe motherboard, meaning:
  - 80 column text and double Hires graphics. Virtual 80-column card on slot #3
  - 128KB of RAM, including the 16KB of a virtual Language card on slot #0
  - 16 KB of ROM
  - Lowercase text.
  - Extended keyboard with up and down arrow keys and "open" and "closed" apple keys
- Floppy disk controller card on slot #6. It includes the infamous Wozniak's state machine.

And the following optional expansion cards:

- Microsoft Softcard with a Z80 on slot #4. This alternate CPU allows to run CPM.
- Superserial Card with a 6551 ACIA and 2KB of ROM that provides a serial port for printers and modems.
- Mockingboard sound card that includes two AY-3-8912 PSGs, two 6522 VIA, and two Votrax speech synthesis chips.

The details of this Apple-IIe recreation follows:

2 Block diagram



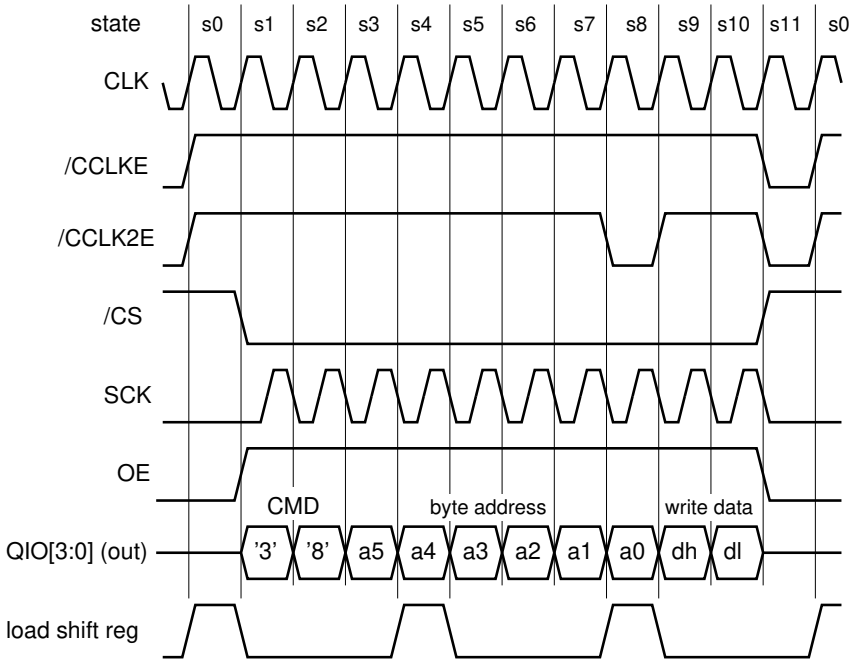
The block diagram of the Apple-IIe clone is presented in the above figure, where we can find three different cores: the main 6502 CPU of the Apple-II, a Z80 CPU for the Softcard, and a GUS16 computer with an small memory that performs several management tasks:

- Initialization of the PSRAM into Quad mode.
- Copying of the ROM image from the SPI Flash (it is stored just after the FPGA bitstream) into its proper location on the PSRAM before the starting of the 6502 CPU.
- The emulation of a pair of floppy disks. This is achieved by means of playing and recording the data of a single disc's track using a circular buffer with DMA. The content of the track is retrieved (and updated in the case of writing operations) form a remote server using a fast serial port with a simple protocol.

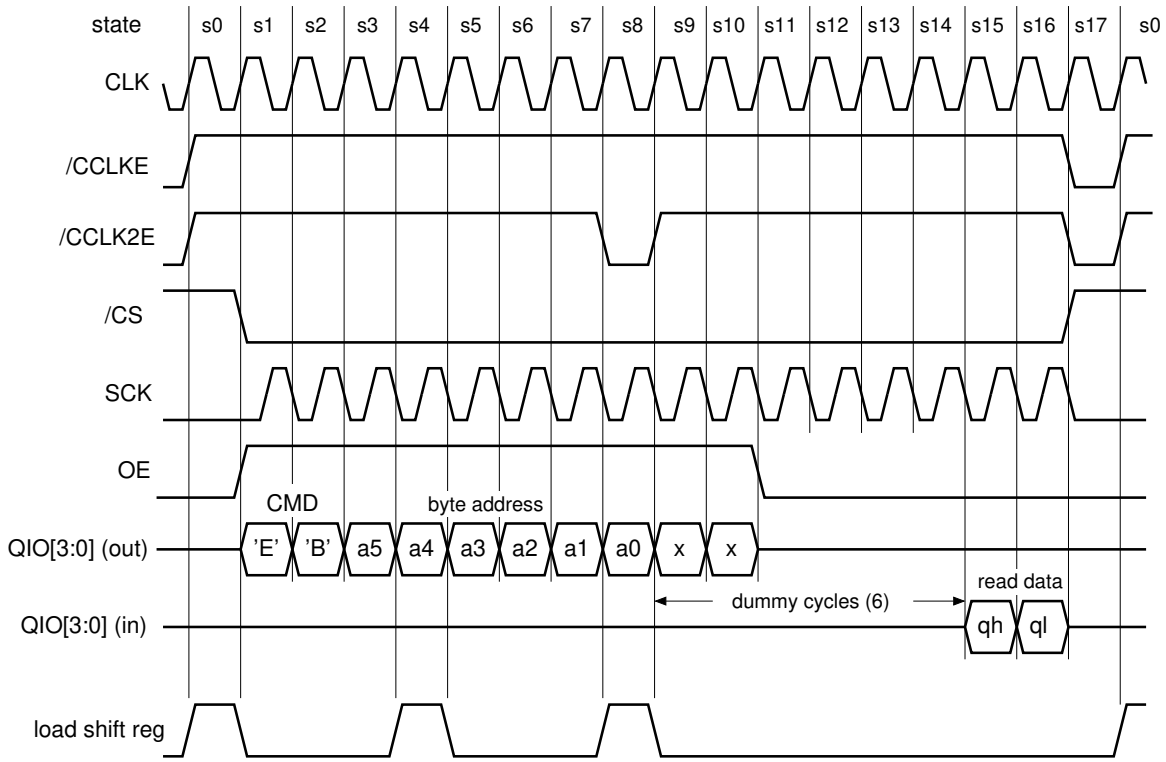
2.1 PSRAM controller

Probably the most important block of the design is the PSRAM controller, that sends the proper commands and generates the corresponding timings for up to 4 operations:

- WRITE byte. 12 cycles long. Its timing follows:

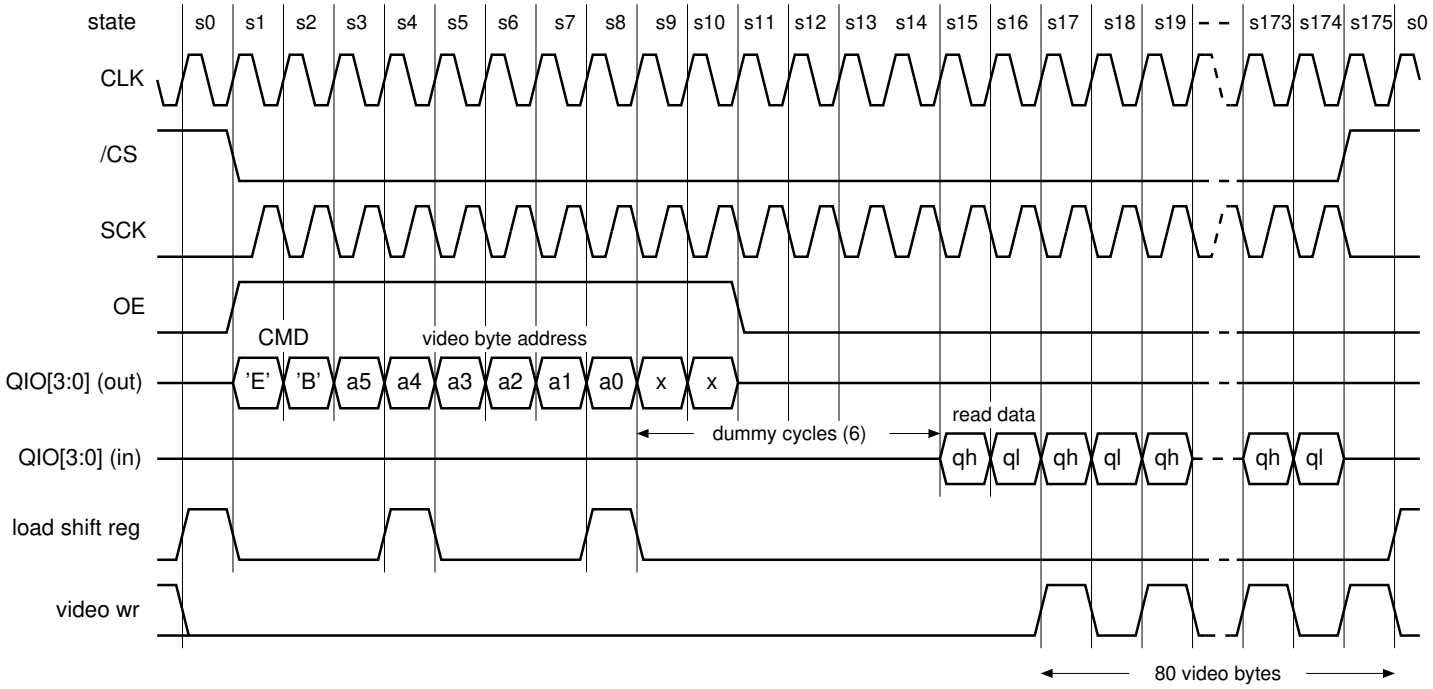


- READ byte. 18 cycles long. It is longer than writes due to the 6 dummy cycles required. Its timing follows:



/CCLKE allows the main CPU (6502 or Z80) to execute a single clock cycle when low, and there is also a second clock-enable signal, /CCLK2E, with another pulse in the middle. This second signal is used by the floppy controller and it mimics a 2MHz clock for the Wozniak's state machine. Only the READ and WRITE operations generate clock pulses for the 6502 CPU and the disk controller (the Z80 can also get pulses for some NOP operations).

The PSRAM controller uses a 16-bit shift register that shifts its data 4 bits to the left on each cycle, and therefore it has to be loaded 3 times: first with the proper command (read or write) and the highest 8 bits of the address, a second time with the lower 16 bits of the address, and the last time with the data to write in its high 8 bits (or with a dummy value in the case of reads). This same register also receives the data read from the PSRAM and its lower 8 bits holds the retrieved byte during the active /CCLKE cycle.



- BURST read (80-bytes). This operation is intended for video refresh and its timing was presented above. It reads 80 bytes from the text or graphic framebuffers and writes these data into an small buffer memory that stores a single video line in 80 columns or double Hires modes. But lines are displayed twice, so, the BURST operation has to be performed every two lines, or 64μs (or 1600 clock cycles at 25MHz). The video controller reads the buffer memory twice at its own pace during the refresh of the two lines of image (when in 40-column mode the even memory positions are skipped). The 'video_wr' signal stores the retrieved video bytes in the line buffer and advances its write address counter:

- NOP. Do nothing and keep the PSRAM signals in an inactive state during one cycle (state s0). This happens if:

  - The PSRAM controller is in a reset state during the PSRAM initialization.
  - We are waiting for the time of the next two video lines.
  - The Z80 is executing code and its MREQ signal is in an inactive state. In this particular case 'cclke' is activated, so the Z80 can execute its internal cycle (Notice that the global MREQ signal is always active if the 6502 is the selected CPU)

The operation of the PSRAM controller is driven by the video controller, that generates a pulse every two lines (except at the end of the frame, that due to the odd total number of lines per frame, 525, the time between pulses is 3 lines: 96μs) These pulses are issued at the end of the visible part of the odd video line, and they restart a counter resulting in the following sequence of operations:

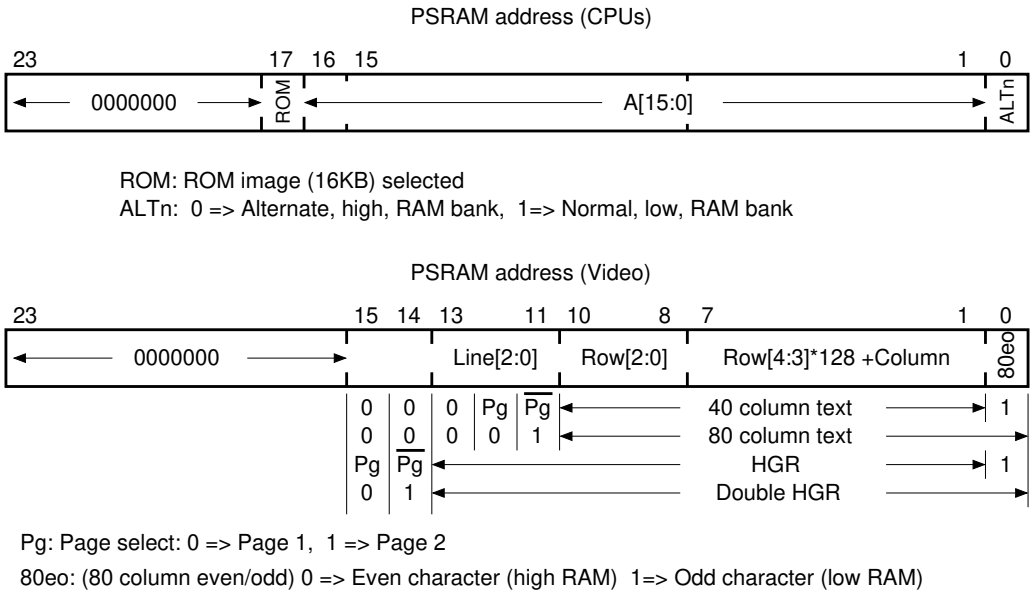
- A BURST operation that reads 80 bytes to the video line buffer.

- A sequence of READ or WRITE operations that depends on the R/W signal of the 6502 (or the RD and WR signals for the Z80 case). Each operation increments the counter until a limit is reached. This limit is 65 operations for the 6502, that gives an effective clock frequency of 65/64μs=1.015MHz, while for the Z80 the limit can be selected between 130 (2.03MHz, more or less the actual Softcard clock frequency) or 240 (overclocked, 3.75MHz Z80) during synthesis. If the Z80 is the selected processor these operations are intermixed with NOPs when the MREQ signal is inactive.

- A variable number of NOPs until the video controller restarts the counter again.

The address presented to the PSRAM controller is conditioned by the BURST read. In the Apple-IIe, when in 80-column mode, the bytes for the even columns are fetched from the alternate (high 64KB) memory, while those for the odd columns come from the main (low 64KB) RAM, but in the PSRAM burst these bytes are read sequentially. Therefore the lowest address bit has to be the selection of the RAM bank, (or it can also be viewed as the inverted higher address bit: ALTn = ~A[16]).

Also, the PSRAM is going to store the contents of the Apple-IIe ROM at an address above the main and alternate banks of the Apple RAM. At the end, the PSRAM address is arranged as shown in the following figure, both for the CPU and video operations:



As we can see, only the lower 256KB of the 8192KB of the PSRAM are actually used.

2.2 The GUS16 "Floppyserial" controller

This block is a complete GUS16-based computer with the following features:

- A GUS16-v6 core running at 25.125MHz.
- 4096 16-bit words of memory (8KB, or 16 BRAMs). The initial content of this memory is stored in the FPGA bitstream and some address ranges are forced to be read-only (These areas are intended for code storage). In particular the memory has the following usage:

Range	Size (words)	attrib.	Usage
0 to 0x1F	32	RO	Boot vector and interrupt code
0x20 to 0x3F	32	RW	Peripheral registers
0x40 to 0x7F	64	RW	Static variables
0x80 to 0x27F	512	RO	Application code
0x280 to 0xF80	3328	RW	DMA buffer
0xF80 to 0xFFF	128	RW	Stack area

- Peripheral set:
  - Simple GPIO ports for bitbanging SPI and QSPI interfaces to the SPI flash and Quad-SPI PSRAM, respectively, along with other control signals.
  - Simple but fast UART with a fractional rate divider that provides a 3Mbit/s communication channel with the PC. Can generate interrupts on RX.
  - Second UART, in this case a 6551 (ACIA) clone, attached to the Superserial card of the recreated Apple-2. Can generate interrupts on RX.
  - Edge detection and interrupt generation for the 4 phases of the emulated "stepper motor" inputs. An interrupt routine samples these signals and updates the current track position of the emulated drives.
  - A circular DMA buffer with 3196 words, or 51136 bits, for the current disk track. Notice this buffer is actually a bit smaller than the space reserved for it (the last 132 words aren't used by the DMA). This was done in order to fine tune the rotational speed of the emulated drive. When drives are ON

the content of this buffer is played continuously, starting with the MSB bit of the first word, unless the disk controller activates its write signal, resulting in the incoming floppy data being stored into the track buffer at its current position (probably after reading the address mark of the desired sector to write). When track changes the data of the new track is read from the PC using the serial port. Also, if the track was previously written, the track data is sent to the PC before reading the new track (This is also checked when the drive is turned off). The format of the read and write data streams is different:

* The data coming out of the "Floppyserial" (read signal) is coded as a Return-to-Zero signal, with 1μs high pulses followed with 3μs low in the case of ones, or a plain 4μs low signal for zeroes. This is the kind of signal expected at the floppy's read amplifier.
* The data sent to the "Floppyserial" (write signal) is coded as NRZI, meaning all bits last 4μs but an one results in a level change in the write signal whereas a zero results in the same level as before. This is the kind of signal sent to the floppy head during writes (the actual signal is a bipolar one, resulting in the reversal of the head current and the magnetic field in the disk when an one is written)

In the recreated Apple-II the CPU clock is no longer periodic and the same happens with all the DMA logic of the floppyserial computer, that depends on a clock-enable signal, 'cclke', in order to advance its state. Thus, the times stated before have to be translated as: one 'cclke' pulse for each microsecond elapsed.

2.2.1 The serial protocol

The communication protocol used by the "Floppyserial" controller is very simple. The data rate is 3Mbit/s, the data width is 8 bits, no parity is used, and a single stop bit is sent. All data transactions start with a command byte followed by parameters or data. Some commands expect an answer from the PC. If that answer isn't received during a short time a timeout error is generated and the command aborted. The current commands sent to the PC are:

Command	CMD	Param.	Data	Answer
PING	0x80	-	-	1 byte: Disk write-protect bits
READ	0x81	DSK/TRK	-	6656 bytes: Track image
WRITE	0x82	DSK/TRK	6656 bytes: Track image	-
SSBYTE	0x83	-	Serial Byte	-

And there are also commands sent from the PC to the "Floppyserial" computer. These are

Command	CMD	Param.	Data	Answer
SSBYTE	0x83	-	Serial byte	-
SSCONF	0x84	4 config bytes		

- In the PING command an answer byte is sent by the PC with the write protected status of the two emulated disk on bits #0 and #1. A value of 0 means the disk is write-protected while an 1 means the disk can be written. Bits #2 to #7 are ignored. This command is sent to the PC quite often in order to test the availability of the disk images.
- The READ and WRITE commands include an additional byte with the number of the track in its bits #0 to #5, and the disk drive in its bit #6. Bit #7 must be zero.
- The SSBYTE command transfers a single byte from the Superserial port to the PC or on the opposite way.

- The SSCONF command is only sent from the PC and includes 4 bytes with the following data:
    1. Byte for the Command Register of the Floppyserial's ACIA with the Parity settings.
    2. Byte for the Control Register of the Floppyserial's ACIA with the Baud rate, number of data bits, and Stop bits.
    3. Settings of the Superserial Dip Switch #1
    4. Settings of the Superserial Dip Switch #2

2.3 Sound

The Apple-2 clone has three sound sources:

- The motherboard speaker flip-flop.
- The Floppy emulation noises that include a disk spinning noise and track change pulses.
- The Mockingboad with its two AY-3-8912 programmable sound generators (and its two Votrax speech synthesis chips).

All these sound sources are mixed together resulting in 12-bit samples at about 1MHz sampling rate. These samples are the input for a 12-bit sigma-delta DAC that generates a single bit signal with the audio encoded into its average value.