AppleFPGA Version 1.0

Legal Stuff

Please report bugs to the author, but before you do so, please make sure that this is not a derivative work and that you have the latest version of this file.

1. Description

AppleFPGA is a System on a Chip (SOC) based on an unenhanced Apple][e using the Digilent Spartan 3 Starter board. The Spartan 3 Starter board includes a Xilinx Spartan 3 XCS3s200 FPGA and 1 Meg of static RAM (SRAM). See the Digilent web site for more details of the Starter board. There is no 6502 processor or other peripherals. All functions are derived using the Hardware Description Language (HDL) Verilog.

AppleFPGA has most features that a real Apple][e uses. The memory includes 64K main memory with an additional 448K. There are two types of storage, dual floppy disk emulated with 512K of SRAM and a bootable Serial Virtual Drive (SVD) that supports up to 32Meg of storage. A PS2 compatible keyboard appears to the Apple ROMs to be a standard ASCII type keyboard.

The AppleFPGA supports all video modes that an Apple][e supports, including text screens, high resolution screens and, dual high resolution screen. The video is supplied to a standard VGA connector for any VGA compatible monitor that supports 640 x 480 resolution at 60 frames a second. The video includes a real time clock displayed below the standard Apple video. This time clock is compatible with the standard ProDOS clock driver.

2. 6502 Processor

The code includes a 6502 “core” written in the HDL VHDL. This code is copyrighted by Daniel Wallner. It is cycle compatible with the standard NMOS 6502. Please inspect the VHDL source code for additional details and restrictions.

The standard NMOS CPU could run at 1 - 2 MHz. This implementation is capable of running at almost 16.67 MHz. The speed is derived from a 50 MHz oscillator and is divided by 3. At this highest speed, the CPU loses cycles when ever the video accesses memory. The effective CPU speed is 16.07 MHz. The CPU speed is switch settable. In addition to this high speed, a nearly Apple compatible 1.042 MHz and a faster 2.083 MHz are also available. An additional switch setting halts the CPU clock to stop the system. The system can be resumed by setting the switches to another position. The Digilent board switches 0 and 1 are used to switch the CPU speed. The switch positions are listed in section 9.

3. Memory

The Digilent board includes 1Meg of high speed SRAM implemented as 2 256k x 16 chips. 64kx8 of one chip is used as Apples main memory including the additional 16K of Language Card memory. The additional 448K is setup to be compatible with Applied Engineering RAMWORKS III card plugged into the AUX slot. The RAMWORKS III supports 4 groups of 8 memory chips. Each group can

be either 256K x 1 or 64K x 1. This implementation appears to be one group of 256K x 1 and 3 groups of 64K x 1.

4. Floppy Disk Drives

The additional 512Meg of SRAM is used to emulate two floppy disk drives attached to the standard Apple floppy disk controller. These floppies are compatible with all the Apple software that I have tried. They can be formatted with Filer or Copy II+ Version 7.4. You can use the Disk copy feature in Copy II+ to write images into the floppies. I have successfully booted from the floppy after writing DOS 3.3, Castel Wolfenstein, Zaxxon, and ProDOS images. The timing of the floppies is based on the CPU clock. This means that by changing the CPU clock to a different speed does not affect the ability to access the floppies. Also, because of this feature, the floppies run much faster when the CPU speed is set to the higher frequencies. At the highest CPU frequency, the floppy is emulating a disk where the rotational speed is 16.07 x 300 RPM = 4821 RPM,.

Each track of the floppy uses 6312 bytes of the SRAM. This gives the floppy the capabilities to use over 41 tracks. AppleFPGA implements a full 40 tracks on each disk. The ability to use half track is also supported. But just like the standard Apple Floppy, when you use half tracks, the next track needs to be at least one full track away or data will be over written. The reasoning is different, but the results are the same.

The one issue with the floppies is the incompatibility with the bit image files available on the Internet. These images incorrectly use 6656 bytes for each track. With a disk rotating at 300 RPM the Apple writing a byte every 32 CPU cycles, the maximum number of bytes would be approximately 6400 bytes. Self sync bytes would lower this value even more. The AppleFPGA floppy is a byte oriented interface and compensates but does not emulate self sync bytes. Because of this, the total number of bytes for each track is set to 6312. Disk drive alignment programs will report that the disk is running a little fast, this is to be expected, since the self sync bytes do not take any additional time.

There is a switch, SW4, which can be used to swap the floppies. With the switch in the OFF position, floppy 1 is Drive 1 and floppy 2 is Drive 2. When the switch is set to the ON position, the floppies are reversed. This switch is useful when a program required two floppies. When you boot from the first floppy and the program asks you to replace the floppy in drive 1 with the second floppy, flip this switch and the data in floppy 2 will appear to be in drive 1. There are two switches that are used to write protect the floppy, SW2 and SW3. The switches stay with the floppy and not the drive. If a floppy is write protected, swapping the drive numbers by using switch 4, does not affect the write protection of the floppy. There are two LEDs that show floppy activity, LED 6 and LED 7. Like the write protect switches, the LEDs are locked to the floppy and not the drive.

5. Serial Virtual Drive (SVD)

The main storage mechanism for AppleFPGA is a virtual disk file located on a PC. More information on this can be obtained from Terence J. Boldt's excellent Apple II project web site, apple2.boldt.ca. The PC software to make this work can be downloaded from his site as either an executable or source file. The executable file runs at 19200 Baud. AppleFPGA has been successfully tested at speeds up to 57600 Baud. An executable to run at the 57600 Baud speed will need to be compiled. One compiler available to accomplish this task is the free compiler from Borland. Setting up this compiler is beyond the scope of this document. But after the compiler is setup, the source will need to be modified to run at the faster speed. Line 60 sets the baud rate. Change the CBR_19200 to CBR_57600. Also, lines 4, 5, and 6 will need to be completed. The three include files needed are <windows.h>, <stdio.h>, and <conio.h>. Add one of these file names to each line of the source. After modification, the executable can be built.

The source code for the Apple ROM is included in the distribution file. It is assembled into slot 7 and uses some of the slot 4 ROM space that is not being used by the clock ROM. Blank drive images are also included with the distribution. Files can be added to the images by one of several methods. One method is to use one of the available Apple II emulators that support HDV files. Another method is to use A2 Oasis Disk Image manager. Several disk images can be created. Just like a floppy, a disk can be “changed” after the system is booted. To change a disk, the PC program should be terminated, then restarted with the desired image. ProDOS will pick up the new image. Also, just like the floppy, be sure that the ProDOS does not have files open when the image is changed.

6. Clock

AppleFPGA supports a ProDOS compatible clock in slot 4. The clock keeps track of the seconds, minutes, hours, days, and day of the week. The clock is not too sophisticated. It assumes all months have 31 days. It will increment the day at midnight, but the month and year are never incremented. This means that on October 31 at midnight, the clock will become October 1. The ROM source code for the clock is included in the distribution included in the file for the SVD. This clock only supports ProDOS mode. It either redirection command is given, IN#4 or PR#4, the clock ROM will not respond. At each entry point is nothing but a return from subroutine.

The clock uses all 16 IO bytes that are allocated to slot 4. Each byte, except for one, holds a clock data byte that is ASCII encoded.

C0C0	Year Thousands (hard coded to 2)
C0C1	Year Hundreds (hard coded to 0)
C0C2	Year Tens

C0C3	Year Ones
C0C4	Months Tens
C0C5	Months Ones
C0C6	Day of Week (0-6)
C0C7	Days Tens
C0C8	Days Ones
C0C9	Hours Tens (0-2)
C0CA	Hours Ones
C0CB	Minutes Tens
C0CC	Minutes Ones
C0CD	Seconds Tens
C0CE	Seconds Ones
C0CF	See text

To set the clock write the appropriate values into bytes C0C2 to C0CE, then write a 128 into C0C0 or C0C1 then write a 0 into C0C0 or C0C1. When a 128 is written into either of these two bytes, the clock will latch the values that have been written into the other locations. It also, will stop the clock from running. By writing a 0, the clock will restart with the new values. A program to set the clock, SETTIME.BAS, is included in both the dual 16 Meg SVD image and the single 32 Meg SVD image. This program will set the time for this clock and will not work with any other type of clock.

The clock will retain it time through a Reset, but it will not retain the time is the FPGA is reloaded. Also, the clock is not battery backed up, it will not retain the time when the power is removed. The byte at C0CF holds a counter, 0 to 255, that increments every 63.84 uS. This could be used as a seed for a random number generator

7. Joystick / Paddles

The schematic for the Joystick interface is included in the distribution. The Joystick does not work exactly like the original Apple interface. The original Joystick interface depended on the charging of a capacitor for timing the position of the device. Since AppleFPGA runs at 3 different speeds, the charging of the cap would limit the speed to the original 1 MHz. AppleFPGA replaces the charging of the cap with a digital to analog converter. This converter changes output depending on the speed of the processor. This gives the Joystick position CPU speed independence. Because of the way this interface is implemented, the Joystick is not very linear. Some additional work will need to be done to try to make this interface more linear. This will appear in a future revision. But for the present time, some “calibration” of the two parameters, OFFSET and RANGE, that control the position conversion can be adjusted.

If the Joystick interface is not needed, the additional circuit does not need to be built. AppleFPGA will run correctly without this interface connected.

8. Speaker

An additional schematic for the speaker circuit is also included in the distribution. As with the Joystick, this circuit is not needed unless speaker support is desired. This circuit was designed to power a small speaker. There is a volume control to adjust the speaker level. Be careful in adjusting the volume to the higher levels, there is a possibility to burn out the small transistor.

Toggling of the speaker is done by reading from IO location C060. Writing to this location is not defined. AppleFPGA uses writes to this location to accomplish two different tasks. The first task is to write enable the ROMs, by reading from this location, the ROMs will become write protected. But by writing to this location, the ROM will become write enabled. Also, the value written to this location is important. The value written will set the color of the text display. The text screen color can be changed to one of eight different colors.

Value	Color
0	Black
1	Red
2	Green (Default)
3	Yellow
4	Blue
5	Purple
6	Aqua
7	White

Make sure you read from this location after setting the text color to write protect the ROMs. By hitting the Reset button, the color will revert back to the default color. If you want a different difficult color, edit the applefpga.v file to change the reset value of user_c.

9. Switches, Buttons, and LEDs

There are eight slide switches on the Spartan 3 starter board. Six of these switches are used and two are not used. Here is a list of the function of each switch.

Switch	Switch OFF	Switch ON
SW0	*	*
SW1	*	*
SW2	Floppy 1 write enabled	Floppy 1 write protected
SW3	Floppy 2	Floppy 2

	write enabled	write protected
SW4	Floppy normal	Floppy swapped
SW5	Serial Port 57600	Serial Port 19200
SW6	Not Used	
SW7	Not Used	

*

SW0	SW1	CPU Speed
OFF	OFF	1.042 MHz
ON	OFF	2.083 MHz
OFF	ON	16.07 MHz
ON	ON	STOP

There are 4 push buttons on the board. Only one button is used, BTN 4. When this button is pushed, the CPU will be reset.

There are 8 LEDs on the board. All of these LEDs are used. Here is a list of their functions.

LED	Function
7	Slot 7 active
6	Floppy 2 being accessed
5	Floppy 1 being accessed
4	Apple ROMs write enabled
3	Floppy controller PH_3
2	Floppy controller PH_2
1	Floppy controller PH_1
0	Floppy controller PH_0

The 4 seven segment displays will permanently display A][E.

10. 6502 Source Code

As mentioned previously, the ROM source for the SVD and the Clock is included in the distribution in the 6502Source directory. Both ROMs are included in the same file along with a boot up ROM designed to load the Apple ROMs from the Serial Virtual Drive over the serial link. There are two versions of the file. There are two different versions. One version gives one 32Meg drive. The other splits this file into two 16Meg drives. Be sure to use the correct drive image with the ROM version that you choose.

The source code can be modified with any text editor and re-assembled using a 6502 Assembler. The code was written specifically for the TASM 3.0.1 assembler by Speech Technology Incorporated.

11. Getting Started

To get AppleFPGA running requires some amount of work. This document assumes that the Xilinx WebPACK has been downloaded from the Xilinx website, installed, and is working. This project is setup using version 7.1.04i, but any later version should be able to load the project files.

Unzip the AppleFPGA file into a folder somewhere on your hard drive. Before a working version is compiled and loaded on the Starter board, a couple of decisions need to be made. The first involves how you want to load the Apple ROMs. Most people will not want to load them over the serial port. The other option is to compile them into the project so the FPGA gets loaded with the ROMs automatically. A utility has been included in the distribution that will convert ROM files into the format needed for compiling in Verilog. This program will run in a pure DOS environment or in a DOS box under any version of Windows. To use the program, enter the command:

toverilog input output *start **length

*start should be entered with 4 hex digits
**length is optional and is entered in hex

The input file is the binary ROM file that you want to convert. The output file is the file that you want to write the Verilog code into. This file does not need to exist. If it does not exist, it will be created. If it exists, it will be overwritten and all existing data will be lost.

The start parameter is the starting position (in hex) where you want to start the conversion. The length parameter is optional, and provides a way to convert shorter ROM files. If the length is not given, the default value of 0800 (2048 bytes) is used. This is the maximum that the Verilog files can accept.

Place the Un-enhanced Apple IIe ROM image and the floppy ROM image into the utility directory. This document assumes that the Apple IIe ROM file is 16K long and that the first 256 bytes are blank. The remaining 14K is the ROM that is addressed from C000 to FFFF and holds the internal Apple IIe ROM. To convert this ROM into the needed files, give these commands.

```
toverilog app2e.rom c0i.v 0000
toverilog app2e.rom c8.v 0800
toverilog app2e.rom d0.v 1000
toverilog app2e.rom d8.v 1800
toverilog app2e.rom e0.v 2000
```

```
toverilog app2e.rom e8.v 2800
toverilog app2e.rom f0.v 3000
toverilog app2e.rom f8.v 3800
toverilog disk.rom c6.v 0000
```

You can open these files using the editor in WebPACK. Copy the contents of each file and replace the default lines in the rom_xx.v files included with the distribution. The disk ROM will replace the C600 section in the rom_c0s.v file. Make sure you update the INIT__xx numbers. The clock ROM and both versions of the SVD ROM are already included in this same file. If you want to use the dual 16 Meg drive, the file is ready. If you want to use the single 32 Meg drive, remove the double slashes from in front of the lines and place double slashes in front of the dual 16 Meg lines. Only one of them can be enabled at a time. The ROM space in C100-C2FF and C500-C5FF is not used. You can place any code in these areas that you want or you can leave them blank. Remember, C000-C0FF is used as IO address and C300-C3FF cannot be used on an Apple IIe.

After all the ROMs have been updated, the project can be recompiled and the board reprogrammed. AppleFPGA should boot and begin looking for the SVD.

12. The Future

I am considering AppleFPGA an unfinished project even though everything is working. I am sure there are some bugs, because I have not tested every program that has been written. Some one, some where has a program that does something undocumented that will break AppleFPGA. Please report bugs to

applefpga@yahoogroups.com

Please give all the information that you can in the bug report. Since everything that I want to do is working, I do not have much ambition to chase bugs. The more information that you give, the more likely I will be able to fix the bug.

Use the same email address to send upgrade suggestions. Since the FPGA is nearly 100% utilized, not many new features can be incorporated. But one feature I would like to add is IDE/Compact Flash support. There are a couple of people that have added this to Apple II systems. But I do not like the way the firmware is being done on either system. But I do not know anything better. Also, I am not having much luck with building the add on interfaces. I am trying to figure out something better.

Please send email when you get this working. I would specifically like to know if people are building the joystick interface, or they would prefer something different. I have some code that would replace the joystick interface with 4 buttons but the Apple would still think that it was a joystick. Since my Joystick is

so bad, I might try to adapt some other type of device as a replacement. Here is a web page that I am using to do get some ideas.

http://www.epanorama.net/documents/joystick/pc_circuits.html