ZX Fusion: A ZX Spectrum Implementation on an FPGA With Modern Peripherals

Gustavo Jacinto and Rui Policarpo Duarte *

Posted Date: 15 November 2023

doi: 10.20944/preprints202311.0956.v1

Keywords: embedded system; FPGA; Z80; ZX Spectrum+

Article

ZX Fusion: A ZX Spectrum Implementation on an FPGA with Modern Peripherals

Gustavo Jacinto and Rui Policarpo Duarte *

Instituto de Engenharia de Sistemas e Computadores Investigação e Desenvolvimento em Lisboa, Lisbon, Portugal; a46006@alunos.isel.pt

* Correspondence: rui.duarte@tecnico.ulisboa.pt

Abstract: The ZX Spectrum was a popular 8-bit home computer by Sinclair Research in the 80s. Even though some of these computers may still work, the audio tapes, the TV with an analog tuner and the micro-switch joystick, used with the original ZX Spectrum, nowadays are outdated and hard to find in good working order or replicate. Since many other old closed systems are also very difficult to update to support modern peripherals, there is a necessity to provide a methodology to adapt such systems to support new peripherals while being compatible with existing software. The work proposed in this paper is focused on recreating a ZX Spectrum+/48k computer and interface it with modern peripherals on an FPGA. This is accomplished by adding a co-processor to assist with the control of the new peripherals which would either require to complex architectural changes to the original system and in the end it would perform poorly due to the low performance of the Z80 CPU. This work distances from previous ones on emulating a ZX Spectrum since it focuses on the use of different upgraded peripherals and the use of a NIOS II soft-processor as a co-processor to manage the SD Card accesses. A demonstration of the proposed modernized architecture was made by successfully running a diagnostics ROM and playing original ZX Spectrum games from an SD card for game with a PS/2 keyboard and a pair of joysticks.

Keywords: embedded system; FPGA; Z80; ZX Spectrum+

1. Introduction

When the ZX Spectrum was first released in 1982, it was considered a breakthrough in personal computing due to its the use of commonly available analog TVs and audio cassettes, and at a reasonable price for the average consumer. The capabilities of modern computers and game consoles have long surpassed the ZX Spectrum but there is a generalized interest in reviving some of its games. The main limitation on being able to play ZX Spectrum games is that it requires, besides a computer still in working order, an audio cassette player, a TV set with an analog RF tuner and an original micro-switch joystick. Moreover, the loading of a game requires an audio tape playing between 5-10 minutes which nowadays may be too much to ask to wait for.

To avoid the difficulty of having all the hardware of that era in perfect working order, many ZX Spectrum emulators have been done for different platforms, either in software and hardware. Despite that, software emulators tend to have behaviors or glitches, that do not exist when running a game on the original ZX Spectrum, caused by the underlying system. Moreover, many of these implementations do not faithfully replicate the operation of the architecture of the original ZX Spectrum. To counter that, this paper proposes a methodology to upgrade such old systems without having to re-engineer the complete system.

The main reason for investigating hardware implementations rather than software ones, was due to achieve the maximum fidelity with the original ZX Spectrum. Moreover, many hardware implementations of the old computer were developed by retro gaming communities in ways that are nearly impossible to reproduce and imply the assembly of a custom Printed Circuit Boards (PCBs) that may not be available. Field-Programmable Gate Array (FPGA) technology was chosen due to its easy reconfigurability and testability, as well as integration and customization, despite allowing a complete

 doi:10.20944/preprints202311.0956.v1

system to be programmable within it. The use of an FPGA development kit allows a hardware design without the need to design and assemble a custom and complex PCB. Research on similar previous works was conducted to reuse any existing validated IP cores, i.e. Z80 Central Processing Unit (CPU).

The objective of this work is to recreate a ZX Spectrum+ on a modern hardware platform and be able to load and play original games as a demonstration of the developed system. This paper details the recreation of a ZX Spectrum+ computer on a modern hardware platform capable of running software exactly as in the original computer while using modern peripherals, such as a Video Graphics Array (VGA) monitor, an SD card for faster game loading and saving, and available retro joysticks. The target hardware platform is DE2-115 FPGA board from Terasic which includes the connectors and interfaces for the new peripherals. The proposed system supports the following peripherals:

- **VGA** video output, used by common computer monitors;
- **PS/2** keyboard input and the original **ZX Spectrum's keyboard**;
- **SD card** for faster loading programs and games;
- **Audio input** for loading audio cassette tapes, allowing old games to be played;
- **Audio output** for playing sounds and music;
- **Kempston and Sinclair joysticks** inputs.

This paper is organized as follows: section 2 presents the background on the ZX Spectrum computer architecture and previous works on its recreation attempts. Section 3 is dedicated to present the new reconfigurable architecture which has an original ZX Spectrum and the new I/O peripherals. Section 4 presents the results obtained from the implementation of the proposed architecture. Results are discussed on section 5, and conclusions are drawn on section 6.

2. Background

2.1. The ZX Spectrum+/48k

The ZX Spectrum is an 8-bit home computer that was developed by Sinclair and released in 1982. One of the first models released was the 48k, which had 48KB of Random-Access Memory (RAM) and 16KB of Read-Only Memory (ROM). The ZX Spectrum 48k is comprised of a Z80 Zilog CPU, an Uncommited Logic Array (ULA) for interaction with memory modules and peripherals such as the beeper for sound output, the keyboard, and the video and RF modulators for video output to analog televisions.

The Z80 is a processor with 16 bits of addressing space and 8-bit data bus, has 3 modes of maskable interrupts, a non-maskable interrupt, separate control signals for memory, and a separate I/O address space and support for Direct Memory Access (DMA) requests.

The ULA is a semiconductor device from Ferranti [1] that was used to design and produce custom logic chips. It was fabricated with various generic building blocks on the bottom 4 metal layers, connected with a top metal layer designed by the customer to support a custom logic circuit. The ULAs were cheaper than soldering several components on a PCB, thus lowering the cost of the ZX Spectrum. The ULA is responsible for keyboard scanning, reading the data from video memory and sending it to the video modulator, and serving as a peripheral for the Z80 to have access to the audio I/O, as well as control over the border color on the screen. It also generates the clock for the Z80 at 3.5MHz. The ULA controls the Z80 clock to handle contention between the two (video memory accesses). The video takes priority, so if the Z80 and the ULA are both accessing the lower RAM, where the video is also stored (0x4000 to 0x7FFF), the ULA stops the Z80's clock before control signals are set. [2] The ULA receives a 14MHz clock and divides it internally down to 7MHz.

The ZX Spectrum's active screen has a resolution of 256x192 pixels, or 32x24 attribute blocks. Each attribute block is equivalent to a square of 8x8 pixels. These blocks can have two colors each, designated "ink" and "paper", for foreground and background colors, respectively. Color data stored in the RAM specifies the color of the INK, PAPER, and brightness of each block, as well as if they are

 doi:10.20944/preprints202311.0956.v1

flashing (switching paper and ink colors). Each byte in video memory corresponds to 8 pixels, where 1 pixel corresponds to 1 bit in that memory position. Its value determines if the color rendered is INK or PAPER. Every group of 32 bytes corresponds to a full row of pixels from left to right on the active screen. The video memory is split into 3 groups of 8 attribute block rows, resulting in the 24-block height of the screen.[3]

The keyboard on the ZX Spectrum consists of a matrix with a series of half-rows that can be selected with the top byte of the address bus. When the Z80 reads the ULA's port at 0x7FFE-0xFEFE, it sets the top byte to select a half-row and the data received corresponds to the state of the keys in that half-row, active-low [4].

2.2. Field-Programmable Gate Array

An FPGA is an integrated circuit that has the ability to reconfigure the hardware inside it to create a custom logic system. Similarly to ULAs, they are composed of programmable building blocks (logic elements) and interconnect used to create complex digital circuits. Unlike ULAs, they are re-programmable, making them easier to test designs on. They are also more complex than ULAs and can hold more logic elements, as well as memory units and PLLs [5]. FPGAs can be programmed in Hardware Description Languages such as VHDL or Verilog.

In this work a Cyclone IV-E FPGA EP4CE115 from Intel fabricated in 60 nm process, and was used on an DE2-115 board from Terasic. This FPGA has 114000 Logic Elements, 266 DSP Blocks, 3.888Mb of M9K (8192 bit) RAM blocks and 4 PLLs. Each Logic Element is composed of a 4-input Look-Up Table (LUT) connected to a carry-chain block followed by a 1-bit register.

2.3. Related Work

In the scope of the work proposed in this paper, the most relevant ZX Spectrum implementations found for FPGA were analyzed to determine if any IP blocks or what unique features could be reused. Some of the implementations investigated include:

- **ZX Spectrum 48K with ULAPlus by Andy Karpov** is an incomplete public "sandbox" project the author made to test;
- **Speccy2010 by Syd et al.** [6], an implementation for an Intel Cyclone II which had an inaccurate contention handler, that could work both ways (ULA stops the Z80 or the Z80 stops the ULA). It has an SD Card slot for tape file support and uses a microcontroller to implement it;
- **ZX Spectrum on FPGA by Mike Stirling** [7], an implementation made on a DE1 board from Terasic (Cyclone II FPGA). It supports SD Cards with tape files by implementing an interface called ZXMMC+ and using ResiDOS with it. This project has timing problems due to the contention between the ULA and the CPU having been fixed;
- **ZX-UNO by Superfo et al.** [8] was made for the Xilinx Spartan-6 FPGA, consists of a collection of implementations for different 8-bit computers, including the ZX Spectrum. For the SD card reading, it implements the DIVMMC interface, similar to the previously mentioned ZXMMC+. The accuracy of this implementation seemed adequate to the original system, in terms of clock speeds and contention;
- **A-Z80 by Goran Devic** [9] is an implementation of the Z80 on FPGA, containing an example project with a working ZX Spectrum implementation made for an Altera DE1. This one does not support SD Cards but supports audio input for loading software. It has some US keyboard keys mapped to key combinations in the ZX Spectrum to allow easier typing of symbols such as "-" and "_", which originally required SYMB SHIFT together with the keys "J" or "0" respectively;
- **ULA by Miguel Angel Rodriguez Jodar** [10] consists of an implementation of the ULA in a Xilinx Spartan-3 Starter Kit. It follows the details described in [2], so it is reconstructed faithfully. It includes many ZX Spectrum implementations with a few differences between each, differing in ULA type or peripherals;

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

- **ZX Spectrum Next by Jim Bagley et al.** [11] is an implementation of the ZX Spectrum for a Xilinx Atrix-7 FPGA. It is the most complete project here and has multiple enhancements to the base ZX Spectrum and options to change them. It allows audio tapes and SD Cards to load programs. Unlike the previous implementations, this one can load snapshot files through the SD Card. It has a full custom case and circuit board made for it to resemble a modern ZX Spectrum. It also allows HDMI video output;
- **ReVerSe-u16** [12] is a collection of implementations of retro computers, including the ZX Spectrum. It uses a Cyclone IV FPGA like the target platform. The board only contains an HDMI video output but the code shows a VGA implementation. It contains different implementations of the ZX Spectrum based on the Z80 implementations used. The ZX Spectrum 48k implementation that uses T80 has it running at 50MHz, which is inaccurate regarding the original ZX Spectrum. A mini-SD card slot is on the board, but no mention of it was found in the code.

3. Modernized Architecture

The proposed architecture to modernize outdated computing systems like the ZX Spectrum+ relies on a co-processor attached to the address/data buses to issue memory access requests and occasionally control the Z80 CPU.

The target FPGA is a Intel Cyclone IV. In this work, a NIOS II was used as a bridge between the ZX Spectrum and the peripherals that it could not support with its original architecture, such as loading and saving files from/to an SD Card. To load and save of programs, the software on the NIOS II supports .Z80 and .SNA files stored on an SD card with FAT32 or FAT16. The NIOS II processor interprets and decodes the files before writing its contents directly to memory. Besides the SD card access, the NIOS II is also responsible for the contents of the new menu for loading files.

3.1. High-Level System Architecture

A diagram of the general architecture of this implementation is illustrated in Figure 1. The blue rectangles are used to represent I/O interfaces and the green ones are custom IP Cores that did not exist in the original the ZX Spectrum+.

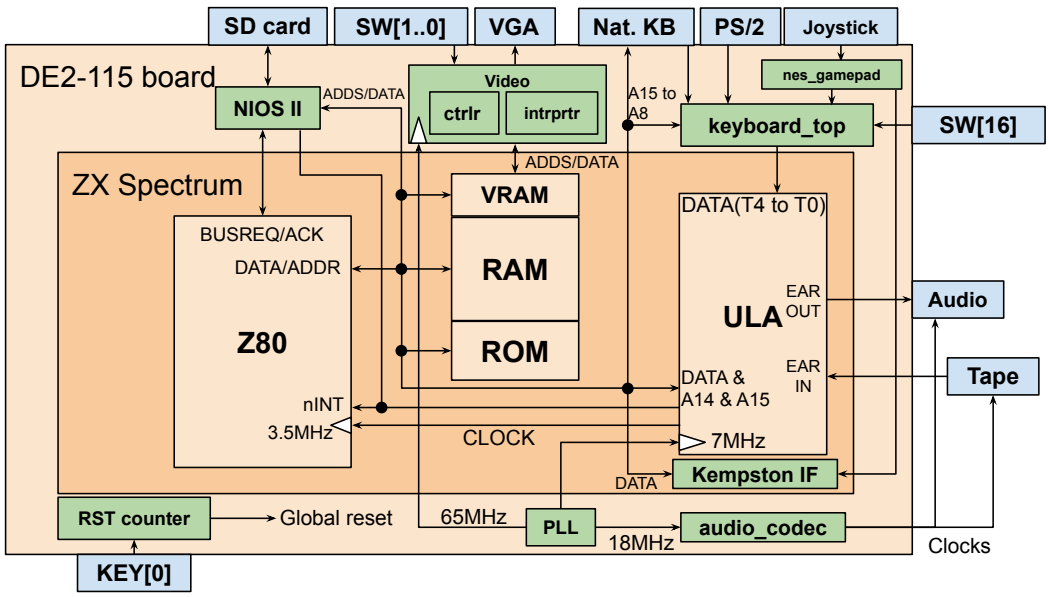

Figure 1. A block diagram of the whole project as it was implemented.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

3.2. Z80 CPU

As aforementioned, the Z80 is the heart of the ZX Spectrum+. This work uses the implementation from T80 [13]. It was chosen for being extensively tested and used in several projects. To test the T80 CPU a minimal system was created. It included a ROM and three RAM modules (Screen, Color, and Data). The Video RAM (VRAM) (Screen and Color RAM) is implemented with dual-port RAMs, to have the CPU and the VGA controller accessing the memories simultaneously.

3.3. ULA

The ULA in the riginal ZX Spectrum was responsible for reading the ideo data and send it to the video modulator. Since it is no longer needed, it was reduced to two IP blocks:

- **ULA Ports** based on [7] and acts as the peripheral of the Z80, providing access to the keyboard and audio and keeping track of the border color;
- **ULA counter** new video counters (horizontal, vertical, and flash) with outputs that can trigger the T80's interrupt, generate the T80's clock, stop it for contention handling, and output the flash clock for the video component.

3.4. Video Output

The video output resolution is 1024x768 which is supported by most modern monitors. It is also a multiple of 4 of the original ZX Spectrum active screen resolution (256x192). The video controller is also capable of producing 1x and 2x scaling factors, making the visible area smaller and the border larger. A VGA signal of 1024x768 at 60Hz requires a pixel clock of 65MHz, which is greater than the ULA's internal clock of 7MHz. With this resolution the border is only 1 pixel thick at the edge of the screen. If the user prefers having a larger border, two other video modes can be set with the switches SW[0] and SW[1] on the board. The architecture of the video component is illustrated in Figure 2.

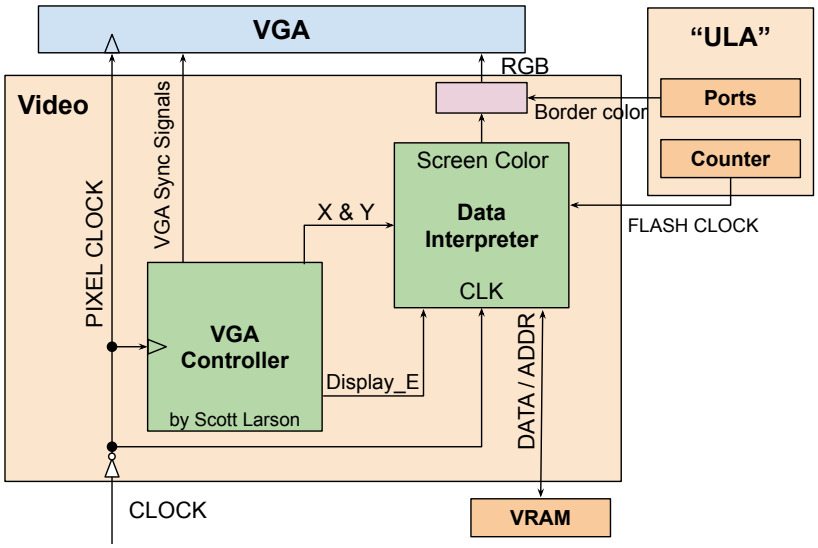

Figure 2. A block diagram of the video component.

The VGA Controller from [14] receives the video clock and generates the VGA's sync signals as well as the current row and column that is being drawn. The Data Interpreter receives these row and column numbers and calculates the addresses of the current screen and color info being rendered. The VRAM is driven by a clock with the opposite polarity of the video clock, so that new video data is fetched when the current data is being displayed.

The VRAM memory addresses are computed from the row and column values, as seen in Figure 3. The computations are in Equations 1 and 2:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

$$PIXEL\_ADDR <= group\_num \,\&\, pix\_row\_num \,\&\, att\_row\_num \,\&\, att\_col\_num \tag{1}$$

and

$$COLOR\_ADDR <= group\_num \,\&\, att\_row\_num \,\&\, att\_col\_num \tag{2}$$

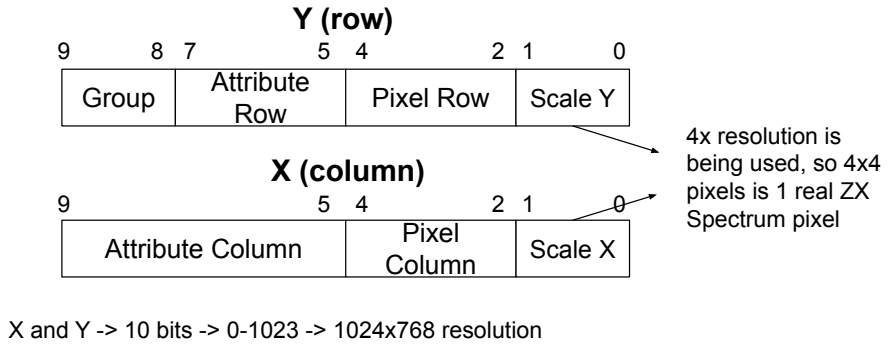

Figure 3. The address are computed from the row and column numbers.

3.5. PS/2 and Original ZX Spectrum+ Keyboards

Since original ZX Spectrum + keyboards are becoming scarce, and the DE2-115 FPGA board has a PS/2 connector, it was used to connect a keyboard to the system. The keyboard IP Block consists of a PS/2 Controller from[15] and what was called an "Input Receiver". The PS/2 Controller supports bidirectional communication with the keyboard. It receives key codes and sends LED state updates. The "Input Receiver" receives the top byte of the address bus and key codes of keys that are pressed or released and, according to them, output data that correspond to those key presses depending on the half-rows selected by the address bits. The state of the shift and alt keys are considered for shortcuts in US and PT keyboards, and trigger combinations of CAPS SHIFT and SYMB SHIFT to write the specific symbols.

A native ZX Spectrum+ keyboard can be selected by activating switching SW[16] on in the target platform. This connects the top address byte and the 5-bit data bus directly to the keyboard. If the Sinclair mode is on for the joysticks (SW[17] off), the controller's state is deserialized in the IP block "nes_gamepad" and interpreted as the half-rows they correspond to, 1-5 or 6-0, since Sinclair joysticks would map to these keys.

3.6. Audio I/O

The audio input and output is supported by a WM8731 codec from Wolfson on the target platform. The codec was configured for a reference clock of 18.432MHz (same value as in the datasheet), a sample rate of 48KHz, a 1 bit datawidth and CHANNEL_NUM at 2, for stereo audio. The calculation is as follows:

$$\frac{REF\_CLK}{SAMPLE\_RATE \times DATA\_WIDTH \times CHANNEL\_NUM \times 2} - 1. \tag{3}$$

It should be noted that the audio output of the ZX Spectrum is done with only 1 bit, so the bit clock would be appropriate for this. LRCK uses only the sample rate and the reference clock in its calculation, in a similar way. The calculation is as follows:

$$\frac{REF\_CLK}{SAMPLE\_RATE \times 2} - 1. \tag{4}$$

These calculations, with the aforementioned constants, result in 192000-1 and 96000-1 for BCLK and LRCK respectively. The stereo clock is half of the bit clock since the bit clock was calculated for a data width of 1, meaning the stereo clock must change for every period of the bit clock.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

The audio output of the ZX Spectrum is sent as data directly to the CODEC and the generated clocks are able to transmit it successfully.

Audio input however required transformation due to the CODEC being 24-bit. This meant having to translate the 24-bit value into a 1-bit value for the ZX Spectrum input. A component was made called "audio_adc" which contains a 4-bit counter and a 24-bit shift register that receives the data sent by the CODEC. To facilitate the conversion, only the MSB is sent to the EAR input of the ULA, so when the 24th bit is received, the MSB changes, and the ULA receives it. This means that values 0x000000 - 0x7FFFFF are interpreted as 0, whilst values 0x800000 - 0xFFFFFF are interpreted as 1.

3.7. Joysticks

The joystick is the prefarable controller to play games. In the 80s there were two types of joysticks: Kempston and Sinclair. Kempston joysticks had the switches (N,S,E,W,Fire) connected to an input port decoded at address 0x001F while Sinclair joysticks were connected to the numeric keys on the keyboard. Since finding original Kempston joysticks is very difficult, NES clones gamepads were used in this implementation. These gamepads send the state of the controller's buttons in serial synchronous interface. A component called "nes_gamepad" deserializes the data and sends it to either the keyboard IP block or the Kempston interface IP block, depending on SW[17].

Figure 4. Modern NES clone gamepad.

3.8. NIOS-II Co-Processor

The software running on the NIOS II processor is responsible for the support of the new features introduced in the system. It was developed in C and can be divided into three parts: SD interaction (SPI), Z80 Control, and File Access (FAT).

The SD interfacing was based on the DE2-115 SystemCD's SD card demonstration[16] with an added interface for NIOS II application to use. The interface allows listing the included files. A FAT library was used to reading and writting them.

The Z80 Control consists of two separate interfaces: one for **DMA** and one for the T80 to use it as a **perihperal**.

3.8.1. DMA Interface

The DMA interface allows the NIOS II to request memory access to the T80, to perform read/write operations on memory and I/O, and stop the DMA, with or without the Non-Maskable Interrupt (NMI) signal enabled. It also allows waiting, for n attempts, for the value in the CPU's address bus to be

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

higher than the end of screen RAM, 0x5800. The reason for this, and the NMI, is explained further down.

3.8.2. Peripheral Interface

The peripheral interface is used to obtain the last sent command from the T80 and clear the command register. A command register is used, called "nios_per_reg" in VHDL, due to the difference in clock speed between the NIOS and the T80. The register keeps the command until NIOS finishes it, resetting the register. Reads and writes to defined addresses work as commands and not real read/write operations. These commands include:

- a read to 0x17 is INIT, used to request that NIOS load the main menu into memory;
- a read to 0x19 is SD related, used to request a list of 16 filenames of files inside the SD card;
- a write to 0x19 is also SD related, used to request the writing of the selected file's contents into memory and the CPU's registers;

4. Embedded Software

While part of the proposed methodology involves the interconnection of the NIOS II co-processor, it is also necessary to account for the embedded software on both CPUs, the Z80 and the NIOS II.

4.1. Modified ZX Spectrum+ ROM

Even though it was intended to preserve as much as possible of the original system, and that included the ROM contents, it was necessary to introduce new code inside the ZX Spectrum 48k's ROM on an unused region of ROM (0x386E). Moreover, to support the new start menu and the file selection menu it was necessary to write them in RAM as they would not fit the little free ROM. When the system boots up instead of going straight to the basic console, it presents a menu with three choices: load a file from the SD card, go online (future work) or start the basic. The modified ZX Spectrum+/48k ROM is illustrated in Figure 5. Darker colors represent modified code, while brighter colors represent the regions described in the legend.

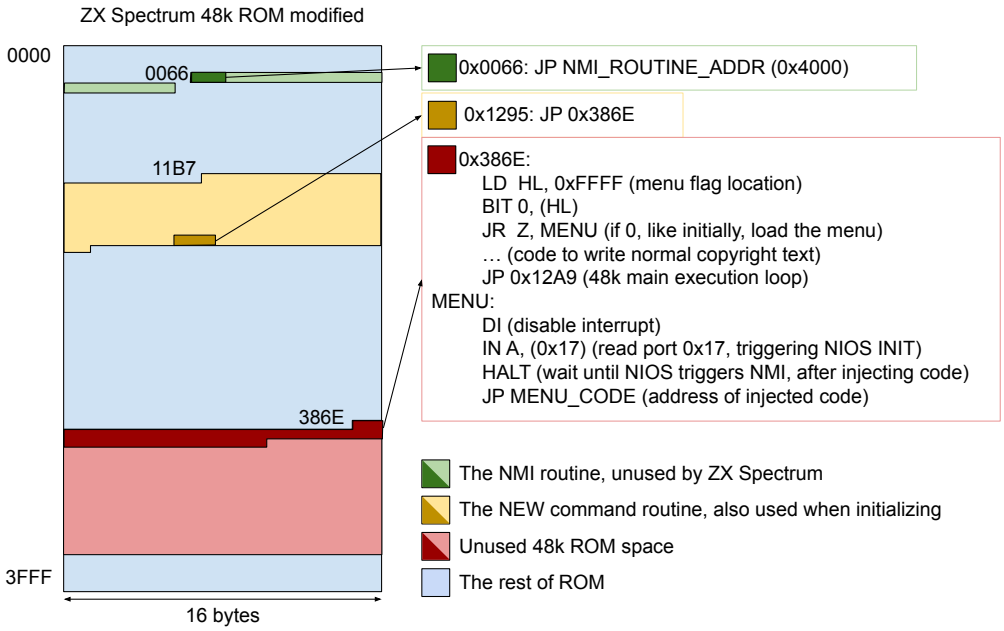

Figure 5. A diagram of the ZX Spectrum 48k ROM's modifications.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

4.2. NIOS II Application

NIOS' main contains a loop that checks the register for any command that has been sent, followed by the execution of that command. A flowchart representing this loop in its complexity is shown in Figure 6. Since the .z80 file format has multiple versions, these have to be considered for interpreting the files, as can be seen in the Figure 6.

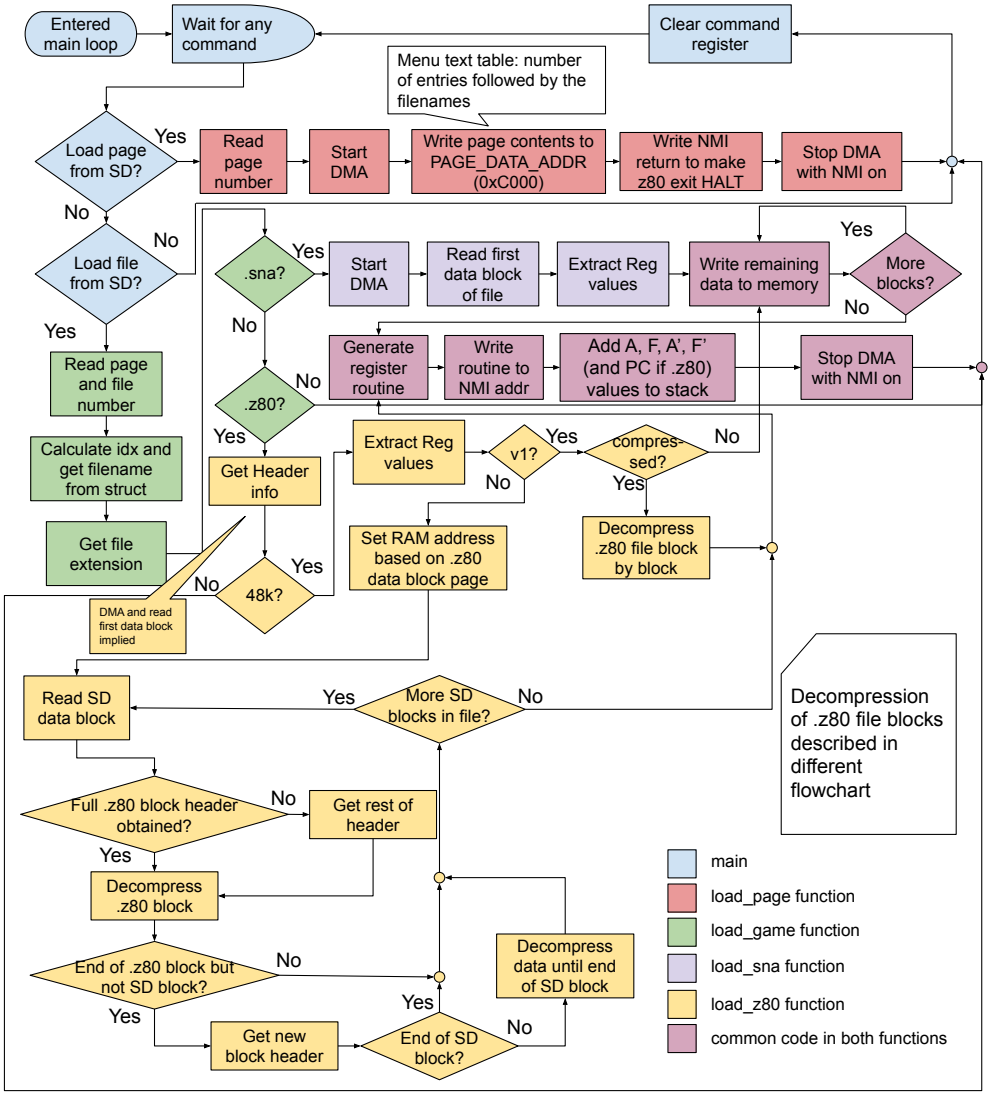

Figure 6. The processes involved in the main program loop of NIOS application.

The new menus were created based on the ZX Spectrum 128K's menus. A disassembly of ROM 0 of the 128K was analyzed and the necessary routines were copied and adapted to generate the menus used. Some routines were changed to receive a parameter to differentiate the main menu from the file menu so that the full code of the menus could be compact, without much repetition. The file menu gets its data from a predetermined address that NIOS writes the filenames to, when the Z80 requests them, as the user selects the "SD Loader" option.

The menus used in the ZX Spectrum's boot are written by NIOS during its initialization. The BASIC option in the menu jumps to the NEW routine in address 0x11B7, provoking a system reset. This makes all of the RAM reset except for values below the address in the system variable RAMTOP (0xFF57). Therefore, the state of the menu flag is saved beyond this address so that the reset does not overwrite it. The flag determines if the ZX Spectrum 48k code continues execution normally (1) or if it

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

requests the menu from NIOS and executes it (0). By default, it is 0, since all of RAM is set to 0 during initialization, making the menu load, but selecting the BASIC option sets it to 1 and resets.

A flowchart of the initialization of the ZX Spectrum Fusion is in Figure 7, where the main menu controls also reside. Interrupts are disabled before the HALT to prevent the ULA from triggering them , used to jump to the keyboard scanning routine. The NMI input is still sampled when interrupts are disabled.

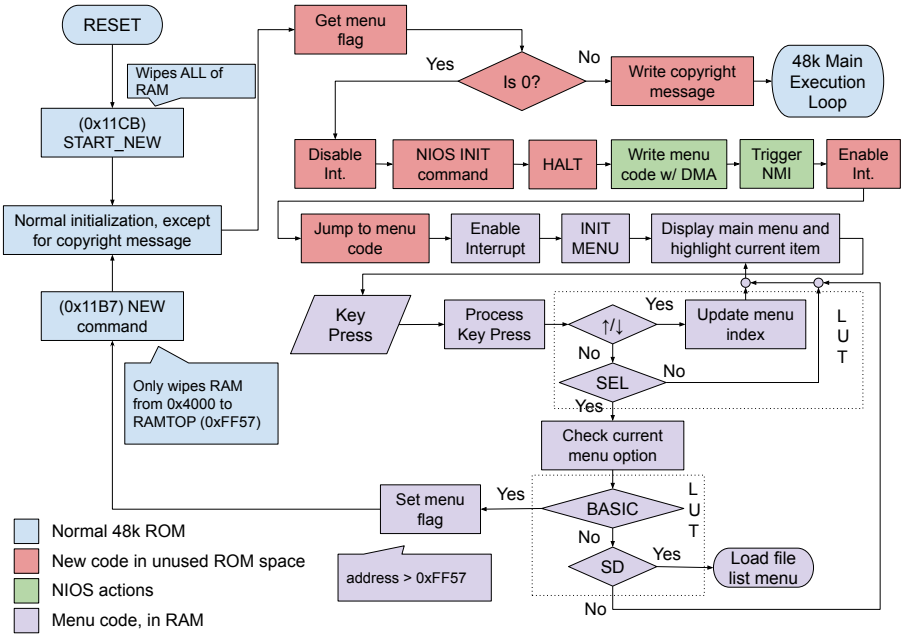

Figure 7. A flowchart of the ZX Spectrum Fusion's initialization.

The file list menu's assembly code flow, along with the process of loading a selected file, is shown in Figure 8.

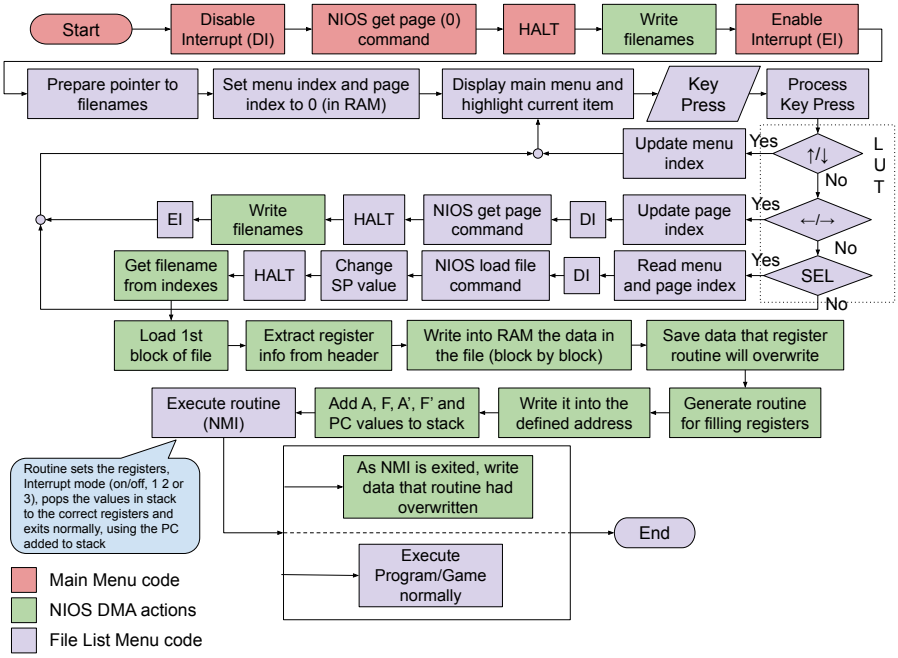

Figure 8. A flowchart of the file menu's processes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

Without NIOS on, the T80 will remain halted forever, waiting for the NMI signal to execute a simple RETN instruction to leave the halted state after the menu is written. Figure 9 shows a flowchart of NIOS' initialization.

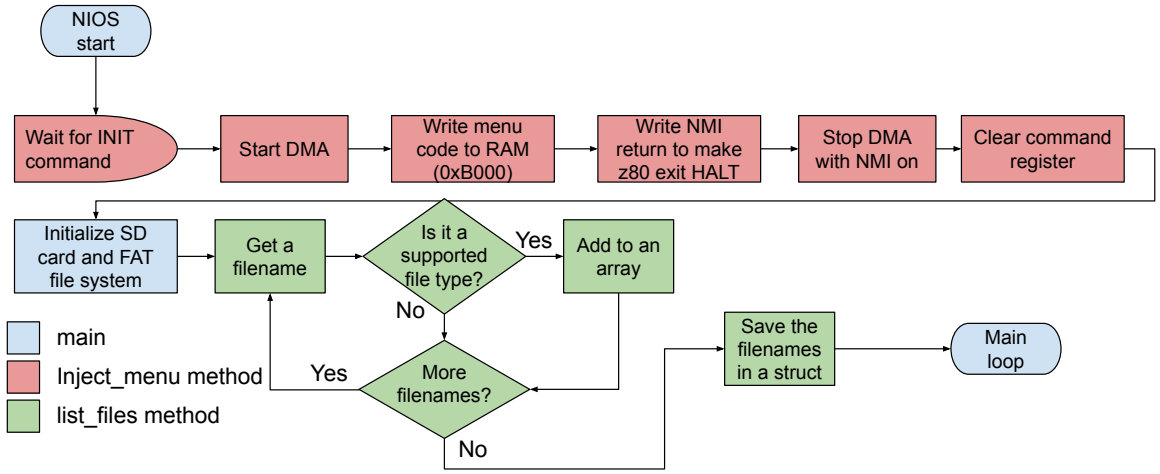

Figure 9. A flowchart of NIOS' initialization.

4.3. File Reader

The file format reader opens the files and, depending on their extension (.z80 or .sna), decrypts them to write their contents into memory and the CPU's registers. The file format reader uses the SD communication part of the software to open and read files. It also uses the DMA interface to write the contents of the software into memory as it reads them from the SD Card, block by block.

When the Z80 CPU sends commands to the NIOS, it enters a HALT state to allow time for NIOS to see the command and act on it. When the contents of a file are loaded to memory, a routine with the register values is generated for the Z80 to execute. This routine is loaded into the screen memory's region to avoid overwriting important program code. Values for the AF and AF' regs are added to the stack, since these lack direct load instructions, but have POP instructions. To avoid problems with the unknown location of the Stack Pointer (SP) before this operation, it is set as the bottom right of the screen region. After these writings, NIOS stops the DMA with the NMI signal on. The unused NMI routine of the original ZX Spectrum, present in address 0x66, was edited to perform a jump to the screen memory's start, address 0x4000. In triggering the NMI, NIOS makes the CPU execute the routine to load all the registers, including the PC to jump to when exiting the routine to execute the program.

The routines that are written in VRAM may cause visual effects in the corners of the screen. NIOS overwrites them with the originally intended VRAM data by using the aforementioned function in the DMA interface. This function reads the address the CPU is accessing to check when it is no longer executing the routine in VRAM. It stays in a loop for n attempts until the condition is satisfied, and then the file format reader restores the video data, preventing the visual effect from staying on screen for more than half a second.

5. Experimental Results

To validate that the proposed methodology works as intended, several tests were made to the new ZX Fusion. It started off with the stand-alone testing of the video component, with a preloaded VRAM. It was observable that the image displayed correctly.

The T80 implementation was tested in a project where the target platform's switches were used to input data and therefore instructions. Machine code of some instructions were loaded manually via the switches. The tested instruction included a data read from memory, an addition (modify), and a write back to memory. The T80 was then added to a project with BRAMs and the video component. At first,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

a small Z80 assembly routine was loaded into ROM that writes an X in the corner of the screen. This worked successfully so an original ZX Spectrum 48k ROM was loaded and the VGA screen showed the starting ZX Spectrum screen with the copyright message below.

The new PS/2 keyboard component was implemented and tested on a stand-alone hardware project, where the CPU address (keyboard half-row multiplexer) was set using the FPGA board switches, and the LEDs on the board associated with the logic levels of the rows selected with that address and also the keys being pressed. The keyboard was working correctly so it was added to the project with the T80 and video component, along with the ULA IP Block. The ULA had to be added since it is responsible for triggering the interrupts that scan the keyboard.

The SD card reading was tested by loading files of different formats, including versions, and of different lengths. Moreover, the menu navigation was tested for the file selection. An SD card with a FAT32 file system and more than 16 files (29) was used to test the navigation since each page can only list 16 files. Every game loaded and ran as expected.

A speaker was connected into the audio output of the target platform to test the implementation of the sound generator. It was compared to online videos of the games being played to confirm their accuracy. Audio input was tested by using an Android phone application called "ZX Tape Player" to play audio tape files. The ZX Spectrum successfully loaded multiple tape files.

The NES gamepads were tested separately in a hardware project titled "nes_gamepad_test". In it, the data being received was displayed in the board's LEDs, where it was verified to be correct. The Sinclair joystick interface was tested on the BASIC prompt since these joysticks map to the numbers of the keyboard. The Kempston joystick interface was tested with a small program written in BASIC. This program reads the 0x1F port of the Kempston interface and outputs a letter (R,L,D,U,F) based on the value being read corresponding to the pressed button (right, left, down, up, and fire).

Additionally, a separate ZX Spectrum Hardware Diagnostics ROM was used to evaluate the implemented system, based on how the test identified it. The project associated with the ROM is called "zx-diagnostics" by Rendanalford on GitHub. [17] The ULA was detected as a SLAM48+ despite being implemented as an Issue 3 ULA. This could be due to the implemented ULA's timings being similar to the detected type of ULA. The test page can be seen in Figure 10. The other tests contained in this screen were successful.

Besides the architectural tests, original games were also tested. Figure 11 shows the ZX Fusion running the game "Daley THompson's Super Test I". It was also observed that the games retained the same *mechanics* as experienced in the original ZX Spectrum.

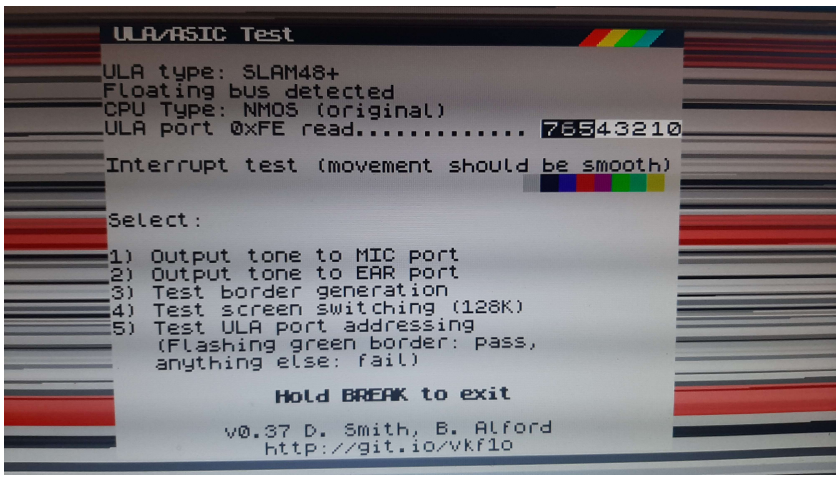

Figure 10. ULA test running result.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 November 2023 doi:10.20944/preprints202311.0956.v1

Figure 11. ZX Fusion running the game Daley Thompson's Super Test.

6. Conclusions and Future Work

This work presented a novel method to interface old computing systems so that they could use modern peripherals while remaining fully compliant with its original applications. To demonstrate its effectiveness an FPGA implementation of a ZX Spectrum+/48k was created. Unlike most studied projects, this project supports snapshot file loading and extra keys on PS/2 keyboards. Other computing systems could laso benefit from this work. For example, the ZX Spectrum 128k could be done by adding memory paging to the design and the menu options of this implementation to the 128k's menu. More features could be implemented such as internet access. The Internet access could allow users to obtain the snapshot files from websites and have them downloaded and loaded directly into the implementation.

Bill of Materials

The proposed ZX Spectrum system was constructed, and can be reproduced, using the following components:

1. DE2-115 FPGA board from Terasic - €423 (academic w/o import tax);
2. IO Expansion PCB for gamepad and ZX Spectrum+ keyboard - €5 (GERBER files on the project's Github repository);
3. 2x NES gamepad with DB9 connector - €9 (e.g. https://www.aliexpress.com/item/1005002264514365.html);
4. 2x IDC10 to DB9 flatcable adapter - €4;
5. 2x IDC10 (2x5 pin) male connectors for PCB - €1;

Data Availability Statement: The complete source code for the modernized ZX Spectrum project is publicly available on GitHub www.disclosed.after.acceptance as an open-source project, under MIT license.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bee, H. Introduction to Gate Array. https://hardwarebee.com/introduction-to-gate-array/, accessed: 2023-08-03.
2. Smith, C. *The ZX Spectrum ULA: How to Design a Microcomputer*; ZX Design Retro Computer, ZX Design and Media, 2010.

3. Beltfield, D. Screen Memory Layout. http://www.breakintoprogram.co.uk/hardware/computers/zx-spectrum/screen-memory-layout, accessed: 2023-08-04.
4. Owen, C. IN and OUT. https://worldofspectrum.org/ZXBasicManual/zxmanchap23.html, 1995, accessed: 2023-09-30.
5. What Is an FPGA? https://www.arm.com/glossary/fpga, accessed: 2023-08-05.
6. Westcott, M. The Speccy2010: A Complete Guide For Non-Russian-Speakers. https://web.archive.org/web/20170706104143/http://matt.west.co.tt/spectrum/speccy2010/, 2011, accessed: 2023-09-06, 2011.
7. Stirling, M. Sinclair ZX Spectrum 48k and 128k on an Altera DE1 FPGA board. https://github.com/mikestir/fpga-spectrum, 2016, accessed: 2023-09-30, 2023.
8. ZX-Uno [ZX Spectrum Computer Clone Based on FPGA]. https://zxuno.speccy.org/maquina_e.shtml, 2013, accessed: 2023-09-06, 2013.
9. Devic, G. A-Z80 CPU. https://opencores.org/projects/a-z80, 2014, accessed: 2023-09-06, 2014.
10. Rodriguez Jodar, M.A. ULA chip for ZX Spectrum. https://opencores.org/projects/zx_ula, 2012, accessed: 2023-09-06, 2012.
11. About - ZX SPECTRUM NEXT. https://www.specnext.com/about/, 2017, accessed: 2023-09-06, 2017.
12. ReVerSE-U16. https://github.com/mvvproject/ReVerSE-U16, 2019, accessed: 2023-09-06, 2019.
13. Wallner, D.M. T80 cpu. https://opencores.org/projects/t80, 2002, accessed: 2023-09-06, 2002.
14. Larson, S. VGA Controller (VHDL). https://forum.digikey.com/t/vga-controller-vhdl/12794, 2021, accessed: 2023-09-27, 2021.
15. University of Toronto, E. PS/2 Controller. https://www.eecg.utoronto.ca/~jayar/ece241_08F/AudioVideoCores/ps2/ps2.html, 2023, accessed: 2023-07-02, 2023.
16. CD-ROM Cypress USB. https://download.terasic.com/downloads/cd-rom/de2-115/CD-ROM_Cypress_USB/, 2018, accessed: 2023-09-27, 2018.
17. Brendanalford. brendanalford/zx-diagnostics. https://github.com/brendanalford/zx-diagnostics, accessed: 2023-09-01.