ZX Spectrum (48K) in a FPGA

Jesús Arias

1 Introduction

The way to go computing here in Spain during the eighties was with a ZX-Spectrum, or, if you could afford it, with an Amstrad-CPC. Apple-twos and Commodores were for dollar-filled americans with NTSC monitors, and anything else was just a curiosity. So, it's no wonder a lot of retro revival is now focused on these systems, specially to the Spectrum due mainly to its sheer simplicity.

Sir Clive Sinclair, and Allan Shugart were the visionaries behind these computers and they surely deserve a recognition as high as Steve Jobs or Bill Gates, but this is not America, you know. And anyway, all these people were just that: visionaries. The actual people who designed these computers were engineers barely known if known at all (Wozniak is the only engineering star I can recall). These people had to work under the pressure of their visionary bosses who were always demanding the impossible for yesterday, and they did what they did. These computers were quite remarkable, but surely they could have been improved.

Looking at any Sinclair product today they all can be categorized as a good-looking, desirable, easy-to-break, crap. Their designers cut so many corners that they ended reinventing the hula-hop. And in that respect the ZX Spectrum is without doubt the highest quality product ever coming out of the Sinclair factories, something the owners of the Sinclair's electric vehicle surely recognize.

The Amstrad line of products feared better, but they were still paces away from, let say, equivalent Sony products (Not to mention Amstrad Hi-Fis). But digging under their surface you can see these products weren't the vanguard of the technology of their time. On the contrary, they were always years behind their competitors. Yet, they managed to win a considerable market share thanks to the CPC computer, and then Amstrad slowly faded away with the PCW typewriters and PC clones.

So let's start dissecting Spectrums and CPCs. The ZX spectrum was a hit because it offered a Z80 computer with a decent amount of memory for an affordable price, something their predecessors, the ZX-80 and ZX-81, didn't, and this was enough to persuade buyers to forget about the many inconveniences of the Spectrum, like...

- An horrible rubber keyboard that lasted no more than a thousand lines of code.
- An equally horrible RF modulator for the video signal. Yes, many TV's of that time didn't have a video input, so the computer had to be attached to the antenna jack. But the SCART connector was also starting to become common for new PAL TVs.
- A sadistic BASIC keyword input.
- Unreliable tape storage. It was not only slow. Not even a plain checksum was included to tell you the data you were loading was corrupted. So you had to wait until the tape stop making noises to realize something was wrong.

But first let's take a look to the electronics inside the case. One may ask what defines the hardware of a particular computer, and in the Spectrum case it is without doubt the Uncommitted Logic Array. The rest is mainly the Z80 itself, the memory, the power supply, and little more. So let's talk a little about the ULA.

1.1 Committing the Uncommitted Logic Array

Uncommitted Logic Array is a very descriptive name given by Ferranti to its line of Gate Arrays (other silicon manufacturers opted to name these kind of chips Gate Arrays). ULAs or GAs were chips with a lot of transistors arranged in a regular two-dimensional array. These chips lacked the final metal layer (in these days the ICs only had one metal layer), so they were not more than a collection of unconnected transistors, or in other words: they were "uncommitted". These chips were kept in wafers and stored until some customer, let say Mr Clive, come asking for the design of some digital chip for this or that computer he had on is mind. The Ferranti engineers then proceeded to design a metal layer for the ULA that matched the customer schematic and some wafers then went to the factory line to get the metal layer that "committed" them to become a video chip for the new ZX Spectrum.

By reducing the design of a custom chip to a single metal layer the cost of the chip was also much more affordable. At least if many thousands of them were to be ordered, this approach still was very expensive for small runs. And, of course, any mistake would be enough to ruin not only your budget, but also to delay your product by months or years. That was the main motivation for the development of modern day Field Programmable Gate Arrays. In these modern chips the connection between subcircuits isn't made with metal lines, but with multiplexers already built into the chip and controlled by digital data. But the basic idea is still the same: The FPGA is uncommitted until you upload some bitstream into its configuration memory, and then it performs the function you like.

But, if someone mention you an ULA you can be sure he's talking about a circuit committed to perform the duty of a video controller in a ZX Spectrum. The ULA of this computer was the best seller product of Ferranti, and for may of us the only known Ferranti product (I think that company also worked for the military, but when the Spectrum computer was over Ferranti went to the bottom with it)

The circuit inside the Spectrum ULA isn't very complex (In fact a modern Spectrum clone, the ZX Harlequin, is built using TTL chips instead). It basically includes:

- The horizontal and vertical counters for the generation of PAL video synchronism.
- The generation of video memory addresses, both for pixels and for video attributes (colors). These addresses are just the concatenation of some horizontal and vertical counter bits.
- The shifting of video data and the multiplexing of foreground and background colors.
- The arbitration of memory contention between itself and the Z80. This if achieved by forcing the CPU clock high during the time it takes to perform the video memory read by the ULA. In the Spectrum PCB resistors were placed in series with the Z80 address and data buses in order to save multiplexers: If the ULA and the Z80 contend for the memory the ULA is going to win because of its lower output impedance. This can be described either as a very clever trick or as a heap of shit due to the signal degradation these resistors introduce (I'm biased towards the second opinion).
- A single output port for setting the color or the border of the screen and for audio and cassette outputs.
- A single input port for keyboard scanning and for cassette input. The I/O port is accessed for any I/O address with the lower bit low. Again, electrical contentions can happen if an IN instruction is executed with more than one address bit low if there are expansion boards attached. Why not to decode the 8 low address bits of the bus instead of just A0 is something that puzzles me. So cramped was the ULA that a simple 8-input AND gate couldn't be fit?

But the main criticism I got about the ULA design is the way memory contention is managed, because when the ULA is displaying video the Z80 is going to be stopped very often. But the Z80 already include a /WAIT

signal that can force it to postpone its read or write operation until the video logic finishes its read. The use of the /WAIT input would require a more deep knowledge of the Z80 timings, but it will result in far less lost cycles than in the Spectrum ULA. BTW, this is how the memory contention is managed in the gate array of the Amstrad-CPC, where it results in one cycle penalty for 3-cycle reads and writes and no penalty for 4-cycle accesses like op-code fetches and I/O.

But if the GA of the Amstrad-CPC looks hi-tech take into consideration the one of the Inves Spectrum+, a clone of the Spectrum 48K made by Investronica, an Spanish company. In the Inves Spectrum+ there is no memory contention at all. Those guys at Investronica managed to run the Z80 without any wait state while also displaying video! That's top quality design! Also in that clone you won't find anything like the resistive multiplexing crap. Its a pity that such fine work was done for the clone of an aging computer. Not only that, the Inves Spectrum+ got a bad reputation due mainly to some games with compatibility issues because they relied on the flawed behavior of the original Spectrum, and also to some negative reviews in computer magazines. They went as far as stating that the Inves computer could be damaged if certain code was executed on it. A fake news of the eighties that ruined the reputation of a clone far better than the original.

"The ZX Spectrum ULA How to Design a Microcomputer" is an interesting book I read after the design of my FPGA replica of the Spectrum. It includes a lot of information about the internals of this chip and its many flaws, some of them deserving at least a comment. In my opinion the more serious are:

- The composite sync generated by the ULA during vertical retraces doesn't follow the PAL standard at all. It is only a single pulse. Even with this many TVs of the era were able to show an stable image. I'm not sure if the same happens with more modern TVs. Mr Clive surely was super happy with this simplification.

- During the vertical border of the image the 16KB DRAM stops refreshing. This time is 7.68ms while the DRAM chips specs state a maximum refresh period of 2ms. I wonder how many DRAM chips were labeled as defective due to this. This is a first order engineering blunder.

- Blinking characters leave vertical lines on the screen when off. This is due to combinatorial glitches and could have been fixed by registering the GRB outputs using flip-flops.

- The refresh pulses of the Z80 interfere with video memory reads in the ULA and causes an "snow effect" even when the 16KB DRAM never gets refreshed by the Z80.

What more can be said about the ULA? Only that it is a power inefficient and delicate chip. Its core logic is bipolar and runs with a 0.95V power supply. An internal voltage regulator is included in order to lower the 5V of the supply pin to the 0.95V of the core and a lot of electrical power is converted into heat in it. Also, the ULA can be easily damaged. The usual procedure for the destruction of an ZX Spectrum involves the connection of an expansion board, like a Kempston joystick, with the computer already powered. The most probable victims are the ULA and the ROM. The Z80 seems to be hard to kill, and that's remarkable because between its pins and the expansion connector there are nothing but copper traces. The ULA probably gets broken just because it's replacements are hard to find, while a more scientific explanation would be the latch-up problem where some parasitic transistors are turned on by a voltage transient and they short the power supply to ground. One should expect a more robust chip from a company that also worked for the military, but I'm starting to think Ferranti won the contracts just because it was British. Well, apart from the Spectrum ULA the only other barely known Ferranti chip was a shitty 3-pin AM radio receiver and almost all competitors were using CMOS at that time.

1.2 And what about Gate Arrays?

Well, in fact the TAHC10 of the already mentioned Inves Spectrum+ was a gate array made by Texas Instruments, and it was a CMOS chip, not a bipolar one like Ferranti's ULAs. But the Gate Array I want to mention now is that of the Amstrad-CPC. In this case it is difficult to point to a particular manufacturer because the chip comes with Amstrad markings (400xx). When compared to the Spectrum ULA the first thing you notice is the GA does less things than the ULA because the CPC in addition to the GA also includes a 6845 CRT controller.

What is doing a 6800 peripheral in a Z80 system like the CPC? Well, it is in charge of video timing and video address generation. It is fully programmable, allowing both PAL and NTSC timings, but this makes little sense in a computer sold along its own video monitor. The GA is thus in charge of video reading and shifting, memory contention management, palette storage, and little else. Some smart guy at Amstrad tough about using tristate outputs for the R, G, and B color outputs obtaining a 27-color palette instead of a 8-color one, and much of the GA complexity is related to this palette. But in overall the CPC GA seems to have little logic inside, and one starts to think why they didn't get rid of the 6845 by integrating a simplified version of it inside the GA.

There is no easy answer for that. Maybe the gate array wasn't big enough for all the logic, yet the Inves GA was capable enough of something like this. Or maybe the designers of the CPC were too conservative and resorted to a well known CRT controller, keeping the GA functions to a minimum in order to reduce the chances for mistakes. This last possibility is also supported by other curious schematic clues. For instance the sound chip, the AY-3-8912 PSG, isn't connected to the Z80 data bus directly. They resorted to drive the PSG bus from two ports of an 8255 PPI and to do some bitbanging for accessing the sound chip in the software.

Putting the PSG in the main data bus would have freed 10 pins of the 8255 at the cost of a few gates for the generation of the two control signals of the PSG. Well, this is already done in the Spectrum-128 and many other computers, so it wouldn't have been so difficult. And by the way, in the board they already have an underused gate array. So, why it is done this way?

Another probable answer is that the CPC board was designed in a rush and anything that worked in a breadboard was translated to the schematic without double thinking. "If it works keep it like that. Don't touch anything" may have been the order from above...

And BTW, the CPC also has the same incomplete I/O decoding we found in the Spectrum. This starts to look like a British tradition...

So the very revered CPC hides under its case a crappy design that ended converted into a standard due to the required compatibility with older systems. The 6845 and the PSG to PPI connection were present in all CPC models. Anyway, these ugly hardware details were hidden from the user via "warranty void if broken" labels and also by what was probably best BASIC ROM of all the computers of the time, even capable to put the GWBASIC of the PCs to shame. The 6845 programmability was also exploited to achieve a fast text scrooling by means of video base address changes, and in overall the CPC was a very decent computer with a fast BASIC, a perfect video quality thanks to its RGB monitor, a good quality keyboard, and a cassette integrated into the case later replaced with a floppy disk in high-end models. Why Mr. Shugart choose an incompatible floppy format is another mystery, but I think the 3-inch floppy, and also the QL microdrive, were the collateral damage of too much success in the market. "Everybody is going to go my way" may Clive and Allan have tough, fully intoxicated with money. Yet, this really happened with the IBM PC and with Intel USB, and dreams come for free.

1.3 The ZX Spectrum 48K

The ULA of the ZX Spectrum provides a single video mode with 256×192 resolution and 1 bit per pixel. In addition to this, there is a 32×24 attribute array where the background and foreground colors of the corresponding 8×8 pixel block are stored, along with an intensity bit that makes the colors more intense when one, and a

blink bit that makes the background and foreground colors to toggle every 320ms when one. These attributes seems to have been inspired by Teletext and they ended being a nightmare for game developers because they are the cause of the well known "attribute clash" problem. The contents of each attribute byte are:

7	6	5	4	3	2	1	0
Blink	Intense	GRB bg			GRB fg		

The three bits in the background and foreground color fields have the green component as the MSB and the blue component as the LSB. The idea behind this order was to achieve an increasing 8-level gray scale on black and white TVs (higher values always more intense than lower ones)

The video memory starts at address 0x4000 and its size is 6 KB for the pixel array plus 768 bytes, starting at 0x5800, for the attribute array. But the mapping between pixels and memory isn't an straight one. First, the screen is divided in three areas, each of 256×64 pixels wide, starting at addresses 0x4000, 0x4800, and 0x5000. In these areas each byte contains 8 pixels, with the MSB being displayed on the left. Increasing the address by one the position on the screen advances 8 pixels to the right until the end of the line is reached. But then 7 lines are skipped, so, the address 0x4020 corresponds to the beginning of the line #8 of the first area. Only when the address in increased by 256 we move to the next line. It seems that this strange mapping was selected to exploit the page mode read of the dynamic RAMs of the Spectrum.

Apart from the video, the ZX Spectrum is a very simple computer, with only one I/O port used for keyboard scanning and for the audio and cassette interface. This port is in fact two, one for reading and another for writing, and both ports are built inside the ULA. The output port also includes 3 bits for the selection of the color of the border of the screen. Interestingly, there are no output bits for the selection of the keyboard row. The high 8 bits of the address bus of the Z80 are used for this purpose instead. These ports are selected when an IN or OUT instruction is executed by the Z80 and the lower address bit, A0, is low. No I/O contentions were considered at all, so it is mandatory to have all the other address bits high to avoid them. The resulting I/O address is then 0xFE (the Z80 IN and OUT instructions use actually 16 bit addresses but the high 8 bits are only used when reading the keyboard) The bit fields of the input and output ports are:

Output port

7	6	5	4	3	2	1	0
-	-	-	Speaker	Cassette out	GRB border		

Input port

7	6	5	4	3	2	1	0
-	Cassette in	-	Keyboard row				

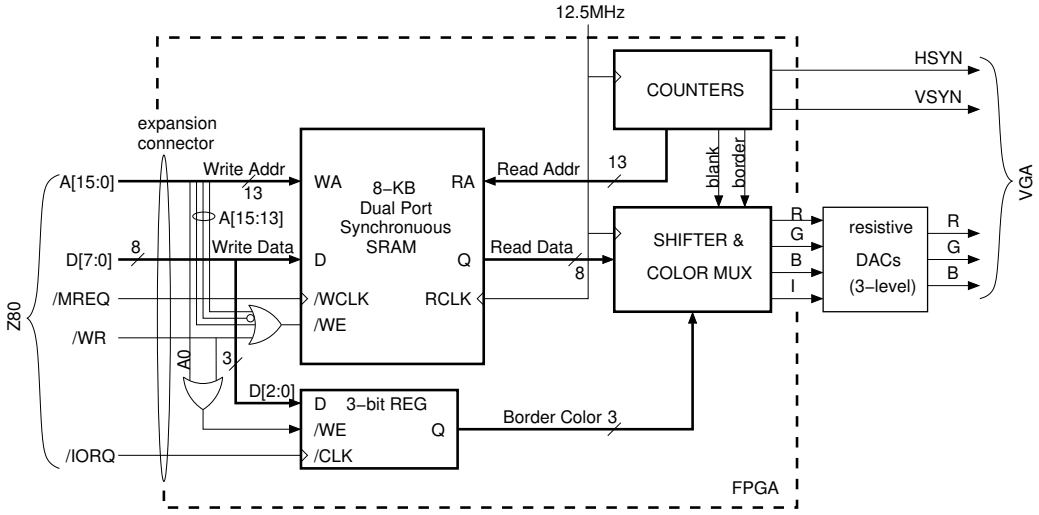
The ULA was short of pins and the cassette and speaker signals were all passed through a single pin. The designers were very proud of this and they even patented what, in retrospect, can be described as a crappy solution. The main idea was to generate a higher amplitude for the speaker pulses that for the cassette output. Two silicon diodes in series with the speaker then muted the cassette signal but allowed the higher pulses to reach the speaker. Well, you aren't supposed to apply a DC current to a speaker, to start with, so a diode in series to a speaker is something that causes nausea to any decent audio electronics engineer. Yet, this crap was patented.

With this knowledge about the ZX Spectrum internals we can start the design of our FPGA replica. We still have to know what keys are present on each keyboard row, that are:

IN address	row	col 4	col 3	col 2	col 1	col 0
0xFEFE	0	V	C	X	Z	Caps Shift
0xFDFE	1	G	F	D	S	A
0xFBFE	2	T	R	E	W	Q
0xF7FE	3	5	4	3	2	1
0xEFFE	4	6	7	8	9	0
0xDFFE	5	Y	U	I	O	P
0xBFFE	6	H	J	K	L	Enter
0x7FFE	7	B	N	M	Symbol Shift	Space

So lets start with the design of a compatible VGA version of the video controller. After achieving this, the rest of the computer is little more than the CPU and the memory.

2 Designing a Ghost-ULA



A project I tough about but I never built was an adapter for watching the Spectrum screen in a VGA monitor. The VGA timing is twice as fast as a PAL video signal and all the video electronics inside the Spectrum was useless. The only way I found to achieve this was to build an entire video controller for the VGA and to attach it to the expansion connector of the Spectrum. This circuit was too complex to build using TTLs, and therefore I tough about using an FPGA for this purpose. Interestingly, the FPGA I knew included a dual-port memory that was very well suited for this application as it allows to perform independent read and writes simultaneously. That means no worries about memory contentions, you can read and write the RAM at any rate you like. The only problem here is the small amount of memory of the FPGA, but the ZX video memory takes only 6,75KB and that was less than the available RAM in the FPGA.

The block diagram of the possible VGA adapter is shown in the above figure. I named it Ghost-ULA because it performs a similar role as the Spectrum's ULA but it is completely unnoticed by the computer. Any Z80 write to the video memory is also done to the GULA RAM, so, after an initial screen erase the contents of the Spectrum video memory and the FPGA memory are the same, and the image on the VGA screen is going to be the same as the one on the TV, but with a much better quality.

The GULA is a write-only device with addresses decoded to enable writings when the processor stores a data in the $4000 to $5FFF address range. But it would be easy to map the location of the GULA memory to any address multiple of 8KB, for instance by using 3 dip-switches, and to have two different screens, one in the TV and another in the VGA monitor. The GULA RAM can even be mapped to ROM addresses ($0000 or

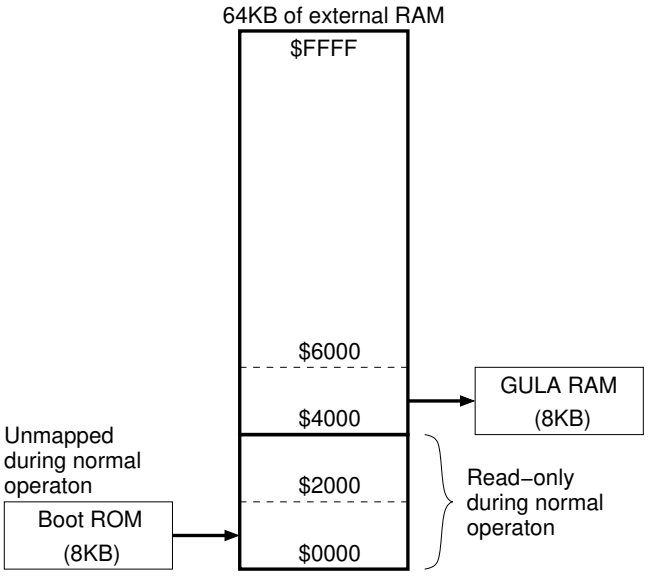
$2000), so no extra memory would be required for the VGA display, the only limitation being the impossibility of reading back the video data.

The GULA idea never ended becoming a board because the TV RF signal wasn't the only thing I disliked about my ZX Spectrum. Its keyboard membranes were also broken and I didn't want to replace them with the same crappy stuff. So I started thinking about building the entire computer inside the FPGA. When a found a Z80 core source in Verilog and I could get a decent estimate of the CPU size measured in logic cells, the new project started to take shape. And, of course, it will include the Ghost-ULA as its video controller.

3 The ZX++, An Spectrum with only Two chips.

I mean: one FPGA and one SRAM. Well, some other chips were also required, like two voltage regulators and one SPI Flash to store the configuration of the FPGA. That was the IC list of a board built around an ICE40HX4K FPGA. This chip is sold as a 3520 logic cell device, but the truth is it actually have 7680 logic cells. More than enough to fit a Z80 that requires about 2200 logic cells. This board was designed for the implementation of a range of 8-bits computers, and includes:

- One ICE40HX4K FPGA with 7680 logic cells and a total 16KB of dual-port synchronous SRAM.
- One 64K by 16-bit SRAM with an additional 128KB of memory.
- One N25Q64 SPI Flash memory with 8MB of nonvolatile storage.
- A 4096 color VGA DAC, 4 bits or 16 levels for each color component. The three DACs are built using resistors.
- A PS2 keyboard connector.
- A joystick connector.
- A single bit amplified audio output. Capable of Pulse Width Modulation.



In the board we have enough memory for the implementation of an Spectrum 48K. The main idea is to use 64KB of the external SRAM for its 48KB of RAM and 16KB of ROM, while the internal memory of the FPGA is used for the GULA and an additional boot ROM. This last memory is mapped to address $0000 at startup and the main purpose of its code is to fill the first 16KB of the RAM with the code of the Spectrum's ROM. This is easy to do because during startup the lower 16KB of the memory can be written. Also, during startup there are more hardware resources available, in particular:

- An output register at IO address $FF with the SPI bus signals (SCK, MOSI, and /Flash_CS) and two control bits. One of them is a clear only bit that selects between startup mode or normal mode. A reset will set this bit to Startup mode, with the Boot ROM mapped, all RAM writable, and extra IO ports. Once written with zero this bit can't be written again. The second control bit is related with the CPU clock divider. After reset this bit is set and the Z80 runs with a 25MHz clock. If clear the CPU clock is divided by 7, resulting in a 3.57MHz.
- An input register at IO address $FF with the MISO input from the SPI Flash memory.
- An 115200 baud UART at IO addresses $7F (data) and $BF (status) for debugging.

During startup the boot ROM implements a master SPI controller via bitbanging and reads the Spectrum ROM from the SPI flash. After this the following code is copied to the upper RAM and executed:

```
01 FF 33    ld    bc,$33FF ; Normal mode & slow clock
ED 41       out   (c),b
C3 00 00    jp    0
```

After the OUT instruction the system only has the standard Spectrum ports accessible:

- ULA write register at IO address $FE and ULA read register also at $FE
- Kempston joystick at IO address $DF

Also the boot ROM is no longer present, its content being replaced by the Spectrum ROM. The JP 0 instruction transfers the control to the ROM code and a "(c) 1982 Sinclair Research Ltd" message finally appears at the bottom of the screen.

The only remaining things we still have to implement to have a fully functional Spectrum are the keyboard and joystick logic. The keyboard is a little problematic because the PS2 interface has nothing to do with the plain switch matrix of the Spectrum, but it is still affordable. The idea here is first to receive the serial scancodes and to provide a data-valid signal for them. Then one must take into account that released keys send the same scancode preceded by a $F0 byte, and a key-release signal have to be generated if that code is received. Finally, the switch matrix of the Spectrum is simulated by a 40-bit RAM where a bit is turned on when a key is pressed or off when released. This RAM is read 5 bits at a time through the ULA's input port, the keyboard row being selected by the upper 8 bits of the Z80 address bus. Supposedly only one address bit should be low when reading, but if more than one bit is low the logical AND of the selected rows have to be returned. In this way the ZX++ gets a professional keyboard, the only problem is that the PS2 keyboard has no BASIC keywords written on the keys :(

And the joystick was a trivial peripheral, as long as it is of the passive switch type. But, as it turned out what I already had in stock was a game controller for the Gigatron computer. This controller is basically identical to a Nintendo's one, but it comes with a female DB9 instead of the weird NES connector, so it can be plugged into the FPGA board. And by an strange strike of luck the ground pin is the same as in a Kempston joystick. The FPGA has to change three pins of the joystick connector to outputs, one for providing power to the game controller chip, other for its parallel load signal, and the third for a shifting clock (the controller is basically a CD4021 chip). Inside the FPGA the clock is derived from the HSYN signal of the VGA, a load pulse is generated every 8 cycles, and the incoming data stream is converted to a parallel 8-bit word that is presented as the input data of the supposed joystick port. Of course, if you only want to use a simple Atari-style joystick all this logic can be removed.

3.1 Compatibility issues

As it happened to the Inves Spectrum+, the ZX++ is going to have problems with some software, especialy games. So let's point out the differences with a real Spectrum and what is expected to happen when running "too smart" programs.

- First, the Z80 runs a 2% faster than the original Spectrum 48K. This isn't supposed to be noticeable, and BTW, the Spectrum 128 also was a little faster than the 48K. But the lack of memory contention surely will make the ZX++ quite speedy.

- Next, one VGA line takes $32\mu s$ instead of 64μs, so any border bitbanging code will fail for sure. You weren't supposed to draw things on the border, you nasty rascal! But the number of visible lines is also doubled, so if an application is using the border to draw just a simple horizon this can still work. Well, it depends on the time the interrupt is posted to the CPU, in this case it is posted when the VSYN signal goes low, but this can be easily tweaked.

- And also the total number of lines in a frame isn't the double. The vertical total for the VGA mode is 525 lines. This is the same as in a complete NTSC frame, not a PAL one with 625 lines. Therefore we are posting an interrupt every 16.8ms, not every 20ms. Well, we got a NTSC Spectrum instead of a PAL one.

- The floating bus isn't simulated properly, so Arkanoid is going to hang. Still, the good thing about FPGA designs is that they can be fixed without touching a soldering iron. But I'm not a big fan of Arkanoid...

- Sound level differences between Speaker and cassette bits are actually simulated using PWM, with the speaker being 7 times louder than the cassette output (I like to ear the data sent to the cassette tape). No problems are expected here. But there is not feedback to the cassette input, so any code trying to determine the board issue by banging the audio bits is going to fail. This method is a crappy one because any noise at the MIC jack can ruin the test, so if you want to know your board issue pick an screwdriver up and break the "warranty void of broken".

3.2 Improving the ZX++

Half the SRAM is now unused and it its tempting to put it to do something useful. But unfortunately there isn't enough memory for an Spectrum 128, and I don't think a 96K or 112K Spectrum ever existed. In order to implement that with the existing board all the external RAM has to be used as RAM banks and the ROMs had to be implemented by reading the SPI Flash directly from the Z80. While this is in theory possible its going to be really slow because the SPI flash expends no less than 40 cycles for the reading of a single byte. This could be improved by using the remaining 8KB of internal RAM as a cache, but all the system starts to look just too complex.

Another direction for improvements are the peripherals and storage. A kempston joystick is already implemented and so is the cassette tape input and output. The ZX++ is able to load programs from an audio source, but this is as agonizingly slow as it was with real tapes, even if now we known the tape is going to be loaded correctly every time.

The boot ROM was able to load .SNA files into memory and to execute them. These files can be loaded through the startup serial port that is a hundred times faster than a tape. But in order to store the initial register values of the CPU a small region of the ROM code had to be overwritten (remember, during startup the low memory is writable). This was OK for the original 48K ROM because there were about 1KB of unused space, but other models have almost no free space in the ROM. A possible improvement would be the transient

mapping of an small internal RAM area for this kind of storage. The boot ROM should have to be reduced to free some RAM blocks.

And finally, with respect to the sound the implementation of an AY-3-89xx PSG into the FPGA is also another possibility, even if the 48K Spectrum lacked this chip.