Subject: Superfast Multicomp implementation!!
Posted by jonb on Thu, 22 Nov 2018 09:02:18 GMT
View Forum Message <> Reply to Message

Hi

I've been looking at some of the Multicomp implementations on the Wiki and wondering what the difference is, apart from the obvious (such as multiple serial ports). What I am looking for is a really fast CP/M FPGA machine. At the moment I have a standard Multicomp exactly per Grant's design running at 25Mhz which is pretty zippy, but I do wonder how much quicker it could be.

I saw a tidy looking board here: https://www.retrobrewcomputers.org/doku.php?id=builderpages:rhkoolstar:start - that uses a Cyclone IV breakout board. I'd like to ask, is it any faster than the Cyclone II or is it just a matter of providing more space / pins?

There was another thread discussing Cyclone IV Multicomp designs and wondering about adding graphics and other capabilities:
https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp ;amp ;th=79&start=0&

Did anything come of this? I did suggest to Grant that the ROM BASIC in his original design could be dropped in favour of a limited graphical capability on the Cyclone II but I don't have the expertise to do it.

Also, on the subject of performance, might a more efficient core like the NextZ80 be used for this? Grant's design uses a core called T80 at the moment.

Sorry, lots of questions! I've been playing with real CP/M hardware fro quite a while now and only just coming back to the Multicomp.

Thanks
JonB

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 22 Nov 2018 12:16:22 GMT
View Forum Message <> Reply to Message

Both Cyclone II and Cyclone IV miniboards are fitted with a 50 MHz crystal, limiting the 'processor' clock to 25 MHz.
However, using the built in PLL the system clock can be increased to 90(+) MHz (see this thread: Multicomp Cyclone II-C - 6809 running 90MHz via internal pll [message #1260]
but you need faster SRAM to make it work, possibly using an DIP-SOIC adapter discussed some time ago . (this works for both Cyclone implementations)

You can implement a limited graphical capability in the on-board text display (like the TRS-80) using graphical characters. i.e. extend the character ROM to include these and use escape codes to display them. This will probably not fit in the Cyclone 2 footprint of Grants design, unless you remove the BASIC interpreter from ROM.

When using the serial interface you already have all the graphical capabilities your terminal provides.

Rienk

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 22 Nov 2018 14:19:34 GMT
View Forum Message <> Reply to Message

OK, so I have a Cyclone II-C board which I bought over a year ago and I will implement that first. As I have not bought the 2x512k SRAM chips yet (it's currently running using jumper wires and a 128k SRAM I can look to replace those later. I'll have a look at the thread you mentioned as well, because this is pretty well what I am looking for.

As to the built in BASIC, it's not necessary for my use (there is always MBASIC under CP/M) so would be nice to free it up and extend the character set. I've also thought about changing the font as it looks too "PC-ish" for my taste. Problem is, I do not know where to begin...

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 22 Nov 2018 14:57:48 GMT
View Forum Message <> Reply to Message

You start by reading other peoples work. and figuring out how it works.
also find and install a compatible Quartus environment for your board (version 13.0 web edition for Cyclone II), which I guess you already have.

Get a working system (from Grants site, the wiki, or from here: http://www.smarthome.jigsy.com/fpga, preferably as close to what you are aiming for, then implement small changes, and make them work, learning as you go.

If you are just looking for a fast CP/M you can combine any implementation with the PLL by Bingo (originally Max Scane)

for implementing graphical features you have to dive in the SCBTextDisplayRGB code and figure out how that works

(or if you are lazy, you can find someone to do it for you, something I cannot recommend

Rienk

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 22 Nov 2018 15:10:49 GMT
View Forum Message <> Reply to Message

OK, Rienk, thanks.

I'll start with the fast implementation. Bingo's mod requires a 10ns SRAM, only available in a SOIC package, so I am now trying to find a source for the SOIC-DIP adaptor. The link originally posted is down and a search turned up nothing for Micronauts.com but I did find this, which looks similar:

http://www.signalogic.com/index.pl?page=sramadap

I've sent an email to the firm asking for a price with shipping. It looks like it might be applicable here.

Otherwise I would have to re-engineer the adapter board. Presumably not pin to pin, but rewired for the pin names to match. I'll probably end up doing this anyway because it is likely that Signalogic are going to charge a fortune. Get that fabbed at DirtyPCBs with panelling and I could have hundreds of the little blighters on my hands!

Any takers?

Subject: Re: Superfast Multicomp implementation?
Posted by Andrew B on Thu, 22 Nov 2018 17:03:16 GMT
View Forum Message <> Reply to Message

I bought a cheap Artix-7 board off eBay... It has 512MB of DDR3 RAM onboard... I wonder how fast a Z80 core could run on there?

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 22 Nov 2018 17:56:13 GMT
View Forum Message <> Reply to Message

You can ask guus.assmann on this forum. He probably has some SOJ / DIP converters available

If that does not work the included file might be convenient.

File Attachments
1) 10part90x100.zip, downloaded 59 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 22 Nov 2018 18:49:12 GMT

View Forum Message <> Reply to Message

Indeed it might.. what will I do with 80 of them, though? LOL!

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Thu, 22 Nov 2018 19:24:05 GMT
View Forum Message <> Reply to Message

>> I've also thought about changing the font as it looks too "PC-ish" for my taste. Problem is, I do not know where to begin...

Begin by realising that there is no such thing as ROM on the Cyclone. The character generator "ROM" is a RAM that is initialised from the serial eeprom at reset and that has no write port. You can change it to have a write port and then you just have to work out a way to address it and push in data.

If you aim to mess with the VDU design I recommend that you start with my version, on my 6809 github, as I fixed a bunch of bugs with the scrolling and the ANSI escape handling. I know also that Reink has done a bunch of mods on the z80 version, around using a ROM lookup table to implement keyboard mappings (but, boo hoo, my FPGA has no spare memory resources so I was unable to deploy it)

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 22 Nov 2018 22:15:00 GMT
View Forum Message <> Reply to Message

OK, but there is non volatile memory somewhere onboard, or where would the boot loader come from?

Grant talks about ROM on his web page but looking at the files in the ROM directory there is no actual data. So it is, I guess, defining the ROM chip but not its contents. However:

```
 altsyncram_component : altsyncram
 GENERIC MAP (
  clock_enable_input_a => "BYPASS",
  clock_enable_output_a => "BYPASS",
  init_file => "../ROMS/Z80/BASIC.HEX",
  intended_device_family => "Cyclone II",
  lpm_hint => "ENABLE_RUNTIME_MOD=NO",
  lpm_type => "altsyncram",
  numwords_a => 8192,
  operation_mode => "ROM",
  outdata_aclr_a => "NONE",
  outdata_reg_a => "UNREGISTERED",
  widthad_a => 13,
  width_a => 8,
```

width_byteena_a => 1
)

The init_file has the actual content. Do I take it, then, that the Quartus compiler reads the file (BASIC.HEX) and somehow puts it on the FPGA?

Clearly I've much to learn!

Subject: Re: Superfast Multicomp implementation?
Posted by etchedpixels on Thu, 22 Nov 2018 22:42:05 GMT
View Forum Message <> Reply to Message

Have a look at Will Sowerbutts SocZ80 system.

http://sowerbutts.com/socz80/

It's on a Papilio Pro FPGA board with DRAM. The T80 (FPGA Z80) CPU itself runs at 128MHz and the RAM is fronted with a 16K cache and MMU.

Apart from emulators on a fast x86 it's the fastest Z80 system I own.

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 22 Nov 2018 22:46:21 GMT
View Forum Message <> Reply to Message

Tge quartus compiler stores the configuration on the EPCS4 eeprom.
Every time you power up the machine the Cyclone chip is configured from that file.

in short, 8192 bytes or RAM are initialized and loaded with the BASIC file data.
the operation mode is set to ROM (no write port), so for all practical purposes you now have a programmed ROM in your system

There is a 'wizard' in the Quartus environment that can produce the above file, from specifications you enter in it

---

Subject: Re: Superfast Multicomp implementation?
Posted by ale500 on Fri, 23 Nov 2018 05:02:34 GMT
View Forum Message <> Reply to Message

Getting DDR3 to work needs quite a bit of work(There are tutorial and reference designs). The Artix 7 series are very fast, something like 2 times what you get with a Cyclone 4, of course they

are much more recent. It probably has enough block RAM to run a 64 kBytes CP/M machine without external components.

Subject: Re: Superfast Multicomp implementation?
Posted by Andrew B on Fri, 23 Nov 2018 06:43:42 GMT
View Forum Message <> Reply to Message

The Artix board I have had 225 KB of block RAM and ~50 KB of distributed RAM, so there is enough there for 64K + an MMU with a few pages.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 23 Nov 2018 07:30:26 GMT
View Forum Message <> Reply to Message

etchedpixels wrote on Thu, 22 November 2018 14:42Have a look at Will Sowerbutts SocZ80 system.

http://sowerbutts.com/socz80/

It's on a Papilio Pro FPGA board with DRAM. The T80 (FPGA Z80) CPU itself runs at 128MHz and the RAM is fronted with a 16K cache and MMU.

Apart from emulators on a fast x86 it's the fastest Z80 system I own.

SocZ80 is promising, but it doesn't look to be well documented and the Papilio Pro is somewhat difficult to acquire and relatively expensive.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 23 Nov 2018 09:22:26 GMT
View Forum Message <> Reply to Message

OK, I've been digging around the font question. I know of a rather clean looking 8x8 font for the Amstrad PCW called SanSerif. There's a picture of it in this PCWWiki forum thread I started showing the "standard" PCW font vs. Sanserif:

http://www.cpcwiki.eu/forum/nc100-nc200-pcw-pda600/lousy-pcw -screen-font/

The font comes wrapped in a short program SANSERIF.COM so you just run it at the CP/M command line to get it to take effect. I pulled the COM file, disassembled it, extracted the font bitmaps and generated Interl HEX files using the HxD hex editor. All that's necessary is to place them in the Multicomp\Components\TERMINAL\ directory, then rename them to CGAFontBoldReduced.HEX and CGAFontBoldExtended.HEX and rebuild the design in Quartus.

As I do not have a working Multicomp to hand (it's in pieces pending arrival of parts for the Cyclone II-C board) I can't test the font just yet. It's possible the characters won't look as good as they do on the PCW if the Multicomp's pixel shape is different, but if anyone wants to try, the files are attached.

File Attachments
1) CGASanSerif Multicomp Fonts.zip, downloaded 49 times

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Fri, 23 Nov 2018 11:20:55 GMT
View Forum Message <> Reply to Message

It works. The actual image is a lot sharper...

File Attachments
1) 20181123_122137.jpg, downloaded 679 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 23 Nov 2018 12:16:03 GMT
View Forum Message <> Reply to Message

I think that'll do nicely... There are a number of other 8x8 fonts that could be used but that's the cleanest looking one IMHO (out of the PCW fonts, at least).

By the way, there are three LEDs on the Cyclone II-C board. The one nearest the SD card is the drive activity light. One of the two others might be a power light. What is the third LED for?

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Fri, 23 Nov 2018 14:06:53 GMT
View Forum Message <> Reply to Message

Whatever you like. The LEDs are connected to pins 3, 7 and 9 of the FPGA
For your convenience I included the schematics of the miniboard.

File Attachments
1) MiniboardSchema.gif, downloaded 61 times

Subject: Re: Superfast Multicomp implementation?

Posted by b1ackmai1er on Sat, 24 Nov 2018 10:24:05 GMT
View Forum Message <> Reply to Message

[quote title=rhkoolstar wrote on Fri, 23 November 2018 03:20]It works. The actual image is a lot sharper... quote]

Hi Rienk,Is that the VGA output?

Regards Phil

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Sat, 24 Nov 2018 12:14:38 GMT
View Forum Message <> Reply to Message

yes, onto a 8.0" HE080IA-01D CHIMEI INNOLUX TFT LCD Screen Display 1024x768 and a controller board with AV, VGA and HDMI inputs.

I did not try AV, you want to see?

Rienk

Subject: Re: Superfast Multicomp implementation?
Posted by b1ackmai1er on Sat, 24 Nov 2018 12:25:53 GMT
View Forum Message <> Reply to Message

Only if it's easy

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Sat, 24 Nov 2018 13:10:18 GMT
View Forum Message <> Reply to Message

ok, I turned down the backlight all the way
AV does not scale on this display

File Attachments
1) 20181124_140709.jpg, downloaded 609 times

Subject: Re: Superfast Multicomp implementation?

Posted by b1ackmai1er on Sat, 24 Nov 2018 13:24:02 GMT
View Forum Message <> Reply to Message

Thanks! It's so long since I have switch my Multicomp on I have forgetten how to Time to revisit

Subject: Re: Superfast Multicomp implementation?
Posted by will on Tue, 27 Nov 2018 20:53:24 GMT
View Forum Message <> Reply to Message

Hello, I wrote socz80, it was my first Z80 project, and my first non-trivial FPGA project, what a deep rabbit hole this turned out to be...

The T80 core will clock to quite high speeds (100MHz+ on a Spartan 6). One nice property of the Z80 bus is that address and data signals are asserted and held stable for a cycle or two before the control signals are asserted so designs with timing dependencies on address/data decoding can often be clocked faster than the FPGA timing analysis tools suggest. I suspect there must be some way to express this to the timing tools, but I couldn't figure it out.

If you have enough SRAM on the gate array you'll get a very fast system indeed. If instead you have DRAM on your board you'll quickly find that (a) writing a DRAM controller is no fun at all and (b) modern DRAMs are designed for high speed burst transfers but not for low latency. Your Z80 will be idle virtually all the time, waiting for the DRAM to respond. This is why socz80 employs a relatively large SRAM cache to conceal this latency. I believe the newer Papillio boards have an external large (512KB or 2048KB) and fast (100MHz) SRAM instead of the DRAM, which likely addresses both issues pretty well.

Will

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 28 Nov 2018 10:30:20 GMT
View Forum Message <> Reply to Message

Hi Will

Grant's design uses SRAM but it's fairly slow at 55ns. In a separate thread user Bingo600 used a PLL in the Cylcone II to generate clock frequencies of 90Mhz and above. He discovered that the 6809 processor implementation (an optional piece of Grant's Multicomp design) can be clocked to 90Mhz with faster SRAM, so I wanted to see if the same could be done with the Z80 core. His thread is here:

https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp ;amp ;th=98&goto=1263

..though unfortunately he doesn't post his VHDL, and as a bit of a beginner with this I don't know where to start.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 28 Nov 2018 11:08:22 GMT
View Forum Message <> Reply to Message

rhkoolstar wrote on Thu, 22 November 2018 14:46Tge quartus compiler stores the configuration on the EPCS4 eeprom.
Every time you power up the machine the Cyclone chip is configured from that file.

in short, 8192 bytes or RAM are initialized and loaded with the BASIC file data.
the operation mode is set to ROM (no write port), so for all practical purposes you now have a programmed ROM in your system

There is a 'wizard' in the Quartus environment that can produce the above file, from specifications you enter in it

Thanks, Rienk.

I've just completed removal of the NASCOM BASIC from the boot ROM. Now I have a 1K monitor, which I have fitted into a new 1k ROM device in Quartus, and it has reduced total memory bits used from 89% to 40%. I now want to apply the changes to the VGA and keyboard implementation to free up some logic elements. Small steps..

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 28 Nov 2018 16:06:03 GMT
View Forum Message <> Reply to Message

On James Moxhams website http://www.smarthome.jigsy.com/fpga under VHDL you will find a Cyclone IV solution with the PLL implemented.
This is for a Cyclone IV miniboard with a 25 MHz oscillator (the so called blue miniboard)
The VHDL for the Cyclone IV very closely resembles the one for a Cyclone II

In the file Max-Scane-Sept-23-2015 you will find PLLto50.vhd as well as the definition in the main file CycloneIVb.vhd under SYSTEM CLOCKS
You can insert a PLL with a wizard too.

As you probably guessed, this is Max Scanes work.

This should set you on the right track for now

Rienk

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 28 Nov 2018 16:43:37 GMT

View Forum Message <> Reply to Message

Thanks Rienk, I will check it out now.

I have another question about the RTC module that is fitted to the Cyclone II-C carrier board. As you might imagine, I am looking for an interface implementation for it! There is an RTC in Max's zipfile but I think it is implemented inside the FPGA as a counter for one of the clocks (counts down each millisecond), rather than communicating with an external board.

The RTC itself is a DS1302 which is capable of keeping time, day, month, year and accounts for leap years up to 2100 AD which really ought to be enough (have we heard that before, I wonder?

So I would like to build out some VHDL that communicates with this module and exposes a set of I/O ports to the operating system so that I can get and set the date/time. It might be a good project to help me learn stuff. I think I will need to free up space by implementing the VGA optimisations first...

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 28 Nov 2018 17:31:27 GMT
View Forum Message <> Reply to Message

I also found out why the drive LED wasn't lighting up. The resistor values were too high. I've changed them from 5R6 to 150R and all's well.

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 28 Nov 2018 19:34:53 GMT
View Forum Message <> Reply to Message

Personally I don't use an RTC. I use an ESP8266 serial WiFi module on one of the UARTS. It feeds time via NTP.

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Wed, 28 Nov 2018 20:02:30 GMT
View Forum Message <> Reply to Message

>> So I would like to build out some VHDL that communicates with this module

the DS1302 uses an SPI interface. SPI is an interface that has no proper/consistent specification unlike, for example, I2C -- however, the behaviour is clearly described in the data sheet. It is simple enough and slow enough and used infrequently enough that you can just knife-and-fork it with GPIO under software control. In my 6809 design I build a GPIO module which was designed to occupy a minimal memory space (but you're on a Z80 so you have an independent I/O space to play with). I can oint you at some sample code but it's in 6809 assembler...

Neal.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 28 Nov 2018 21:45:07 GMT
View Forum Message <> Reply to Message

Well, it is fitted to a specific port on the Cyclone II-C carrier board called "RTC" in the schematic, so I hoped it was already supported. But no matter. It uses three lines which are physically wired to pins 40, 41 and 42 of the Cyclone II. As it appears to have a synchronous interface and you just toggle the bits to/from the data pin by flapping the clock pin about, it shouldn't matter whether or not it's driven at a specific clock speed - as long as I don't exceed its maximum (whatever that is). So what I need to do, I think, is modify the output latch VHDL code I found here and map it to a Z80 I/O port; then I can write a bit banging driver for it to put in the BIOS. The good thing about this is it should not use many of the limited remaining LEs in the Cyclone, as all the cleverness will be in the CP/M BIOS.

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Wed, 28 Nov 2018 22:17:26 GMT
View Forum Message <> Reply to Message

>> Well, it is fitted to a specific port on the Cyclone II-C carrier board called "RTC" in the schematic, so I hoped it was already supported

yes, it was added at my request and cunningly shoe-horned in by James. It is supported.. in my 6809 design ;-)

Here's the 6809 code; its structure and algorithms may be some some interest. For nitros I chose to make this a command-line utility that can read the RTC and set the system time; that avoided increasing the sode of the driver (or in your case the BIOS)

Neal.
https://sourceforge.net/p/nitros9/code/ci/default/tree/level 1/cmds/mc09rtc.asm

---

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 29 Nov 2018 07:18:43 GMT
View Forum Message <> Reply to Message

Thanks, Neal, it looks useful. Dragon, eh? I have one sitting on a shelf but it is only a 32.

I think adding RTC support to CP/M will be a nice little project. Did you write VHDL for it too?

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Thu, 29 Nov 2018 22:36:50 GMT
View Forum Message <> Reply to Message

I bought a dragon thinking it would be fun and would cure my 6809 craving, but I was disappointed with its keyboard and video output and it stayed in the cupboard. Once I built my multicomp that dragon went back the way it had co
me (ebay!!

I ported nitrOS9 to the multicomp and that's the only place where I've used the RTC. I ought to add support for it to my FLEX port as well..

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 08:06:49 GMT
View Forum Message <> Reply to Message

But to use the RTC in the Multicomp you must have needed to define some VHDL to set up the connection between the 6809 and RTC? That's what I would like to take a look at.

Meanwhile I am having fun with the PLL wizard. I'm setting one up that runs at the normal clock speed (input = clk, output multiplier = 1) and once I have this going I can try to ramp it up. I'm not expecting miracles until I have the fast RAM fitted. It occurred to me that one might use a PLL as a clock divider to remove some of the clock code in Microcomputer.vhd (thus saving some LEs). I also achieved a small improvement by changing the compilation settings in Quartus II. It has a sort of "optimisation adviser" that suggests better settings.

At the moment the PLL isn't working but I will persist.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 12:30:09 GMT
View Forum Message <> Reply to Message

Got the PLL running and feeding the CPU and SD card clocks directly, so the clock processing code in Microcomputer.vhd for both clocks is deleted.

Some performance results, using a fairly large C compilation SUBMIT job as a test:

With 25Mhz CPU, 1Mhz SPI clock, 268 secondsWith 25Mhz CPU, 25Mhz SPI, 107 secondsWith 33Mhz CPU, 25Mhz SPI, 82 secondsWith 33Mhz CPU, 50Mhz SPI, 82 secondsWith 33Mhz CPU, 66Mhz SPI, 82 secondsWith 33Mhz CPU, 75Mhz SPI, boot failWith 33Mhz CPU, 100Mhz SPI, boot fail

Results 4 and 5 are odd - perhaps the CPU becomes the bottleneck when the SPI clock is too fast?

This is using the ASC SRAM (slow) and standard SD controller VHDL, so it applies to Grant's original design. The card is a generic 2GB MicroSD (not SD-HC) so I could try a high speed card and the patched SD controller code, but the lack of performance increase above 25Mhz SPI suggests it isn't worth doing. The CPU clock cannot be pushed higher at this time (it won't start up if faster than 33Mhz), possibly due to the low SRAM access speed.

So, for the time being I am running with configuration 3 and it feels very snappy indeed. The next step is to fit fast memory and ramp up the CPU clock some more. I received some RAM adaptor cards this morning (thanks Guus! but the ICs are still in the post.

In the meantime, I will look at the RTC implementation.

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Fri, 30 Nov 2018 13:05:11 GMT
View Forum Message <> Reply to Message

the spec for a SDSC states:
Quote:the maximum SD Bus speed is specified by the
maximum SD clock frequency (25 [MB/sec] = 25000000 [Byte/sec] at 50 MHz) and data size is based
on memory boundary (power of 2).
You are probably driving the card out of spec in cases 5-7 (nothing wrong with that but ymmv)

Take care to mount the RAM chips the 'wrong way around' on the adapter cards when you get them. I.E look carefully at the footprint.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 14:14:47 GMT
View Forum Message <> Reply to Message

Oh yes, I see what you mean, pin 1 indicator (a circle, oddly enough) is at the bottom... thanks!

Regarding the SD clock, why might it work at 50Mhz but not yield any performance increase on a major bit of disk I/O over 25Mhz? Do you think it's the CPU speed holding it back?

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Fri, 30 Nov 2018 14:24:47 GMT
View Forum Message <> Reply to Message

jonb wrote on Fri, 30 November 2018 05:30Got the PLL running and feeding the CPU and SD card clocks directly, so the clock processing code in Microcomputer.vhd for both clocks is deleted.

Some performance results, using a fairly large C compilation SUBMIT job as a test:

With 25Mhz CPU, 1Mhz SPI clock, 268 secondsWith 25Mhz CPU, 25Mhz SPI, 107 secondsWith 33Mhz CPU, 25Mhz SPI, 82 secondsWith 33Mhz CPU, 50Mhz SPI, 82 secondsWith 33Mhz CPU, 66Mhz SPI, 82 secondsWith 33Mhz CPU, 75Mhz SPI, boot failWith 33Mhz CPU, 100Mhz SPI, boot fail

JonB,
Could you publish the files you used for the benchmark? I have a 22MHz Z80 SBC with a compact flash disk. I'm curious to know how that compares with a SD disk.

I also found an Altera Stratix board (1S25) made by Parallax (the Propeller people) around 2004. The Stratix FPGA has large internal RAM blocks (128KB) so I can build a Z80 SBC with T80 VHDL, RAM blocks and SPI interface to SD disk without adding any external components. Something to play around with over the holiday.
Bill

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Fri, 30 Nov 2018 14:27:12 GMT
View Forum Message <> Reply to Message

Not odd, reversing the chip makes routing the board easier.
This way the adapter board should fit inside the DIP footprint so you can use two of them in adjacent sockets

Yes, the SD card routine in BIOS uses many more CPU cycles to read or write a byte.
Also there is overhead for calculating and setting the Logic Block Address for every block of 512 bytes.
You can only speed this up by clocking the CPU higher or making the BIOS code more efficient.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 15:13:52 GMT
View Forum Message <> Reply to Message

Ah right, I should have considered the blocking/deblocking algorithm and LBA derivation. I've used this code myself in an IDE driver.

@Bill, I will have to do a bit of "Kermiting" to get it uploaded. It's a CP/M port of stevie (a vi clone written for the Atari ST) which is quite usable on a 25Mhz CP/M box like the Multicomp, but

painfully slow on a real 4Mhz Z80 box. The code compiles with Aztec C, which is on Drive C User 2 of the Multicomp demo disk image. Just copy the files in the archive to this location then do SUBMIT BUILDALL and time the result. You also need a patched version of SUBMIT.COM as the normal one doesn't like running off a non boot drive. I've attached a copy.

[Edit: Darn it. Kermit is not happy sending from the Multicomp but it receives just fine, so I'll go out on a limb and post the original P2000C source, which should work on the Multicomp.]

[Edit 2: STevie does not display CP/M text files correctly because it is not expecting CR/LF (the DOS and CP/M end of line sequence). So instead it does the carriage return, but shows the line feed as a hex character [0x0D]. You can still edit files, though; even binary (that's why there is a representation of a non printable character). If you use TYPE to look at any of the source files, you will find they don't contain CRs.]

File Attachments
1) STevie.zip, downloaded 53 times
2) SUBMIT.zip, downloaded 49 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 15:21:30 GMT
View Forum Message <> Reply to Message

rhkoolstar wrote on Fri, 30 November 2018 06:27Not odd, reversing the chip makes routing the board easier.

I meant it was odd to use a circle on the silkscreen to indicate pin 1. It's usually a dot, or a notch in the outline. But then, I've very little experience with these SMD chips. Let's hope they arrive quickly... can't wait!

Subject: Re: Superfast Multicomp implementation?
Posted by etchedpixels on Fri, 30 Nov 2018 16:22:43 GMT
View Forum Message <> Reply to Message

I am surprised you got the SPI as fast as you did. SD is not rated for that high a clock rate so the fails are no surprise (I'm surprised you got over about 33MHz working). The faster SD modes are multi-bit wide rather than faster clock so need a different connection arrangement. The really fast ones also do link tuning and other deep magic.

By way of comparison. On a real 3.5MHz Z80 (ZX Spectrum) the highest raw rate I can get off an SPI (ZXMMC) card is 218KBytes/second, and the highest rate I can get off ATA is the same. In both cases it comes down to how fast the ini instruction runs. Even in PIO0 the upper limit for the transfer rate off an ATA card is 3.3MBytes/second so a 25MHz Z80 is physically not capable of reaching the transfer speed. On the ZXMMC it's arranged so that straight ini and outi loops can drive the SPI. You are fundamentally bounded by how fast you can issue ini instructions even if in unrolled loops. The only way to beat that limit would be to implement a DMA engine.

The performance is also bounded the operating system overhead. CP/M is not optimized for large fast I/O (CP/M 3 is a chunk better as it can use 512 byte sectors directly, it also allows multi-block read/write but most apps don't use the feature). 218KByte/second does not translate to anything remotely like that in real use on CP/M or indeed on any other OS.

On the LBA mapping - if you are cunning with your disk geometry the LBA translation becomes trivial. I prefer not to - but to take the hit and also add partition table support.

Alan

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 17:02:42 GMT
View Forum Message <> Reply to Message

Thanks for the explanation, Alan. I'll see what happens when I can ramp up the CPU speed, then.

Meanwhile, tried the updated TERMINAL module from http://www.smarthome.jigsy.com/fpga. I hate to say this, but it doesn't seem to work properly on the Cyclone II. The screen gets corrupted when starting up WS.COM. I've checked the declaration in my top level VHD file (Microcomputer.vhd) and it is the same as the corresponding declaration in CycloneIV.vhd. So for now I reverted back to the original module.

Despite these minor problems, I am very pleased with the way it is turning out.

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Fri, 30 Nov 2018 18:42:08 GMT
View Forum Message <> Reply to Message

>> to use the RTC in the Multicomp you must have needed to define some VHDL to set up the connection between the 6809 and RTC? That's what I would like to take a look at.

yes, the code is here:

https://github.com/nealcrook/multicomp6809/blob/master/multi comp/Components/GPIO/gpio.vhd

and you may also need to refer to the top-level hookup here:

https://github.com/nealcrook/multicomp6809/blob/master/multicomp/MicrocomputerPCB/Microcomputer.vhd

Neal.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Fri, 30 Nov 2018 18:57:49 GMT
View Forum Message <> Reply to Message

Great, Neal, I'll check it out immediately.

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Fri, 30 Nov 2018 22:37:38 GMT
View Forum Message <> Reply to Message

JonB,
Thanks for the files. I compiled it on my 22MHz Z80 with compact flash disk. The time to run buildall.sub is 103 seconds. So if this is a 25MHz equivalent Z80, it should run about 90 seconds, so CF still has a significant advantage over SD with SPI clocking at 25MHz.

I see what you meant by STevie being slow. When editing a new file it seems OK, but when I insert new lines in the middle of a modest size file, the screen refresh is slow. Hardware handshake for the serial port is required or I can easily overrun the input buffer. The carriage return is weird just like you've said.
Bill

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Fri, 30 Nov 2018 22:39:46 GMT
View Forum Message <> Reply to Message

Out of interest, how big is the vi executable created from STevie? Bingo600 and I worked on resuscitating a vi clone in C under 6809 FLEX -- changing it to support ANSI control sequences. The final executable was ~15Kbytes. It had the usual restriction that the file to be edited had to fit in memory. Perofmance-wise it runs fine on a ~25MHz 6809.

Neal.

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Fri, 30 Nov 2018 22:49:52 GMT
View Forum Message <> Reply to Message

vi.com is 28K.
Bill

PS, I have not used vi for 20 years, my head doesn't remember the commands but my fingers do! It is a weird sensation puttering around the screen; My fingers know what to do but my mind is blank, just observing how cursors magically moved around, text added and file saved and exited. It is like magic!

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Sat, 01 Dec 2018 10:55:30 GMT
View Forum Message <> Reply to Message

>>my head doesn't remember the commands

me too. My Children use that "muscle memory" to play Piano. Mine seems to be allocated to keystrokes for emacs (and, sadly, WordStar)

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Sun, 02 Dec 2018 10:35:44 GMT
View Forum Message <> Reply to Message

More fun & games: I have integrated the TERMINAL component from Max Scane's September 23 2015 zip file and it's working nicely. Now I have the following usage on my Cyclone II:

Logic Elements 85%
Combinatorial functions 81%
Dedicated Logic registers 25%
Total memory bits 45%

So, I think there is enough space to implement some more goodies.

Regarding vi.com (well, "stevie.com" really).. I'm the same. Struggle with any other text based editor, especially Wordstar. The new bindings in the TERMINAL component certainly help, though. If I could just recall the "save" command... (^Ks, how arcane).

There are other vi implementations; one in particular can be found on comp.os.cpm which is quick enough but it is too big to be any use on CP/M - you can edit a 6k file maximum. I set STevie to 16k which allows me to edit its source files. STeveie is slow because of the way it determines how to refresh the screen. It's got a double buffer and compares one against the other in order to work out what's changed. This is also how ncurses works on Linux (and curses before that on other *nix systems) but of course it runs on faster hardware. STevie also does not take advantage of the

termianl escape sequences other than clear screen and cursor location, so a rewrite is definitely on the cards...

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Sun, 02 Dec 2018 12:33:00 GMT
View Forum Message <> Reply to Message

nealcrook wrote on Fri, 30 November 2018 14:39Out of interest, how big is the vi executable created from STevie? Bingo600 and I worked on resuscitating a vi clone in C under 6809 FLEX -- changing it to support ANSI control sequences. The final executable was ~15Kbytes. It had the usual restriction that the file to be edited had to fit in memory. Perofmance-wise it runs fine on a ~25MHz 6809.

Neal.

Hi Neal, could you post the [a link to] source code, please? I'd like to try it on the Z80 Multicomp.. Thanks!

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Sun, 02 Dec 2018 13:32:28 GMT
View Forum Message <> Reply to Message

>> the TERMINAL component from Max Scane's September 23 2015 zip file

be aware that this code inherits a bunch of bugs in the escape sequences from Grant's original. You can see a list of bugs and fixes in the git log for my version:

commit 81d4ca77b268c411e3170dd84c65550735597c93
Author: nealcrook <neal@pippaluk.org.uk>
Date: Fri Nov 4 18:29:30 2016 +0000

    Complete the fix to the line insert and line delete flows. The previous fixes
    did not work in the corner-case of insertion or deletion on the bottom line.
    In fixing that, I made some other adjustments so that the insert and delete
    copy always proceeds from 0 to line end; this should reduce the number of
    comparators needed because it makes the behaviour consistent with the line
    clear flow.

    Also, a few white-space changes.

commit f5c9a12364bb8043379da1ab1ac7edd8bd62772e
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Mon Jul 18 22:45:31 2016 +0100

Make ESC key generate 0x1b. This required taking F11 and F12 out of the
set of "special keys" that generate hardware signals out rather than
key-codes (but that is no loss because this design does not use that
capability.
Fix line insert and line delete - previously the data copy was happening
in the wrong state so that, for example, insert would corrupt the lower part
of the screen with a single repeated character. That may be a bug that I
introduced when I switched this piece of the design to single-edge clock.
Previously, insert copied characters starting at the right and ending at the
left, while delete copied characters starting at the left and ending at the
right. There was no reason for this difference and they both now work in
the same direction (right to left).
Both line insert and line delete misbehave on the bottom one or two lines
(lines 24 and/or 25). Don't think I caused this bug. The fix will require
a re-jig of the insert and delete loops.

commit 5cf41525c35821c4bdbde98c8e97e3c784ce2000
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Sat Jun 11 19:32:23 2016 +0100

Tweak keyboard mapping so that ESC key generates 0x1b (escape).

commit aa7cc221554c3c03ba5a4e8a19ed915bd055dc0d
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Tue Mar 22 23:14:09 2016 +0000

Modified keyboard map to add support for 102-key keyboard #~ key
(key42) and \| key (key45). This was a trial-and-error process as
the scan codes for these keys are not documented clearly.

commit f44d4eddb40ce8f449fdc5f2009d3dc53789ffdc
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Sat Feb 13 15:55:01 2016 +0000

Clean up the functional reset generation in the UART to make it
glitch-free. Extent its use to reset the pointers of the input FIFO.
Add glitch-free functional reset generation to the VDU. Use it to reset
the pointers of the keyboard input FIFO.
Previously, after running CUBIX and pressing "RESET" could not start up
BUGGY successfully: the PS/2 and/or UART input would be non-responsive.
My theory is that the pointers were getting messed up and these reset
changes are intended to fix that. Both VDU and UART now support the 6850
"master reset" of writing 3 to the control register.
Added "master reset" of serial port in BUGGY. Still need to do the same
in FLEX.

commit df18391019efdc4ea5eda80a43e30296995ee945
Author: Neal Andrew Crook <neal@pippaluk.org.uk>

Date: Fri Jan 1 20:26:13 2016 +0000

    Bug-fix to ESC[K (erase to end of line). Previously, the cursor position
    would be corrupted (cursor sent HOME) by thsi sequence. Now the cursor
    position is preserved.

commit e9ffefd3ceab3b1e3e7e04ffe37569c6d0a700c5
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Fri Jan 1 19:38:25 2016 +0000

    Whitespace changes and comment fix-ups; no functional changes.

commit 940f7594431edec8da26feaf59fc3634b53f07a0
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Tue Sep 22 21:47:12 2015 +0100

    Couple of whitespace changes.

    Main change is to use a function to truncate an 8-bit value (expressed in hex)
    to a 7-bit value. This allows the keymap to be represented in a slightly
    clearer way. Added comments to the keymap to show the keys and to describe
    the format.

commit c302e303647dc8936a2b4cd36f54d497e0af0ca7
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Tue Sep 22 20:23:27 2015 +0100

    Change all processes to use rising edge clock. Previously they used a mixture
    of rising and falling, and this made the timing very tight (did not close
    timing with 50MHz clock). Moving everything to rising clock has no impact
    on functional performance; the pixels just come out 2 cycles later. Has a
    great beneficial effect on timing though. This effectively re-pipelines the
    output which required one counter comparator to change.

    Fix functional bug brought to light by CUBIX. CUBIX does line endings as
    LF CR rather than the usual CR LF. There was a bug in the display logic on
    scroll: the line was cleared from the cursor position to the end rather than
    from the start of line to the end. After a CR the cursor position was always
    at the left so this bug was hidden. The effect of the bug was a wierd effect
    on the bottom line which then scrolled up the screen. Took *ages* to work out
    what was going on but, once spotted, was very straightforward to fix.

commit ba6cbea7ebc14b49863ed95ce8bcc0710130e2d3
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Tue Sep 22 20:15:25 2015 +0100

    No functional change. Add names to all the processes, to aid debug in the
    simulator. Add more comments to the keyboard decoding code.

commit bc78e557e263de2faadd79041807cf5accbd070b
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Tue Sep 22 20:09:30 2015 +0100

    No functional change. Two groups of syntax changes so that the design
    will load into modelsim without overflow error and without "Actual
    expression of formal is not globally static"

commit 5bf7eb860f9d2991a1067f9fd85e8e4556b383d5
Author: Neal Andrew Crook <neal@pippaluk.org.uk>
Date: Tue Sep 22 19:50:37 2015 +0100

    Grant Searle's original sdcard and terminal VHDL code, posted with his
    permission.

Subject: Re: Superfast Multicomp implementation?
Posted by nealcrook on Sun, 02 Dec 2018 14:42:49 GMT
View Forum Message <> Reply to Message

Here are the sources for ved2, which I extracted from flex drive 2 of the latest sd card image for multicomp09 (multicomp09_sd_apr2017.zip)

IIRC this could compile and run on a Linux system and so the C must be "reasonably" modern (but probably more K&R than ANSI). However, I could not get that to work today. I will look and see if I have some additional files on another machine.

I'm not sure how much work will be involved in getting this to work under your z80 C compiler but if you do get it running it will be interesting to compare code size and performance against the 6809 version

Neal

File Attachments
1) ved2.zip, downloaded 45 times

Subject: Re: Superfast Multicomp implementation?
Posted by etchedpixels on Mon, 03 Dec 2018 00:12:09 GMT
View Forum Message <> Reply to Message

Be aware that Harry Suckow (who ran Manx and owns the rights) has only given permission for educational and fair use redistribution of Manx products rather than abandoning them.

Alan

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Tue, 04 Dec 2018 09:53:52 GMT
View Forum Message <> Reply to Message

Fun with the RTC..

This is a port of Neal's 6809 program to CP/M that reads the DS1302 RTC and sets the Multicomp's internal RTC. As I haven't added the internal RTC VHDL yet, this program reads the time / date from the chip and dumps it as hex.

Or not.. I think it's broken...

File Attachments
1) DS1302.jpg, downloaded 422 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 06 Dec 2018 11:22:46 GMT
View Forum Message <> Reply to Message

I've now built a "fast RAM" board using a Samsung K6R4008 and SOIC36 - DIP adapter board. Nasty tiny soldering.

It doesn't work, and what's more, it seems to have damaged the Cyclone board.

I've attached a picture. In the foreground is the board with chip mounted per the advice given here (chip to be mounted upside down relative to IC socket). Behind it is another board, unpopulated, showing the circle that indicates Pin 1 of the SOIC package. Just want to confirm I have built it correctly, because Bingo600's adapter board has the IC the other way round. A different adapter, but I thought I'd better check.

Comments?

.

File Attachments
1) IMG_5059.jpg, downloaded 56 times

Subject: Re: Superfast Multicomp implementation?
Posted by b1ackmai1er on Thu, 06 Dec 2018 13:40:12 GMT
View Forum Message <> Reply to Message

Holy cow, how did you even solder that! I would never be able to do that.

I thought the carrier boards were meant for TSOP chips like this ... ???

I thought the chip had to be reversed on the carrier board ... so check if you've inserted it the wrong way around in your multicomp or if the chip has to be facing down ... i.e. carrier pins should be sticking out the other way ?

Regards Phil.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 06 Dec 2018 13:46:01 GMT
View Forum Message <> Reply to Message

Same way that Bingo600 did, I guess. Yes, they do look like they should accept TSOPs.

You've linked to a 44 pin TSOP but the carrier board takes a 36 pin package, so that particular chip won't fit.

I wonder if I have fake ICs?

The chip is mounted upside down - that is, relative to the orientation of the notch on the socket, Pin 1 of the chip is furthest away from the notch. I'm having doubts that that is the right way to mount it.

Subject: Re: Superfast Multicomp implementation?
Posted by b1ackmai1er on Thu, 06 Dec 2018 13:54:10 GMT
View Forum Message <> Reply to Message

Yes sorry, I just checked the original post and my multicomp and that orientation does look right.

Your layout does match Bingo600's as you say.

Have you tried checking for hidden solder bridges.

I am keen to try this myself on an ECB-RAM Floppy board

Regards Phil

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 06 Dec 2018 16:20:45 GMT
View Forum Message <> Reply to Message

Hi Phil

Yes, I checked adjacent legs to see if there were any shorts. None found.

It's not the same as Bingo's. His adapter board is designed so that the IC has the same orientation as the DIP socket in the carrier board. But I wasn't sure, so I asked.

I now have another problem. The Cyclone II-C card might be damaged. I am unable to run the Multicomp in it, even with the minimal Microcomputer.vhd (as shown on Grant's site, a 4k Z80 machine running BASIC on the Cyclone board, with no other attachments). This configuration runs fine off the carrier board.

I am also unable to bring up the CPM machine's boot prompt with the Cyclone board on its own (goes without saying it won't run on the carrier board, either). I believe I should get the prompt but go no further on a bare Cyclone II board.

Bit of a nightmare.. will have to go back to first principles again.

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 06 Dec 2018 18:37:34 GMT
View Forum Message <> Reply to Message

Hi Jon,

As far as I can see you did everything right. It is a 3v3 36 pin SOJ, connected as it should. Just check the Vss and Vcc lines pin 10 & 27.
Possible failures may include soldering (finicky allright) or bad ic. Really not much more choices here.

Your miniboard failure might be unrelated to the memory chip. I have a defective one, the failure of which is a mystery to me. You can program it, but it just does not work.

Rienk

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Thu, 06 Dec 2018 18:54:14 GMT
View Forum Message <> Reply to Message

One possible failure maybe mechanical connections. The process of plugging and unplugging

memory board disturbed a weak or intermittent joint resulting in failures unrelated to the memory board itself. Visually inspect boards for broken solder joints and conductive particles. Let the board warm up a bit, press and twist the board and see if you observed different behavior.
Bill

Subject: Superfast Multicomp implementation?
Posted by guus.assmann on Thu, 06 Dec 2018 20:14:32 GMT
View Forum Message <> Reply to Message

Hello,

For my boards, I've soldered the memory chip the same way as you did. The silkscreen is correct.
And if the complete memory adapter is in the multicomp, I can see the Ram-chip.
I'm not sure how you put the adapters in your board.

One thing I can confirm.
If you solder the ram chip 180 degrees wrong on the adapter, it will not work.
However, after removing it and puting it on the right way, my Ram-chip still did work 100% ok.

BR/
Guus

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 06 Dec 2018 21:24:46 GMT
View Forum Message <> Reply to Message

OK, so I have the memory chip installed correctly on the adapter. Good. It is plugged into the carrier board same way round as the Alliance RAM chip - with the notch to the edge of the board. This must surely be correct.

The Cyclone miniboard works if I program it with a minimal system (Z80, 4K internal RAM, BASIC). However, If I then plug it into the Cyclone II-C carrier board, it doesn't work. I checked the serial connections, they match and have continuity. Something odd is going on. But I cannot see why I get no monitor with the "full CP/M" configuration (with the miniboard not connected to the carrier board). I should at least see the boot prompt. I have examined the carrier board for damage; I can't see any. But it did flex a lot when I fitted the RAM adapter. Maybe a track got damaged. The programming cable was underneath it at the time and I noticed that some of the IC socket pins had pierced the insulation slightly. When I removed the programming lead it started working again, but has since failed. Maybe another indication of board damage due to flexing?

I've ordered another miniboard. It'll be here by February, I am sure...

Thank you all for your advice, I do appreciate it.

Cheers

JonB

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Mon, 10 Dec 2018 15:51:11 GMT
View Forum Message <> Reply to Message

Spent some time over the weekend with the Multicomp.

I reflowed all the joints and reprogrammed the Cyclone. Took a while but it is working now (at full speed - 33Mhz with a 50Mhz SPI clock for the SD card). Phew, that's a relief. I think the moral of this tale is "support the board when inserting RAM chips", because it does not like being bent.

I'm now hoping to get the fast RAM working (still unsure whether they are fake chips). I assume I need to set the jumpers to the 512k setting but are there any VHDL changes required (just for 128k initially)?

Subject: Superfast Multicomp implementation?
Posted by guus.assmann on Mon, 10 Dec 2018 20:17:02 GMT
View Forum Message <> Reply to Message

As per request, a detail of my board.
Notice the position of the red jumpers.
The pins on the Ram adapter boards are quite thin where they sit in the socket.
So no problem there. Soldering was not so easy, but I managed.

BR/
Guus

File Attachments
1) Multicomp.jpg, downloaded 53 times

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Tue, 11 Dec 2018 00:46:42 GMT
View Forum Message <> Reply to Message

JonB,
I managed to overclock my 22MHz Z80 to 29.5MHz by raising the voltage to 5.5V. It passes ZEXALL.COM so I thought I'd try the STevie benchmark to see whether it scales proportionally to 29.5MHz with the CF interface. It did the benchmark at 77 seconds and yes, it does scale.
Bill

c>submit buildall

c>CZ WINDOW
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS WINDOW
8080 Assembler Vers. 1.06D

c>CZ NORMAL
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS NORMAL
8080 Assembler Vers. 1.06D

c>CZ MISCCMDS
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS MISCCMDS
8080 Assembler Vers. 1.06D

c>CZ LINEFUNC
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS LINEFUNC
8080 Assembler Vers. 1.06D

c>CZ HEXCHARS
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS HEXCHARS
8080 Assembler Vers. 1.06D

c>CZ CMDLINE
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS CMDLINE
8080 Assembler Vers. 1.06D

c>CZ EDIT
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS EDIT
8080 Assembler Vers. 1.06D

c>CZ MAIN
C Vers. 1.06D Z80 (C) 1982 1983 1984 by Manx Software Systems

c>AS MAIN
8080 Assembler Vers. 1.06D

c>LN -O VI.COM WINDOW.O NORMAL.O MISCCMDS.O LINEFUNC.O HEXCHARS.O CMDLINE.O EDIT.O MAIN.O C.LIB
C Linker Vers. 1.06D
Base: 0100 Code: 6519 Data: 0543 Udata: 1394 Total: 007df4

c>

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Tue, 11 Dec 2018 15:18:06 GMT
View Forum Message <> Reply to Message

I've built another fast RAM module and I'm happy to say it is working.

I think I may have accidentally fried the other chip by feeding 5v to the 3.3v line on the board (oops). Anyway...

Here are some additional performance results, using the full STevie build as a benchmark:
CPU clock 50Mhz, SD clock 50Mhz: 57 seconds
CPU clock 55Mhz, SD clock 50Mhz: shows monitor prompt but will not boot CP/M
CPU clock 60Mhz, SD clock 50Mhz: shows monitor prompt but will not boot CP/M
CPU clock 75Mhz, SD clock 50Mhz: will not boot (but reset pops up some characters)
CPU clock 100Mhz, SD clock 50Mhz: will not boot

So it seems that for the time being 50/50 Mhz is as far as I can take the CP/M Multicomp. A faster SD card may allow me to up the SD clock (and / or the CPU clock to 60Mhz), using Neal's SDHC VHDL.

I now have some options to explore:

SDHC VHDL for the SD ControllerMore RTC fiddlingFit a second fast RAM module for 1MBBuild out MMUsTry CP/M Plus and the other OS-es that use paged RAM

Insane to think the Multicomp only cost about £30, yet I'm having so much fun with it...

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Tue, 11 Dec 2018 15:57:29 GMT
View Forum Message <> Reply to Message

57 seconds! Very cool. I can see the attraction of having a fast processor. There is a snap to it that is very pleasing.

How well does the FPGA Z80 runs ZEXDOC.COM? During my overclocking test of the Z80, I've noticed that a Z80 may fail some ZEXDOC.COM tests but still run up CP/M.

Bill

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Tue, 11 Dec 2018 16:36:19 GMT
View Forum Message <> Reply to Message

I don't know, Bill - you'll have to tell me how to run it / where it is. I have the Multicomp demo disk (might be derived from the N8VEM demo image or whatever it is called), is it on one of the user areas?

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Tue, 11 Dec 2018 16:59:26 GMT
View Forum Message <> Reply to Message

I found ZEXALL and ZEXDOC distributed in YAZE-AG (Yet Another Z80 Emulator). ZEXALL tests all undocumented instructions which T80 VHDL probably won't pass, but it should pass ZEXDOC which tests the documented Z80 instructions. Here is the ZEXDOC.COM that I extracted from YAZE. Just type "zexdoc" at CP/M and wait. It took 26 minutes with a 30MHz Z80, so it should takes 15 minutes or so at 50MHz.
Bill

File Attachments
1) zexdoc.zip, downloaded 33 times

Subject: Re: Superfast Multicomp implementation?
Posted by wsm on Tue, 11 Dec 2018 20:12:23 GMT
View Forum Message <> Reply to Message

Bill: I'm not so sure that ZEXDOC is actually a good benchmark program, especially for FPGA implementations. I was curious to try it on a Z180 even though I know there are a couple of minor differences in DAA, RLD and RRD. I'd forgotten that I'd tried this program quite a LONG time ago with the same result ... it executes invalid Z180 opcodes which cause TRAPs.

Specifically, the first TRAP is in the "aluop a,<ixh,ixl,iyh,iyl>" test which tries to use just the high and low portions of the IX and IY registers: i.e. opcode DD84 which is an undocumented "ADD A,IXh". This may be valid on the Z80, however it TRAPs on the Z180.

I do have the source but it's project #799 on the possible "to do" list. There are quite a few of the tests that need to be commented out: i.e. alu8rx, incxh, incxl, incyh, incyl, ld8ixy etc.

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Wed, 12 Dec 2018 01:21:59 GMT

I may be way off, but my expectation for the FPGA Z80 is it should pass the ZEXDOC with little or no errors. When turning up the processor clock, we need a instruction set test to make sure the longest path instruction is not broken so a thorough instruction tests, even with known errors, like ZEXDOC is needed.

Bill

PS, ZEXDOC runs reasonably well on Z280. This is the result of Z280 in the 8-bit Z80-compatible mode:

```
------------------------------ZEXDOC--on ZZ80RC #2-----------------
b>zexdoc
Z80 instruction exerciser
<adc,sbc> hl,<bc,de,hl,sp>....  OK
add hl,<bc,de,hl,sp>.....  OK
add ix,<bc,de,ix,sp>.....  OK
add iy,<bc,de,iy,sp>.....  OK
aluop a,nn.....  OK
aluop a,<b,c,d,e,h,l,(hl),a>..  OK
aluop a,<ixh,ixl,iyh,iyl>.....  OK
aluop a,(<ix,iy>+1).....  OK
bit n,(<ix,iy>+1).....  OK
bit n,<b,c,d,e,h,l,(hl),a>....  OK
cpd<r>.....  OK
cpi<r>.....  OK
<daa,cpl,scf,ccf>.....  ERROR **** crc expected:9b4ba675 found:ae5fe733
<inc,dec> a.....  OK
<inc,dec> b.....  OK
<inc,dec> bc.....  OK
<inc,dec> c.....  OK
<inc,dec> d.....  OK
<inc,dec> de.....  OK
<inc,dec> e.....  OK
<inc,dec> h.....  OK
<inc,dec> hl.....  OK
<inc,dec> ix.....  OK
<inc,dec> iy.....  OK
<inc,dec> l.....  OK
<inc,dec> (hl).....  OK
<inc,dec> sp.....  OK
<inc,dec> (<ix,iy>+1).....  OK
<inc,dec> ixh.....  OK
<inc,dec> ixl.....  OK
<inc,dec> iyh.....  OK
<inc,dec> iyl.....  OK
ld <bc,de>,(nnnn).....  OK
ld hl,(nnnn).....  OK
```

ld sp,(nnnn)..... OK
ld <ix,iy>,(nnnn)..... OK
ld (nnnn),<bc,de>..... OK
ld (nnnn),hl..... OK
ld (nnnn),sp..... OK
ld (nnnn),<ix,iy>..... OK
ld <bc,de,hl,sp>,nnnn..... OK
ld <ix,iy>,nnnn..... OK
ld a,<(bc),(de)>..... OK
ld <b,c,d,e,h,l,(hl),a>,nn.... OK
ld (<ix,iy>+1),nn..... OK
ld <b,c,d,e>,(<ix,iy>+1)..... OK
ld <h,l>,(<ix,iy>+1)..... OK
ld a,(<ix,iy>+1)..... OK
ld <ixh,ixl,iyh,iyl>,nn..... OK
ld <bcdehla>,<bcdehla>..... OK
ld <bcdexya>,<bcdexya>..... ERROR **** crc expected:478ba36b found:96e927a2
ld a,(nnnn) / ld (nnnn),a..... OK
ldd<r> (1)..... OK
ldd<r> (2)..... OK
ldi<r> (1)..... OK
ldi<r> (2)..... OK
neg..... OK
<rrd,rld>..... OK
<rlca,rrca,rla,rra>..... OK
shf/rot (<ix,iy>+1)..... ERROR **** crc expected:713acd81 found:dfbbe74b
shf/rot <b,c,d,e,h,l,(hl),a>.. ERROR **** crc expected:eb604d58 found:43e7b72d
<set,res> n,<bcdehl(hl)a>..... OK
<set,res> n,(<ix,iy>+1)..... OK
ld (<ix,iy>+1),<b,c,d,e>..... OK
ld (<ix,iy>+1),<h,l>..... OK
ld (<ix,iy>+1),a..... OK
ld (<bc,de>),a..... OK
Tests complete

Subject: Re: Superfast Multicomp implementation?
Posted by wsm on Wed, 12 Dec 2018 05:24:50 GMT
View Forum Message <> Reply to Message

I just wanted to point out that although the user is led to believe that ZEXDOC only tests documented instructions, it actually tests undocumented instructions as well. As such, I find the distinction between ZEXDOC and ZEXALL very misleading.

I agree that an FPGA exerciser should test as many of the instruction permutations as is practical but I'm not going to debate whether it should emulate the original core for undocumented

instructions. My approach is very simple ... undocumented means unsupported and not valid for the programmer to use. Just because an undocumented instruction works on Zilog's Z80 silicon, there is absolutely no guarantee that it will work on other Zilog (or the many other manufacturer's) devices that are documented as "code compatible with Z80". Unlike the Z80, the Z180 TRAPs invalid opcodes which I find to be a useful aid in debugging and checking for compatibility.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 12 Dec 2018 07:13:09 GMT
View Forum Message <> Reply to Message

ZEXDOC ran in just under 16 minutes and passed all tests on the 50Mhz Multicomp.

When I have finished sorting the RTC out I'll time it properly.

[Edit: 15 minutes 30 seconds to be more precise..]

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 19 Dec 2018 12:02:11 GMT
View Forum Message <> Reply to Message

I've discovered why it won't clock past 50Mhz. The TERMINAL component cannot handle the speed! So now, I am testing by activating the serial port.

Build of vi.com:

CPU 60Mhz, SD 15Mhz - 52 seconds
CPU 60, SD 30 - 48 seconds
CPU 60, SD 40 - 48 seconds (CPU is not keeping up with the SD card)
CPU 60, SD 50 - doesn't boot CP/M
CPU 65, SD 25 - (can't setup PLL with 65/30) - no boot
CPU 70, SD 25 - (can't setup PLL with 70/30) - no boot
CPU 75, SD 20 - no boot

In each instance of boot failure, the TERMINAL screen shows corruption. Recall that it always prints the prompt on there: "Press [SPACE] to activate console", so perhaps this is causing the Multicomp to crash once we exceed its maximum operating speed. Let's try to comment the TERMINAL out (from Microcomputer.vhd) and see if serial-only operation allows a boost:

CPU 65, SD 25 - no boot
CPU 75, SD 25 - no boot
CPU 70, SD 33 - no boot

Hmm. This does work at the lower speeds so I assume it's not failing because I've not removed the TERMINAL component properly.

For now, then, I think it points to a maximum clock speed of 60/40 for serial only, or 50/50 with the TERMINAL component. Given 50/50 occasionally fails to boot, and that there is no difference between 50/50 and 50/40, I will stick with 50/40 until the TERMINAL component's performance can be improved.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 19 Dec 2018 14:03:47 GMT
View Forum Message <> Reply to Message

..and now, after all the experimentation, I can't boot CP/M at all... I wonder what has gone wrong now?

Edit: Not to worry. During testing the boot track must have got corrupted somehow. Efforts to restore it failed, until I realised that it's necessary to upload CPM22.HEX before CBIOS128.HEX in the monitor, as there is an overlap (where the CBIOS vector table is).

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 19 Dec 2018 14:21:55 GMT
View Forum Message <> Reply to Message

It seems to me that you should run the timing specific components (terminal, baudrate uart etc) from the input signal (50 MHZ) of the PLL, not from the output

---

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 19 Dec 2018 14:55:54 GMT
View Forum Message <> Reply to Message

Hi Rienk

Yes, that is what I do. Only the CPU and SD card clocks come off the PLL - TERMINAL and baud rate divider both use the 50Hz clock (I haven't incorporated a BRG yet).

I was wondering if TERMINAL could be run faster, but I expect it uses clk for the timing of the video signal and will break in short order if I change it to use the PLL.

I edited my previous post to say that I'd restored it now.

Thanks
JonB

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 19 Dec 2018 15:56:28 GMT
View Forum Message <> Reply to Message

Now I am experimenting a bit more, with the more basic configuration (4k internal RAM, BASIC interpreter, PLL, Z80, UART). It seems to be happy to run at 95Mhz. At 100Mhz it will boot into the BASIC interpreter, but locks up as you enter the first line of code. My plan is to build it up incrementally in order to see where the bottle necks are.

Let's swap out the internal RAM for the external fast RAM module:
95Mhz - works.
100Mhz - works.
110Mhz - works.
120Mhz - works.
125Mhz - works.

At this point I am beginning to get a bit

130Mhz + fast RAM - tries to boot but fails. Maybe it is working, but the CPU is too fast for the serial port? Let's try going from 115200 baud to 230400... to do this I set the baud rate setting line in Microcomputer.vhd to "serialClkCount <= serialClkCount + 4832;". Amazingly, it actually works at this speed, with the 125Mhz CPU clock, but it will not run at 130Mhz so baud rate's not the issue.

Just for fun, though, I've upped the baud rate to 460800 and it still works (even faster on screen updates). The next step was 921600 baud with a clock increment of 19,328 and while it did boot to BASIC it didn't quite work when I flooded it with data (as in: 10 FOR X+1 TO 10000: PRINT X;: NEXT X).

So, I have a maximum CPU speed with external RAM of 125Mhz and a 460800 baud serial port. I know that the moment I build in the SD card or VGA TERMINAL it will fail.. maybe I can introduce CPU wait states or something like that?

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 20 Dec 2018 10:09:55 GMT
View Forum Message <> Reply to Message

I've integrated Neal's SDCARD component (the one updated for SDHC) and reran the STevie compilation test. It completed in 2:01 with 50/50 Mhz clock speeds, which surprised me. I've reverted back to standard for now.

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 20 Dec 2018 10:35:49 GMT

View Forum Message <> Reply to Message

Neals SDCARD component is very conservative. It normally runs with a 500 kHz serial clock. My version (attached) initializes as 250 kHz but runs at 25 MHz otherwise.

The SD card specification states that the initialization clock speed should not exceed 400 kHz. This is to accommodate some very slow early cards.
My version switches to full speed after init. full speed, being half of the master clock (in my case 25 MHz)

File Attachments
1) sd_controller.vhd, downloaded 44 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 20 Dec 2018 12:20:47 GMT
View Forum Message <> Reply to Message

Thanks, Rienk - there will be a short delay while I fix a corrupted A: drive on my SD card... Not sure what happened but I suspect running Neal's SDCARD component at 50Mhz may have contributed to it..

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 20 Dec 2018 13:30:54 GMT
View Forum Message <> Reply to Message

The price of progress...

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 20 Dec 2018 15:40:42 GMT
View Forum Message <> Reply to Message

OK then, how about this (attached)? It's another SanSerif type font, but thinner than the PCW one. I think it needs to be less tall as well..

(Edit: Fixed the position of the full stop in the hex file attachment)

File Attachments
1) SanSerifThin font.jpg, downloaded 74 times
2) CGASanSerifThinReduced.hex, downloaded 48 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 20 Dec 2018 16:10:45 GMT

View Forum Message <> Reply to Message

Or.. with a small mod to the VGA parameters in the SBCTextDisplayRGB.vhd file we can have a proper square pixel (at the moment they are twice as high as they are wide).

```
  constant VERT_PIXEL_SCANLINES : integer := 1;
```

instead of

```
  constant VERT_PIXEL_SCANLINES : integer := 2;
```

The thin font looks a bit better like this, but the lines shown on the screen are all you get (in other words, it only uses half of the panel). It may be possible to double the number of lines but I expect the DisplayRam would need to be doubled, too. You also have the problem of CP/M programs assuming 80x24.

File Attachments

1) square pixels.jpg, downloaded 55 times

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Sun, 23 Dec 2018 16:42:33 GMT
View Forum Message <> Reply to Message

I decided to venture into the high speed arena too.

I now have a Cyclone II system (my version .. see builderpages) running stable at a 120 MHz system clock. (PLL times 12 devided by 5.
I only had to re-calculate the values of the baudrate generator. Every component uses the 120 MHz base clock except the SBCTextDisplayRGB.
Both CPU clock and the SDCARD SPI clk run at 60 MHz (SDCARD inits at about 450 kHz)
With some tweaks I could fit a VGA terminal plus 2xUART+BRG, MMU and SDCARD components.
two IS61LV5128AL-10 SRAMS are fitted.
ZEXDOC runs under 13 minutes

Subject: Re: Superfast Multicomp implementation?
Posted by tingo on Sun, 23 Dec 2018 17:45:50 GMT
View Forum Message <> Reply to Message

Nice! This thread makes me want to get back to my multicomp experiments

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Mon, 24 Dec 2018 10:25:16 GMT
View Forum Message <> Reply to Message

Hi Rienk

How did you manage that? Or, perhaps I should say "What am I doing wrong?".. I assume since you ran ZEXDOC that you have CP/M up?

Cheers
JonB

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Mon, 24 Dec 2018 12:37:13 GMT
View Forum Message <> Reply to Message

Not too much to it.
I used my base setup from my builder pages.
I added the PLL, and generated new baudrate and interrupt values based on the factor 2.4
I had trouble fitting it all so I made some optimizations (I basically accepted all changes suggested by Quartus) and modified the SDCARD module a bit.
There was no benefit in changing the VGA timings, so I left them alone running on 50 MHz.
It now fits and runs.

You will need my BIOSes for this setup. CP/M images for Dos+, CP/M 2.2, CP/M 3, MP/M 2, 'ROM' BASIC, ZSDOS and ZPM3 can be found on my builder pages
Alternatively you can exchange the ROM file for your specific configuration.

I attached the VHD files. make sure the pin assignments fit your hardware.

Success Rienk

File Attachments
1) Microcomputer_VGA.zip, downloaded 48 times

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Mon, 24 Dec 2018 13:52:20 GMT
View Forum Message <> Reply to Message

Thanks, Rienk! I'll check it out.

Just a minor point though. You say the clock is 120Mhz but ZEXDOC is completing in just under 13mins. My 50Mhz setup completes in 15.5 minutes. I'd expect you to be getting 6 mins or less... am I missing something?

Anyway, happy Christmas!

Subject: Re: Superfast Multicomp implementation?
Posted by plasmo on Mon, 24 Dec 2018 13:59:51 GMT
View Forum Message <> Reply to Message

ZEXDOC is CPU and memory intensive, so 120MHz Z80 should finish the test in 6.5 minutes, unless there are lots of waits for memory access.
Bill

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Mon, 24 Dec 2018 14:03:18 GMT
View Forum Message <> Reply to Message

The CPU clock is 60 MHz. (120/2). 5/6*15.5=12.9

I tried 125 MHz. This mostly runs, but not stable. so I stuck to 120.

Oh, and in case you want to know, I'm using a Chinese knockoff 2GB SDHC micro SD card. It says "Kingston" on it, but somehow I doubt that...

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Tue, 25 Dec 2018 13:13:42 GMT
View Forum Message <> Reply to Message

OK, I see. You're using a clock divider in your in your VHD, so running the PLL at 120Mhz. What I do is wire the PLL output direct to my CPU and SD card so I do not have any intermediate VHDL. I set the CPU and SD at 50Hz in the PLL Megafunction wizard. I can run at 60/40 (as documented above) and I guess that'd give me the same ZEXDOC score as you. But... the T80 core can run at a genuine 120mhz with the external RAM and UART, so I am looking to get the TERMINAL and SD card to handle it, too (not the speed necessarily, but to not crash when the CPU is that fast).

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 26 Dec 2018 15:57:42 GMT
View Forum Message <> Reply to Message

I don't know. Anything above 60 MHz CPU clock is either unresponsive or unstable. Sometimes it seems to work, but then zexdoc will stall somewhere in the middle.
Also with a higher clock it won't be faster. I feel maybe clock pulses get skipped. Timequest does not like it at all

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 27 Dec 2018 08:15:31 GMT
View Forum Message <> Reply to Message

Try this...

It's a minimal Multicomp with BASIC only, running the Z80 at 120Mhz and the serial line at 460800 baud. Requires fast external RAM. As I said before, it fails once TERMINAL or SDCARD components are added so it's no use for CP/M, but it does demonstrate what's possible.

If we want to take CP/M beyond 60Mhz, I think it'll be necessary to work out why TERMINAL and SDCARD don't like running with a faster CPU.

File Attachments

1) Multicomp speedtest.zip, downloaded 41 times

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Thu, 27 Dec 2018 10:56:34 GMT
View Forum Message <> Reply to Message

I suspect that with added components the logic chains gets too long. Every gate adds delay and the added delays 'overrun' the clock. The sd card and terminal component do not cause the problem, it's the combination of all the parts. So you can make a really fast simple ROM-based machine, or a slower complex one. I don't think there is a simple fix for this. We're just running into the hardware limitations.

Subject: Re: Superfast Multicomp implementation?
Posted by b1ackmai1er on Thu, 27 Dec 2018 13:10:45 GMT
View Forum Message <> Reply to Message

Just as an idea to explore ...

Maybe you need to make the modules asynchronous so they are "speed independant".

i.e. add ready/wait signals, add input buffers.

Regards Phil.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Thu, 27 Dec 2018 13:23:20 GMT
View Forum Message <> Reply to Message

You mean synchronous, surely?

Neal suggested something similar. He said that the design was asynchronous like old computers but this is not how it's done today. I don't really know how to make it synchronous, or if the additional logic will fit. I did see that the Z80 wait_n line is fixed at 1, though (ie not waiting) in the standard Microcomputer.vhd.

Subject: Re: Superfast Multicomp implementation?
Posted by etchedpixels on Thu, 27 Dec 2018 15:09:24 GMT
View Forum Message <> Reply to Message

You make them asynchronous to the CPU core.

When you write to the relevant I/O port you latch everything you need and assert wait within the CPU clock domain. When your device finishes the I/O it deasserts wait.

You can also run at a subset of the CPU clock without all the extra glue logic with a bit of care. The penalty you pay is that if you talk too fast to it then stuff breaks. That's sort of how the 1MHz floppy controllers were nailed onto 4MHZ processors and is how things like the VDP on the MSX work. Even an LDI takes 16 clocks, so your interface doesn't have to run at anything like the CPU clock to keep up with the processor.

To give you an idea of how much you don't need the clock - the Sinclair Spectrum has a clock signal on the expansion connector that is basically broken and unusable on many machines. All the expansions work just fine without needing access to the CPU clock signal.

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 02 Jan 2019 13:38:16 GMT
View Forum Message <> Reply to Message

Not sure.
I've been experimenting with my Cyclone IV board. I can run it up to 132 MHz (66 MHZ CPU/SPI clock) with the system remaining stable.
At 140 MHz (70 MHz CPU/SPI clock) the system will boot, run CP/M, copy files (with verify) etc, but running ZEXDOC will show errors.
ZEXDOC will produce the same behavior either running from SD card or RAM disk.
This does not feel like a peripheral interface overrun.
Perhaps the increased LE count creates too much load on certain gates to sustain the necessary slew rates. (just a guess)

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 09 Jan 2019 12:38:45 GMT

I posted 2 configurations (for Cyclone II and IV) in my builderpages.
It took some effort to get all components to 'behave'. Finally I got it stable using two PLL outputs of the same frequency but shifted by 90 degrees. One supplies CPU/ROM/SD and the other the textterminal. Don't ask me why this works, but it seems to do the trick.
The only issue I still have is that the IV system occasionally drops an output character in the serial terminal. I'm still working on that

The Cyclone II operates at 120 MHz (60 MHz CPU frequency). The Cyclone IV at 132 MHz (or 66 MHz CPU).
noteworthy: The Cyclone IV system clock speed represents 'true' clockspeed (compared to a real Z80) while the Cyclone II operates at 83 percent of that (even when not overclocked). Thus here the IV solution is 33% faster than the II while the clock is only 10 % faster.

btw I use this mbasic80 code as a benchmark:

```
10 PRINT "Find prime numbers between 1-5000"
100 L=5000
110 DIM N(L)
120 INPUT "How many iterations";Q
125 SM=PEEK(67): SS=PEEK(68)
130 FOR T=1 TO Q
140 FOR I=1 TO L
150 N(I)=1
160 NEXT I
170 P=2
180 FOR I=P+P TO L STEP P
190 N(I)=0
200 NEXT I
210 P=P+1
220 IF P=L THEN 240
230 IF N(P) <> 0 THEN 180 ELSE 210
240 NEXT T
250 EM=PEEK(67): ES=PEEK(68)
260 S=(INT(SM/16)*10 + SM MOD 16)*60 +(INT(SS/16)*10 + SS MOD 16)
270 E=(INT(EM/16)*10 + EM MOD 16)*60 +(INT(ES/16)*10 + ES MOD 16)
280 S=E-S
290 IF S<1 THEN S=S+3600
300 PRINT S;" seconds"
305 P=0
310 FOR I=1 TO L
320 IF N(I)>0 THEN P=P+1
330 NEXT I
340 PRINT P;" primes"
```

0x4C,0x4D stores minutes and seconds in BCD in my system.
10 iterations for system II produces 83 seconds, for system IV 56.

Subject: Re: Superfast Multicomp implementation?
Posted by etchedpixels on Wed, 09 Jan 2019 14:08:32 GMT
View Forum Message <> Reply to Message

Did you run all the validation tests and did it find any propogation timing errors, or other warnings about power or fan out ? In theory the tool chain should tell you if you are violating a timing constraint.

If you've got clock propogation or other timing problems then clocking bits of the board the way you are will sometimes fix it (and indeed is a real world thing where you delay the clock to match the propogation time of the related signals). If you have bidirectional signalling however just delaying the clock usually introduces new problems. There are "proper" text book ways to do this with latches between the time domains or with an asynchronous FIFO each way. There are also pulse stretching techniques that are hairier but use less resources.

Once you have a FIFO you can clock the terminal entirely independently (eg far slower) than the system proper.

Alan

Subject: Re: Superfast Multicomp implementation?
Posted by rhkoolstar on Wed, 09 Jan 2019 15:05:54 GMT
View Forum Message <> Reply to Message

Currently I am just winging it. I wasn't planning on redesigning the whole system yet. There are still many things I don't really understand but I must say it is a whole lot of fun to have for just a few bucks.

Subject: Re: Superfast Multicomp implementation?
Posted by jonb on Wed, 09 Jan 2019 17:39:30 GMT
View Forum Message <> Reply to Message

Amen to that. Love the response of the thing, so quick. I'm almost reluctant to go back to using real hardware now!

But, latest project: https://stardot.org.uk/forums/viewtopic.php?f=45&t=16316

..I can't help myself!

Subject: Re: Superfast Multicomp implementation?
Posted by bingo600 on Thu, 31 Jan 2019 12:27:12 GMT
View Forum Message <> Reply to Message

Well done guyzz - Seems to be some speedy Z80's there.

In order to "shut up Quartus warnings" , i used some timing constraints , that "might" have improved the routing.
If you want to try it out , uncomment the one needed.

Or you might already have done this .. I'm no VHDL (or Quartus/ISE) expert , gust used google.

```
#**************************************************************
# Create Clock
#**************************************************************

# 50MHz Osc in
create_clock -name {pll_clk} -period 20.000 -waveform { 0.000 10.000 } [get_ports {pll_clk}]

# 50MHz Pll clock ut
#create_clock -name {clk} -period 20.000 -waveform { 0.000 10.000 }

# 75MHz Pll clock out
#create_clock -name {clk} -period 13.333 -waveform { 0.000 6.667 }

# 80MHz Pll clock out
#create_clock -name {clk} -period 12.500 -waveform { 0.000 6.250 }

# 90MHz Pll clock out
create_clock -name {clk} -period 11.111 -waveform { 0.000 5.556 }

# 100MHz Pll clock out
#create_clock -name {clk} -period 10.000 -waveform { 0.000 5.000 }

# Haven't really got any idea about what this means , but it "shuts up quartus"
derive_pll_clocks
```

/Bingo