Subject: TinyZZ, a Z280 SBC
Posted by plasmo on Fri, 26 Jan 2018 18:13:30 GMT
View Forum Message <> Reply to Message

Created a new topic to continue discussion from here:
https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp
;th=93&goto=4155&#msg_4155
I don't want to hijack the CPU280 thread, so this topic is about a different implementation of Z280, TinyZZ. It is not retro, is build from scratch. Since I have had no previous experiences with Z80 or its derivative chance are excellent the design won't end up where I wanted. The main design objective is CHEAP. It is a disposable computer to experiment with. I don't want to tear down experiments just to salvage and re-use the computer. To keep the cost down and setting up experiment easier, it will not have boot ROM or flash. It will boot out of CF. The UART bootstrap feature is used to boot up the first time and configure the CF. CPLD is large enough to have spare pins and logics to support experiments.

This is the pathfinder board. It has multiple ways of booting up and different RAM choices so I can learn and hopefully get smart along the way.

File Attachments

1) TinyZ280_scm.pdf, downloaded 798 times
2) DSC_33230125_F.jpg, downloaded 785 times
3) TinyMon.zip, downloaded 502 times
4) glitchmonHCS.zip, downloaded 486 times

Subject: Re: TinyZZ, a Z280 SBC
Posted by mikemac on Sat, 27 Jan 2018 16:08:42 GMT
View Forum Message <> Reply to Message

plasmo wrote
but I'm more interested in the multiprocessing feature of Z280. Question about memory size: 16meg DRAM is only $1 more expensive than 2 meg DRAM, but can anyone really use 16 meg DRAM?

I think you answered your own question as to why to use the 16MB DRAM: multiprocessing. It's a great way to suck up memory!

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sat, 27 Jan 2018 17:15:37 GMT
View Forum Message <> Reply to Message

I want to try multiprocessing on multiprocessors, so each processor slice may have 2 megabyte of local memory, sharing 2 meg of global memory, and can reach into other processor's 2 meg local memory. With that scheme 16meg of memory can be partitioned into a 2-meg global memory and

7 local processor memories. This may be a better use of the 16 meg space. Another reason why each processor slice needs to be cheap.

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Sat, 27 Jan 2018 20:55:14 GMT
View Forum Message <> Reply to Message

The Z280 fully supports multiprocessor configurations, and that would be a really cool thing to try out. I haven't tried it, but it's documented in Chapter 10 of the Zilog manual. Figures 10-4 and 10-5 give block diagrams, and the text discusses the global bus concept using the /GREQ and /GACK signals. Now an SMP Z280 system; that would be cool.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sat, 27 Jan 2018 23:47:17 GMT
View Forum Message <> Reply to Message

I need to peruse Fritz's Z280 archive to find out more about the arbitration timing. Chapter 10 is an introduction to multiprocessor architecture. How does it handle instruction rerun, test-and-set, arbitration deadlock are the few details I need to have.

One step at a time, get it to boot off CF is the next step...

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Mon, 29 Jan 2018 20:40:24 GMT
View Forum Message <> Reply to Message

One minor note: the UART in the Z280 seems to run more stably, especially at higher baud rates, with a 12.288MHz clock (on the CPU280, we use the Z280's built-in oscillator and place a 24.576MHz crystal). The divisors for the standard baud rates are better. I think any multiple of 2.4576MHz should work well.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Mon, 29 Jan 2018 22:17:36 GMT
View Forum Message <> Reply to Message

I have a 1/2 size programmable divisor clock (ECS-300) underneath the full size 24MHz oscillator. Right now it is programmed to generate 921KHz which is divided 16 to get 57.6K clock for the UART. What I plan to do is divide the 24MHz clock by 13 to get 1.846MHz which is pretty close to 115.2K X16.

I'm booting off UART quite reliably and testing the CF interface right now. This is what the board looks like now.

File Attachments

1) TinyZ280+CF.jpg, downloaded 2342 times

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Tue, 30 Jan 2018 04:54:38 GMT
View Forum Message <> Reply to Message

You guys probably knew all about it, but I just find this cool new Z280 instruction that is a perfectly fit for CF read/write:

```
 LD HL,1000h ; write CF data out to memory starting at 1000h
 LD C,CFdata ; reg C points to CF data reg
 LD B,0h  ; count 256 16-bit data
getCFdat:
 INIW  ; input 16-bit word from port (C) to (HL), increment HL, decrement B
 JP,NZ,getCFdat
```

One instruction (INIW) moves words from CF data register into memory pointed by HL, auto-incrementing. Loop it 256 times running in the cache and moving a sector worth of data. Bang, it is done. Cool!

Subject: Re: TinyZZ, a Z280 SBC
Posted by Andrew B on Tue, 30 Jan 2018 05:27:54 GMT
View Forum Message <> Reply to Message

That instruction (well, an 8-bit version of it) is on the Z80 as well. I was just reading up on it the other day.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Tue, 30 Jan 2018 12:50:11 GMT
View Forum Message <> Reply to Message

Ha! You are absolutely right. The instruction was there all along. Z280 extends the instruction to word wide which is perfect for 16-bit wide CF interface. In fact, there is a better instruction, INIRW, which repeats until regB is zero, so I don't even need the looping JP NZ,xxx instruction. Very Cool.

Subject: Re: TinyZZ, a Z280 SBC

Posted by plasmo on Tue, 30 Jan 2018 14:57:16 GMT
View Forum Message <> Reply to Message

I believe the CF interface is working. I can read/write/verify CF without problems. The UART bootstrapping is rock solid. So I'm at a crossroad: I can continue down the CF bootstrapping and replacing static RAM with SIMM72 DRAM, or I can port CP/M to this board. I'm leaning toward porting CP/M because it may test the board more thoroughly and also let me try Hector Peraza (hperaza) Z280 assembler (https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp ;amp ;th=93&goto=3700&#msg_3700). Right now I'm hand assembling the few Z280 extended instructions I needed.

If CP/M, which CP/M? I am following the long discussion on "Installing CP/M3 (Plus)..." on vcfed. This morning's post is #475 and it is not done yet! CP/M 3 sounds very intimidating. CP/M 2.2 seems more reasonable, is it? I'm downloading various manuals and binaries from cpm.z80.de, I assume that's the place to get code & manuals. Is "CPM_2_0_System_Alteration_Guide" the first manual I should be reading?

Subject: Re: TinyZZ, a Z280 SBC
Posted by rhkoolstar on Tue, 30 Jan 2018 17:09:23 GMT
View Forum Message <> Reply to Message

Hi Plasmo,

CP/M 3 is not intimidating. All you need is the CP/M plus system guide.(attached)

Also I would suggest checking my implementations on Z80, FPGA and Atmel located on my builder pages. https://www.retrobrewcomputers.org/doku.php?id=builderpages: rhkoolstar:start
All sources and building instructions are available. Always nice to start with a working system.

It is actually a quite clean modular system. Nothing much to it.

Rienk

File Attachments
1) cpm3-sys.pdf, downloaded 526 times

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Tue, 30 Jan 2018 17:14:23 GMT
View Forum Message <> Reply to Message

The other advantage to doing CP/M 3 is that the port is already done for CP/M 3 for the CPU280. So that source code is available for comparison, even if the comments are mostly in German.

Subject: Re: TinyZZ, a Z280 SBC
Posted by Andrew B on Tue, 30 Jan 2018 18:30:28 GMT
View Forum Message <> Reply to Message

IF I were you, I think I'd go with CP/M 3. The documentation is very good. You can build an un-banked version that is not much more complicated than CP/M 2.2 to start with, then upgrade it to banked later.

On John M's S100computers.com site he has a complete (Z80-based) CP/M 3 BIOS implementation example. With that plus the manuals I was able to start modifying IDE and console I/O code in short order.

Subject: Re: TinyZZ, a Z280 SBC
Posted by rhkoolstar on Tue, 30 Jan 2018 20:04:08 GMT
View Forum Message <> Reply to Message

Actually, the banked version is not that hard. You basically only need to write the 'move' module, containing the move, xmove and bank routines.
My move.asm is only 50 lines long, including white space and comments.

That and you need to select the correct dseg and cseg segments for the code. The documentation is very clear on this.

Rienk

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Wed, 31 Jan 2018 01:43:02 GMT
View Forum Message <> Reply to Message

It looks like CP/M 3 is the winner of the People's Choice.   OK, I'll give it a try, Thanks.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 01 Feb 2018 02:00:53 GMT
View Forum Message <> Reply to Message

Rienk,
As I was reading your builder page, (https://www.retrobrewcomputers.org/doku.php?id=builderpages: rhkoolstar:start) I noticed at the end you mentioned "Following the FTDI 232 soap I found the following units based on the CP2102 chip to be cheaper and better"... So since I also had difficulties with FTDI 232R adapters lately, can you tell me your experiences?

Subject: Re: TinyZZ, a Z280 SBC
Posted by rhkoolstar on Thu, 01 Feb 2018 07:49:50 GMT
View Forum Message <> Reply to Message

FTDI was annoyed by the counterfeit chips that were around in a large volume (justified). But they decided to 'kill' these counterfeits with the next windows driver update, which was way over the top in my opinion. They tuned it down a bit, and now they seem to corrupt the data stream without disabling the chips. So I went looking for an alternative, and found the CP2102 based converters. I like them, because they are cheaper and they are 3v3 units, but 5v tolerant, so I don't need to check the voltage setting anymore.
I myself don't use windows drivers, so I was/am not affected by these malevolent drivers. All my FTDI, PL and CP chips work as intended.

The CP2102 boards generally come with a USB-A connector attached, so I de-solder the header, connect a 6 wire cable to the board and cover the board with shrink tube, turning it into a USB-plug. the other and of the cable (usually flat cable) is fitted with the appropriate connector, doing away with the clunky converters on the CPU-boards.

Rienk

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Fri, 02 Feb 2018 21:49:15 GMT
View Forum Message <> Reply to Message

I'd also vote for CP/M 3.

As well as being better documented you get a lot of nice features including 512 byte block sizes without BIOS extras and the ability to have big disks (although even with 16 user areas it's a bit difficult to manage without subdirectories!. I'd agree start with the unbanked version then do the banked. There are a few nasties to watch (read carefully the rules about stacks and what you can and cannot bank), also make sure you define the magic set of CP/M offsets in one file and use them in another, otherwise your assembler will be smart, remove them from the relocations and the CP/M builder will fail to do the needed magic relocations.

To get the best use of memory you need to lay out the banked memory buffers by hand which is tedious but without that the tools do a pretty acceptable job.

There are patches to CP/M 3 for Y2K even.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sun, 04 Feb 2018 16:09:25 GMT
View Forum Message <> Reply to Message

fter receiving excellent recommendations to try CP/M 3, I started with CP/M 2.2 instead. Embarrassed This is because I find a file with CP/M2.2 CCP/BDOS sources in Z80 mnemonics on cpm.z80.de. The opportunity of working with source codes instead of binary image is too good to pass up. More importantly, the source codes are in Z80 so I don't have to struggle with 8080 assembly language. Well not entirely true, I still have to figure out the 8080 mnemonics because all the BIOS and utility examples are in 8080. It seems I've spent all my time translating and then figuring out mistakes in my translation! Mad

The advantage of starting with CCP/BDOS source is not have to worry about manipulating CF sector/track, pulling program out of boot track and putting them back, which is the main theme of the System Alteration Guide. Now I can incrementally add BIOS to CCP/BDOS sources and load the code to memory and execute. Wash, rinse & repeat. It is so much easier and I have a better understanding of what's happening when the code crashed.

The thing nagging at me all along was how to get CP/M 2.2 distro into CF in such way that the newly ported CCP/BDOS/BIOS can read and execute it. The solution was unexpected--unexpected for me, you guys probably knew all along: It dawned on me that CP/M 2.2 file format is compatible with CP/M68K 1..3, so I can use gkermit in the CP/M68K environment to transfer CP/M 2.2 distro into an empty drive then move the drive to the Z280 board. Since I used the same disk parameter block values for both CP/M68K and CP/M2.2, Z280 can read the CP/M 2.2 distro and execute them. Ha, that was easy!

I still have many loose ends and Z280 does have weird bugs above and beyond the OUTJMP. Nevertheless, I'm making progress.

Test CP/M TinyZ280
2/1/18

```
a>b:b>dirB: XSUB     COM : SUBMIT   COM : STAT     COM : READ     ME
B: PIP      COM : LOAD     COM : ED       COM : DUMP     COM
B: DUMP     ASM : DSKMAINT COM : DISKDEF  LIB : DEBLOCK  ASM
B: DDT      COM : CPM      SYS : BIOS     ASM : ASM      COM
B: XXZ          : XY           : XZ
b>statA: R/W, Space: 7476k
B: R/W, Space: 7780k


b>stat *.*
 Recs  Bytes  Ext Acc
   64     8k    1 R/W B:ASM.COM
   96    12k    1 R/W B:BIOS.ASM
   60     8k    1 R/W B:CPM.SYS
   38     8k    1 R/W B:DDT.COM
   80    12k    1 R/W B:DEBLOCK.ASM
   49     8k    1 R/W B:DISKDEF.LIB
   18     4k    1 R/W B:DSKMAINT.COM
   33     8k    1 R/W B:DUMP.ASM
    4     4k    1 R/W B:DUMP.COM
   52     8k    1 R/W B:ED.COM
   14     4k    1 R/W B:LOAD.COM
   58     8k    1 R/W B:PIP.COM
    1     4k    1 R/W B:READ.ME
   41     8k    1 R/W B:STAT.COM
   10     4k    1 R/W B:SUBMIT.COM
    6     4k    1 R/W B:XSUB.COM
    1     4k    1 R/W B:XXZ
    1     4k    1 R/W B:XY
   80    12k    1 R/W B:XZ
Bytes Remaining On B: 7780k

b>type xyThe file CPM22-B.ZIP contains the files from the distribution
disk for CP/M 2.2 for a XEROX 1800 system.

b>dump xy
0000 54 68 65 20 66 69 6C 65 20 43 50 4D 32 32 2D 42
0010 2E 5A 49 50 20 63 6F 6E 74 61 69 6E 73 20 74 68
0020 65 20 66 69 6C 65 73 20 66 72 6F 6D 20 74 68 65
0030 20 64 69 73 74 72 69 62 75 74 69 6F 6E 0D 0A 64
0040 69 73 6B 20 66 6F 72 20 43 50 2F 4D 20 32 2E 32
0050 20 66 6F 72 20 61 20 58 45 52 4F 58 20 31 38 30
0060 30 20 73 79 73 74 65 6D 2E 0D 0A 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

b>pip*xx.txt=read.me
ÿ*xx.com=ddt.com
*^C
```

Subject: Re: TinyZZ, a Z280 SBC
Posted by Andrew B on Sun, 04 Feb 2018 16:15:53 GMT
View Forum Message <> Reply to Message

The 'cpmtools' package has the ability to write a custom binary file to the first sectors of the disk images where the loader code is located.

So you can setup your disk parameters there and create bootable images directly from Windows or Linux.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sun, 04 Feb 2018 23:00:44 GMT
View Forum Message <> Reply to Message

Sweet... I've got a CP/M 3 BIOS or two in Z80 instruction format if you want. The SocZ80 BIOS is Z80, as is the CP/M 3 BIOS for the Microbee (which is to the land down under what the Commodore 64 is to the USA)

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Mon, 05 Feb 2018 01:40:44 GMT
View Forum Message <> Reply to Message

Andrew B wrote on Sun, 04 February 2018 09:15The 'cpmtools' package has the ability to write a custom binary file to the first sectors of the disk images where the loader code is located.

So you can setup your disk parameters there and create bootable images directly from Windows or Linux.

I wasn't able to get cpmtools running in Windows Vista to write CF disk in CPM file format. So when I was porting CPM to Tiny68K, I have to make my own utility to copy CPM image from PC to Tiny68K's memory and then to its CF disk. I was wondering how I'm going to do this with Z280. Fortunately there is another easier way to do that.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Mon, 05 Feb 2018 01:41:52 GMT
View Forum Message <> Reply to Message

etchedpixels wrote on Sun, 04 February 2018 16:00Sweet... I've got a CP/M 3 BIOS or two in Z80 instruction format if you want. The SocZ80 BIOS is Z80, as is the CP/M 3 BIOS for the Microbee (which is to the land down under what the Commodore 64 is to the USA)

Yes, I'd love to see BIOS examples in Z80. I like Z80 mnemonics much better.

I understand CPM3 has sector blocking/deblocking build in. That should be much more efficient than my own newbie efforts at blocking/deblocking for CPM 2.2.

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Mon, 05 Feb 2018 14:25:17 GMT
View Forum Message <> Reply to Message

Ok, I need to apologize to Bill. I accidentally edited his message with the CP/M 2.2 results, when I thought I was replying! (I didn't know I had moderator ability in this thread! Oh no!). Andrew B, is there any way to roll back my mistaken edit? I think I have the notification email, and I think it has all the data..... I've got to be more careful before my coffee kicks in! EDIT: found the email, and will be restoring the message. Is there an emoji for 'egg in my face'?

My reply was that I really appreciated the progress that Bill has done; he's making great progress, and I applaud his tenacity in getting this done; after all, a hobby is doing what you enjoy doing, and if a CP/M 2.2 port is what you enjoy, then by all means go for it....

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Mon, 05 Feb 2018 16:00:48 GMT
View Forum Message <> Reply to Message

The trouble with doing electronics as hobby is that it is an undisciplined random walk. There may be a hazy goal somewhere out there, but the real fun is just walking around and picking up whatever interesting along the way. CP/M 3 sounds good and is where I want to be, but CP/M2.2 is just laying there looking interesting. My motivation for trying CP/M at this stage is to see how well my Z280 board can execute more complex programs. I found there are some "weirdness", but I don't know whether it is my hardware, my newbie Z80 software skill, or Z280 itself. I'm shipping your board this morning and start on documentation. Hopefully you can also try different software on it.

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Mon, 05 Feb 2018 16:21:28 GMT
View Forum Message <> Reply to Message

I liken owning a Z280 to owning an MG. You know there are quirks, and lots of maintenance, but it sure is a fun rig to drive (I don't own one, but I have driven one, just a long time ago and it was a friend's). If you enjoy tinkering it can be very fun.

And hobbies are about having fun, and gaining fulfilment, at least to me. I tend to jump around a lot as well, which I'm sure can be a bit frustrating to others at times... Anyway, looking forward to checking out your board.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Tue, 06 Feb 2018 06:15:11 GMT
View Forum Message <> Reply to Message

I started a builder's page about TinyZ280 here:
 https://www.retrobrewcomputers.org/doku.php?id=builderpages: plasmo:tinyz280

Some of the information are filled in (schematic, gerber files), but most are place holders for software, construction notes, engineering changes and CPLD equations.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 08 Feb 2018 15:24:55 GMT
View Forum Message <> Reply to Message

I completed my builder page on TinyZ280 for now. It only cover Step 1, UART bootstrap. I updated the TinyLoad routine as well as cpm2.2 CCP/BDOS/BIOS. I've made some outrage shortcuts with CP/M 2.2 that will certainly met with disapproval from the CPM veterans, but I just want to check out the hardware and I'm pretty happy with how well CPM2.2 is running. So I'm leaving CPM for now and moving on to next phase of the Z280 hardware development.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sun, 11 Feb 2018 14:05:05 GMT
View Forum Message <> Reply to Message

I modified the CPLD code so Z280 can boot out of compact flash. The cold bootstrap code is small, 128 bytes, that copies a CF loader into memory and execute. My question is where to put the cold bootstrap code. The easiest location is in CF's first sector (track 0, sector 1). But there are so many different CF formats out there, I don't know what kind conflicts will result from a custom 128-byte data located in the boot sector. I did looked up Master Boot Record format and convinced myself that the Z280 cold bootstrap code is small enough to fit within the MBR structure. What about CP/M disk format? I think normally the CP/M loader is in the first 2 sectors, but is it reasonable to relocate CP/M loader to the 2nd and subsequent sectors of a CF?

Subject: Re: TinyZZ, a Z280 SBC
Posted by rhkoolstar on Sun, 11 Feb 2018 15:05:47 GMT
View Forum Message <> Reply to Message

CP/M reserves the first track(s) of the logical disk as system tracks (CP/M ignores these tracks completely), but the usage of those tracks is not defined. You can easily make a multi stage loader, (cold bootstrap - loader - CP/M system) fit in one CF 'track'

Rienk

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sun, 11 Feb 2018 18:41:09 GMT
View Forum Message <> Reply to Message

For Fuzix I've been using the first sector as the boot/partition table block everywhere and the number of OS's that don't have the boot block there (or object) is about two (AmigaOS being the obvious one). It varies how much space there is in the sector for booting but it's a good place IMHO.

For CP/M as has been said you can lay it out how you like. For CP/M 3 the system track only holds the loader which is a kind of tiny CP/M subset that loads CP/M itself off the file system.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 15 Feb 2018 00:51:22 GMT
View Forum Message <> Reply to Message

I think I have a working prototype of TinyZZ! The boot sector of the CF disk contains a small cold bootstrap code that loads program resides in sector 2 & 3 into memory and execute. The board in the picture has 4 megabyte of DRAM. It can handle 16 megabyte, which is the entire memory space of Z280 but that would be gross overkill, right? Is there even a way of using that much memory other than RAMdisk?

I tilt the board slightly so the SIMM memory is visible. The 4-meg SIMM module contains two HY5118164 which is a 1 meg x 16 DRAM. The actual TinyZZ will not have the SIMM socket. It will have just one HY5118164 soldered directly to the board giving it 2 megabyte of memory. It should be a small simple board consists of Z280, CPLD, 24MHz oscillator, 2meg DRAM, and CF adapter. Power consumption is around 350mA at 5V.

Now I'm ready to port CP/M 3 to it.

File Attachments

1) TinyZZ_prototype_F.jpg, downloaded 507 times
2) TinyZ280_boot_from_CF.jpg, downloaded 481 times

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Thu, 15 Feb 2018 18:22:09 GMT
View Forum Message <> Reply to Message

With 2MB of RAM MP/M can support 16 simultaneous users and have lots of spare resources for background jobs. Likewise historically a PDP-11 of similar power would have been considered well loaded with 2MB. Assuming you have 2MB of RAM then that's (assuming one bank for the OS) 31 simultaenous 64K+64K different split I/D processes plus 64K left over, and you'd get even more in that memory if multiple copies of the same program were running. On top of that the hardware supports virtual memory so in a virtual memory environment you'd need 2MB of *useful and current* data in the memory to hit a limit. So divide 12MHz by 30+ running tasks and ask if 1/30th of a processor per task is going to do any useful work, and for virtual memory the number would be even bigger!

So I agree you've got more than enough memory.

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by wsm on Thu, 15 Feb 2018 20:59:43 GMT
View Forum Message <> Reply to Message

I agree that 2MB is more than enough memory for this TINY Z280.

Quote: ... ask if 1/30th of a processor per task is going to do any useful work ...
Not to be argumentative, but this only holds true for heavily compute bound systems. MP/M was primarily designed to allow for multiple users to share all the costly resources, including disks, either through direct connection or via CP/NET. In general, multi-user systems have a LOT of "think" time by the users and/or doing trivial tasks such as recording keystrokes into a buffer. The mainframe time-sharing systems of the early 70's could support hundreds of users while only running at a few MHz and 1 or 2MB of memory.

The TinyZZ has minimal serial ports to support multiple users. Without something like a multiplexor, the above question is actually whether two users would overload the system.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Thu, 15 Feb 2018 23:14:08 GMT

View Forum Message <> Reply to Message

For a Z280 you have virtual memory this is not the case. The idle tasks can be swapped to disk with near enough no impact, so you really do need 30 plus active things.

For MP/M a naive implementation of Z280 would use memory for idle tasks, but even so if you have the maximum 16 users, each of them would have to be running a program on their terminal (maybe idling) *and* have another background task loaded to run out of memory. However the Z280 can do virtual memory transparently. You can put virtual memory *under* MP/M if you are crazy enough (although I seem to remember MP/Mjhad a 2MB limit anyway) and you are back to needing 30 active things to do.

The mainframe world works a bit differently. Not only do you have virtual memory (at some level) you also had very fast I/O subsystems that didn't interfere with the processor and also offloaded most of the work. That dramatically increases the amount of work you can do on a low speed processor. Some setups scaled to a lot of 'users' because they all ran the same application (or the OS was the application) others because input was form based (a smart terminal did all the thinking until you hit send) and almost nobody did much (airline booking for example). You also have a queueing theory problem. The longer it takes your job to run the more other people have finished thinking and started doing, the more they do that they more other people are no longer thinking and it collapses in a heap (usually they day before assignment submissions).

There is a world of difference between doing a CPU driven loop to copy bytes from the CF card and handling each character to send or receive versus the mainframe world'disk X put this list of blocks over there and ping me when you are half way down the work queue' / 'a user has completed filling in a form page and here it is'. The mainframe would be simultaneously swapping to disk, writing out cached data, reading in blocks it guessed will be needed and swapping in processes it wanted to run soon pretty much without CPU overhead. Meanwhile MP/M sucked when someone was copying disks or formatting them.

I've used a four CPU Honeywell L66 and with 60 students on it wasn't a pleasant experience.

There were reasons that towards the end people started to make 'multiuser' Z80 boxes with a single shared hard disk, master CPU and MP/M and a 64K CP/N card for each user linked by a "fast" message link for the network.

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by wsm on Fri, 16 Feb 2018 03:00:01 GMT
View Forum Message <> Reply to Message

I find it interesting how different people's experience gives them different perspectives and I try to learn from that. For reference, besides playing with various micros, I was a systems programmer mostly in the IBM world with S/360-67, S/370's, 4300's & 9370's running various operating systems such as MTS, MFT, MVT, VS1 and MVS both for IT purposes and for real-time process control. My current micro development activities are with 33MHz Z180's using a variety of I/O including RAM disks, flash disks, IDE, SATA, USB etc. while trying to maximize performance.

I think one of our differences in approach is the reference to "CPU driven loop to copy bytes from the CF card". All my current development efforts are based on DMA for "disk" I/O and use multiple LRU buffers. Likewise, byte I/O is also fully interrupt driven with FIFO queues. The one thing that could improve this I/O model would be interleaved memory but discrete dual-ported and quad-ported RAM is both expensive and power hungry. Instead, I just use 10ns 16-bit RAM and CPLD-based DMA controllers. I've already designed my next generation board with an FPGA so it can use embedded DMA with embedded FIFO queues to minimize overhead while maximizing the useage of bus bandwidth.

Virtual memory certainly is very useful but as pointed out, swapping can lead to thrashing if the system isn't balanced. Similarly, if an MP/M system "sucked" when copying disks then I would suspect there is an issue with the scheduler, load balancing and possibly the I/O controller itself. Queueing theory also has a corollary that if you make response time too fast then the users will come to expect it and consequently introduce even more load.

Having worked with low-level 3270 "smart?" terminal data streams, I agree that they do reduce the number of interrupts but there is still a LOT of decoding and formatting that went on in the mainframe before the data was useable by the underlying application. FWIW - My S100 system uses a VDB-8024 display card and a Jade DD floppy controller both with embedded Z80's to offload some of the I/O processing.

As to the dedicated CPU per MP/M user: Besides the polled I/O issues, I believe a lot of that was also due to the high price of RAM in that era, the available instruction set and also some of the application languages. The Z80 can be VERY inefficient for floating point and in my experience there is about a 2.5 times performance difference between interpreted and compiled Basic.

Don't get me wrong, I do understand how multiple CPU's can enhance performance and I've already done basic testing on my own design using up to 16 Z180's at 33MHz connected to a base I/O processor for a possible total of 567MHz. Each of these slave processors is connected to the base processor via a 4KB dual-ported SRAM and there are interrupts in both directions.

With all that being said and back on track, I think the TinyZZ is a GREAT project to enable people to start playing with virtual memory, multi-user, etc. etc. for minimal expense. Kudos to Bill for developing it!!!

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Fri, 16 Feb 2018 04:23:18 GMT
View Forum Message <> Reply to Message

Indeed different experiences led to different approaches. My background in the 80's & 90's was embedded hard real-time navigation and guidance avionic computers. Beside the deterministic computation constraint, the others less hard constraint were smallness, low power & ruggedness. The multiprocessors (up to 15) were tightly coupled, shared memory design with no commercial operating system, just homebrew scheduler. Cost was not a constraint, but every piece of components must justify its place in weight, power & space. I've moved on to other less computational constrained but cost sensitive embedded computers later in my career, and the spartan mindset always stayed with me. It is not about what you designed in--it is what you designed out.

wsm wrote on Thu, 15 February 2018 20:00
With all that being said and back on track, I think the TinyZZ is a GREAT project to enable people to start playing with virtual memory, multi-user, etc. etc. for minimal expense. Kudos to Bill for developing it!!!
Thank you! Z280 really is an interesting processor with many modern features. I have not spent a lot of time with it, but I have not found serious problems so far. Most (all? of the weirdness were my own faults. The internal peripherals seem to work well. The external memory & I/O are running full bus speed, zero wait and appear to work without "weirdness". I understand the DMA may be buggy and I've only used DMA for UART so far. There maybe surprises waiting for me out there still.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Fri, 16 Feb 2018 14:39:46 GMT
View Forum Message <> Reply to Message

The MP/M system sucked when copying disks because it didn't have DMA, so it had to disable interrupts for each disk read/write. It's funny how things work out on these little systems. Will made Fuzix mostly poll the serial port when transmitting - because at higher baud rates we use today (ie not 300 baud) it actually cost more CPU cycles to process the transmit ready via interrupt than to just spin. Likewise a friend who had only of the early Unixes for PC (for 8088/6) discovered it used polling disk I/O via the BIOS firmware. He wrote his own custom interrupt and DMA driven XT hard disk driver - and it made no difference to performance because the DMA controller pretty much stopped the CPU getting to the memory. In fact on the PC/AT IBM amd clones mostly removed the hard disk interrupt.

The real masters of this were Apple. Even on the earlier Macintosh systems the CPU did all the floppy disk work. The localtalk connection ran at 230Kbits with nothing fancy UART wise. All of it was very clever design and a bit of smoke and mirrors. When in tight polling CPU required situations it carefully used what tiny bit of CPU was spare to make sure the most pointer still moved. So long as the mouse pointer moved users were happy. Localtalk likewise used a 3 byte header message to access the bus. Why 3 bytes - because the uart had a 3 byte FIFO. At 230Kbits the message had arrived before the 68000 actually woke up to the fact and looked, but that was fine - the header was followed by a short time delay so the CPU could realise it owned the next message and poll the UART for the data block.

On FP I think the big problem was a lot of CP/M stuff was 8080. Z80 can do FPU pretty passably but you need to be using both register sets and index registers to get the best results. 8080 FPU is just - painful. SDCC does single precision IEEE float pretty well on Z80 which is good enough for me.

But I digress

Tiny ZZ really interests me because it's a virtual memory based machine. Not just toy memory management but properly implemented. The Z800 was designed to run things like Unix. That makes it like the 80286 a very interesting target when playing with operating systems. Two serial ports should be fine as one of them can run TCP/IP especially given the DMA on the uart.

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Fri, 16 Feb 2018 17:54:20 GMT
View Forum Message <> Reply to Message

Plasmo, I'm loving this thread.

The Z280 at 12MHz should be at least as fast as a high-end PDP-11 (say an 11/83 or even /93) and might even be encroaching on VAX-11/780 territory. I need to try out dhrystone on the CPU280.

As to memory size, I am actually planning on fully populating 16MB on my board. Fuzix/UZI280 would be interesting here. Alan, as far as time-sharing goes, it was not unusual for a PDP-11/83 to have 10-15 users, and that's with the J-11 clocked at 4.5MHz (crystal is 18MHz, but the microcycle is a quarter of the clock). Even the VAX-11/780 only ran at 5MHz.

Z280 with 16MB is nearly as useful as 68K with 16MB, in my opinion, even though it's not a flat 24-bit address space. In ways it should be more conducive to something like Fuzix since you have a real MMU, even though per-process space is limited to 64K (but so was the PDP-11's, right?). Now, maybe not for TinyZZ, but I could see a NUMA machine with 8 Z280's having say 1MB local each and 8MB global (or some other split, maybe even 2MB each and no gloabl) and run in a similar system-image way as an SGI Altix would. It would just be, to me at least, the ultimate coolness to say 'nah, this is an 8-Z280 box running' and watch people's faces contort trying to figure that out....

Z280 has built in DMA, and it works quite well in the CPU280 CP/M 3 implementation. It would be fun to look at the advantages versus the disadvantages for something like Fuzix. I wonder how the UZI280 kernel handled I/O?

Subject: Re: TinyZZ, a Z280 SBC
Posted by wsm on Fri, 16 Feb 2018 18:31:34 GMT

View Forum Message <> Reply to Message

Different systems with different CPU and baud rates are obviously going to react differently to interrupt versus polled I/O. Part of my reason for developing interrupt driven queued I/O was a plan to upgrade to an eZ80 which is very roughly like a Z80 at 200MHz. The concept is to also use or develop a multi-tasking operating system.

Interrupt driven queued output allows the application to create new data in parallel with I/O operations and feed it to the queue until the queue limit is reached. This is most effective when multi-tasking with slower devices and I do have some development I plan to do with a device that has a maximum rate of only 9600 baud. Interrupt driven queued input allows for an immediate call to the scheduler / dispatcher if data is not available while also allowing the system to continue to receive data when another task is in control of the CPU.

I can't comment on Fuzix or the system beneath it. What I can say is that I run my console at 115,200 baud and a CP/M-2 LIST command of a very large file pushes the data out at the full baud rate while also doing other things like reading the file and processing clock interrupts. One of the benchmark tests I plan to do is to try using 1K-XMODEM with two buffers which will allow simultaneous disk I/O and serial I/O. The comparisons between polled serial I/O, interrupt driven and DMA based should give some interesting data, especially with USB and also in regards to idle time.

As to DMA controllers and their limitations ... I agree that many implementations just turn into bus hogs that really don't do much to improve performance, especially for single user systems. That's why I developed my own version that uses fly-by and a 16-bit data bus. It should be able to transfer data 28 times faster than the 33MHz Z180's internal DMA ... essentially UDMA 6 rate of 133MB/sec. The limitation can then become the I/O device rather than the bus.

Plasmo ... sorry to hijack your thread but there's been some interesting discussion for those less familiar with overall architecture.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Fri, 16 Feb 2018 19:19:04 GMT

View Forum Message <> Reply to Message

wsm wrote on Fri, 16 February 2018 11:31

Plasmo ... sorry to hijack your thread but there's been some interesting discussion for those less familiar with overall architecture.
Not at all. I enjoyed the conversation very much. The discussions about poll/interrupt/DMA are eerily familiar to what were talked about 20+ years ago. The conclusion was different then because in a hard real time world, the prescribed tasks must be completed every time tick without fail. We can't have an alternate bus master like DMA hogging the bus and we don't have much spare throughput for it to hog. Words like load-levelling alternate bus masters were bandy about rather freely.

I'm pleased with the participation of heavy hitters who worked with Z280 before and can look ahead. It help me imaging the road ahead while still feeling the ground immediately below. The TinyZZ is to help me exploring Z280 the first time around, but the core design is cheap and sparce in term of pc board real estate.   There are plenty of room for 16meg SIMM and Ethernet and multiprocessor interface/connector even in the 10cm * 10cm format.

Please carry on. Like lowen said "I'm loving it"!

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Fri, 16 Feb 2018 20:13:28 GMT

View Forum Message <> Reply to Message

While I'm going to do my own run eventually, I found results from Stefan Nitschke, who ran dhrystone 1.0 on the CPU280 under the Unix-like UZI280. Here are his numbers:
Benchmarks testing the HiTech Compiler using Dhrystone 1.0 (12.5 MHz) :

```
 z80 Version   :  847
 z80 Version + z280 Lib  : 1041 (22%)
 z80 Vers + Optim + z280 lib  : 1219 (44%)
```

For comparison, see the table at http://tech-insider.org/unix/research/1986/0219.html

In a nutshell: The CPU280, with Z280 instructions and 'optimizations' scores in the same general neighborhood as a Sun 2/120, a PDP-11/70, a PC/AT 80286/6MHz PC-DOS 3 system, and many other systems that you'd initially think were far 'better' systems. The CPU280 outscores several runs from VAX-11/750 systems, several PDP 11 systems (11/44, 11/73), and any PC/XT-class systems.

So if you think of the Z280 at 12.5MHz as being the rough equivalent in terms of basic processor speed of the 11/83 or a 10-MHz 68K you'd be in the right ballpark.

I would like to see the 50MHz eZ80 benchmarked, personally. I think everyone would be surprized. The TI-84CE graphing calculator uses a 48MHz eZ80; wonder if there's a C compiler for the beast?

I know that dhrystone is the epitome of the fun to be had with artificial benchmarks, but I am still pleasantly surprized at how well the Z280 holds up against bigger iron, like the 11/70.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sat, 17 Feb 2018 13:54:23 GMT
View Forum Message <> Reply to Message

There's a whole Zilog toolchain and other supporting stuff. Start with: https://github.com/CE-Programming/toolchain. Nothing open however except in Z80 mode. There were several eZ80 projects but they all seem to have died due to the lack of open information on tools, programmers etc.

There are FPGA alternatives - the T80 core used in Will's socz80 is clock equivalent to the real silicon but will run at 100MHz+ with fast enough RAM or a cache. There's also the Nexgen core which is a modern design executing Z80 instructions and does a lot more at a lower clock rate.

And then of course there's the rabbit ...

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Sat, 17 Feb 2018 14:37:06 GMT
View Forum Message <> Reply to Message

Thanks for the pointer to the CE C SDK.

Even though I asked about C, the spasm-ng assembler can do eZ80 code. Also, the Y80e core is available that is supposed to be more efficient than the T80 or a-z80; see https://github.com/freecores/y80e

But this is a bit off topic for this thread.

For the Z280, there are, as far as I know, the following toolchains:

John Coffman has done some work on the assembler for sdcc, but I don't know where that work stands Anders Ostlund has done some work on an assembler for sdcc. The UZI280 libraries for HiTech C coupled with the freeware version, on either the CPU280's CP/M 3 or within UZI280. The PRE280 preassembler distributed with the CPU280 code. Source for this is lost, even by its author. Also usable under CP/M 3 (or the zxcc CP/M emulation subsystem from Windows or Linux). Hector Peraza's Z280 assembler.

I feel like I'm forgetting some, and I apologize if I have.

If a Z280 generator and libs could be written for either the z88dk tool chain or the sdcc toolchain, it would be good.

For other compilers, if it can run on a PDP11 it should be able to run on Z280, if you want native development. I would defer to Alan's expertise on which modern cross compiler chain would be more desirable.

For CP/M the BDS C compiler is completely open, and maybe this could be ported and used for Z280 code generation.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sat, 17 Feb 2018 15:41:45 GMT
View Forum Message <> Reply to Message

I'm a Windows user so I use Zilog Developer Studio, ZDS, to write Z80 assembly code for Z280. The tool supports eZ80 with simulator although I have not tried it. It also have hooks for C compiler, but I have not tried it, either. ZDS is free download from Zilog. There is ZDS II as well.

I tried Hector Peraza's Z280 assembler, ZSM4b5, in CP/M 2.2 environment. It assembled the test codes and generated listing and .rel files. That's as far as I got.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sun, 18 Feb 2018 20:59:52 GMT
View Forum Message <> Reply to Message

For cross compilers I think it comes down to SDCC. Even z88dk has moved to SDCC as it's core.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Mon, 19 Feb 2018 00:48:57 GMT
View Forum Message <> Reply to Message

Since there are some interests in having 16 meg of memory, I modified my CPLD equations and inserted a 16 meg SIMM module in the socket. It appears to work. All I know of the Z280 MMU is from what I read in the Preliminary Technical Manual, July 1987. I programmed the MMU based on these rather brief description. I did some double checks to make sure it is actually read/write the 16 meg of memory, but I'm not really confident of my understanding. My memory diagnostic takes 3 minutes to do 16 meg. Is there a Z280 memory diagnostic out there to double check my hardware?

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Tue, 20 Feb 2018 13:16:07 GMT
View Forum Message <> Reply to Message

Played with the 16 meg memory long enough that I'm pretty sure it is working correctly. Another way of checking the RAM is turning it into a RAM disk, so I set up a 8meg RAM disk and was able to read/write/verify files using PIP. All seems to work properly.

Question: I'm still in CP/M 2.2, is there a command to initialize directory like the 'INIT' command in CP/M-68K? Right now I have to externally fill the RAM disk directory area with 0xE5.

Subject: Re: TinyZZ, a Z280 SBC
Posted by wsm on Tue, 20 Feb 2018 14:44:29 GMT
View Forum Message <> Reply to Message

Quote:I'm still in CP/M 2.2, is there a command to initialize directory like the 'INIT' command in CP/M-68K?
Rather than developing a FORMAT command etc., I've found it just as easy to simply do an "ERA d:*.*" during my testing of RAMdisks. That changes the first letter of all the directory name fields to 0E5h.

Subject: Re: TinyZZ, a Z280 SBC
Posted by rhkoolstar on Tue, 20 Feb 2018 18:51:24 GMT
View Forum Message <> Reply to Message

ERA d:*.* only erases the DIR and R/W files in the current user area.
Better write a small program to fill the directory with $E5

'Real' formatting is off course not needed.

Subject: Re: TinyZZ, a Z280 SBC
Posted by rhkoolstar on Thu, 22 Feb 2018 09:40:05 GMT
View Forum Message <> Reply to Message

This is the tool I am using (in Z80 code)
It is probably way more involved than what you need.

It assumes 32 k memory pages, 0x0000-0x7FFF
and 128 directory entries

Maybe you can adapt it to your configuration

File Attachments
1) rdinit.asm, downloaded 448 times

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 22 Feb 2018 14:32:36 GMT
View Forum Message <> Reply to Message

Distraction, distraction and another distraction. This is why this is a hobby: distraction is far more interesting than the goal (which is CPM3 upgrade, I have to remind myself every morning)

#1: Martin Eberhard has released Xmodem 2.7 on comp.os.cpm several months ago. I finally was able to try it on TinyZ280. I can receive file from PC just fine, but at 57600 serial baud rate I think my PC is choking on the incoming serial stream. My terminal software is Tera Term. Even so, I'm happy to have it. This is a critical capability I needed so I can edit/assemble/test on my PC and then download the file to TinyZ280 to run.

#2: Another Martin on comp.os.cpm found an old copy of vi from a book by W. Miller A Software Tools Sampler. He and Udo Munk corroborated and came up with a working vi! I downloaded and ran it on TinyZ280 and it works. The cursor controls are the old 'HJKL' keys. Someone probably will fix that later on. Very cool, I might even entertain writing small program in the native CP/M environment.

#3: Received 10 Z280 from UTSource this week. With that I have enough parts and test software to do margin tests of the design. I tested 13 16-meg SIMM DRAM modules with the 10pcs Z280 over voltage range from 4.8V to 5.3V running RAM diagnostics and CP/M PIP with verify. All 10 Z280 passed, but 3 out of the 13 SIMM modules failed at high voltage. That were curious because parts run faster at higher voltage, but faster parts swinging over greater voltage range also generate more ground noises and ground bounce is always a concern with 2-layer pc board. I added a couple more ground wires to the board and all three DRAM modules passed the tests.

This reminds me of the curious problem with CPU280 where faster PAL part caused problem that was fixed with a slower part. Perhaps Z280 is more susceptible to ground noise. I examined the board layout of the CPU280. It is 2-layer board, but the board was beautifully designed with robust grid-like ground network. I did noticed there is only one 10uF capacitor. DRAM is a real noise maker, so I'd suggest if anyone experienced memory problem with CPU280 to add a couple more 10uF capacitors close to the DRAM array.

----

rhkoolstar, thanks for the program, I'll modify it for my RAM disk which has 512 directory entries and I used Z280's MMU which has 4K size page. Right now my hardware boots to a monitor where I can erase the RAM disk directory before boot to CP/M. Eventually I'll boot straight to CP/M, so rdinit is a good utility to have.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Thu, 22 Feb 2018 15:21:07 GMT
View Forum Message <> Reply to Message

On #2:do you know what licence the editor in the book is under. I was under the impression it was

(C) Prentice Hall and had no particular permissions to re-use/redistribute outside of the book ?

There are a couple of other good small vi editors. Manx distributed one in source form with the Manx compiler kits which is a beauty. Sadly I don't think anyone knows where to even start to see about getting the Manx stuff freed up or I'd have chased it up for Fuzix.

The other very minimal one is 'ae', written for the obfuscated C contest which is public domain but a bit hard to read

https://groups.google.com/forum/#!topic/comp.editors/SID2De1 37wY

A slightly extended (and non obfuscated version) that can pretend to be both vi or emacs-ish (or with a few trivial changes wordstar-ish) is at

https://groups.google.com/forum/#!topic/comp.editors/NMhsctX nQ6I

and is a lesson in minimalist elegance.

It does need a curses library but that wouldn't be hard to fix. On a lot of compilers it's best to change movemap(to use memmove(not loops then the compiler will end up directly or indirectly using LDIR/LDDR.

It's tiny in space usage, and the extremist could turn it into hand coded Z280 fairly easily I am sure

---

Subject: Re: TinyZZ, a Z280 SBC
Posted by wsm on Thu, 22 Feb 2018 15:55:53 GMT
View Forum Message <> Reply to Message

Plasmo - Since you have a CPLD and possibly a spare macrocell, you might want to think about a different approach to RAMdisk formatting. In my CPLD I have an I/O accessible flip-flop that is reset at power-on and is set on the trailing edge of a read. The reset signal / pushbutton does not affect it. This flip-flop is read within the cold boot loader and if it's in the reset state the loader sets the entire RAMdisk directory (16K in your case) to E5's using DMA propagation. LDIR would also work. Thus the RAMdisk directory is formatted once and only once at power-on and it's contents remain useable across multiple resets.

Since I don't use USER areas on the RAMdisk, a simple ERA *.* can be used to essentially reformat it if it becomes corrupted during testing and I've never required a power-off/on to accomplish that.

Subject: Re: TinyZZ, a Z280 SBC

Posted by plasmo on Thu, 22 Feb 2018 16:35:55 GMT
View Forum Message <> Reply to Message

etchedpixels wrote on Thu, 22 February 2018 08:21On #2:do you know what licence the editor in the book is under. I was under the impression it was (C) Prentice Hall and had no particular permissions to re-use/redistribute outside of the book ?

Udo's github license page says it is "This is free and unencumbered software released into the public domain."
https://github.com/udo-munk/s/blob/s-cpm/LICENSE

"obfuscated C contest"! , there is such a thing then, not some software guys just want to see us hardware guys squirm.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 22 Feb 2018 16:37:53 GMT
View Forum Message <> Reply to Message

wsm wrote on Thu, 22 February 2018 08:55Plasmo - Since you have a CPLD and possibly a spare macrocell, you might want to think about a different approach to RAMdisk formatting. In my CPLD I have an I/O accessible flip-flop that is reset at power-on and is set on the trailing edge of a read. The reset signal / pushbutton does not affect it. This flip-flop is read within the cold boot loader and if it's in the reset state the loader sets the entire RAMdisk directory (16K in your case) to E5's using DMA propagation. LDIR would also work. Thus the RAMdisk directory is formatted once and only once at power-on and it's contents remain useable across multiple resets.

Since I don't use USER areas on the RAMdisk, a simple ERA *.* can be used to essentially reformat it if it becomes corrupted during testing and I've never required a power-off/on to accomplish that.

wsm, thanks for the suggestion. I do have 25% spare logic but the voltage supervisor reset is wired to same input as the push button reset. I do have a separate NMI button, so that can perform the warm boot function where RAM disk directories are not wiped.

Subject: Re: TinyZZ, a Z280 SBC
Posted by wsm on Thu, 22 Feb 2018 17:15:40 GMT
View Forum Message <> Reply to Message

Plasmo - I haven't used the Max7000 chips but it would appear that the Assignment Editor allows you to assign a Power-up Level of Low to a register. Thus the required logic is only on the Set line of a S-R flip-flop for the read trailing edge and the flip-flop's Reset input can be held inactive.

I use this technique on Xilinx chips which have a similar feature. I was leery of using this at first since I always try to write logic that forces registers to a known predictable state during reset.

However after testing this feature on a half dozen boards I've never had a problem and have gained confidence with it.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 22 Feb 2018 17:40:59 GMT
View Forum Message <> Reply to Message

Ah, I see what you meant. I've never used that feature although FPGA and some CPLD do need to know whether voltage level is OK before they load the equations from internal flash (CPLD) or external flash (FPGA). So perhaps the initial state of flip flops are trustworthy. I'll look into that for sure.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Thu, 22 Feb 2018 20:58:49 GMT
View Forum Message <> Reply to Message

I shall have to ask Udo. It's a small world Udo also wrote the Z80 emulator I use for most of the Fuzix development and testing, and later it turned out he worked on Coherent (a commercial Unix clone) much of which was later opened up and some commands from it are used in Fuzix, and also most of the Z80 assembler I use witin Fuzix for native assembly

The obfuscated C contest is one of the great traditions. People squash amazing things into unreadable code including things like a basic interpreter, chess, and my favourite of all time - a PC emulator

https://ioccc.org/2013/cable3/hint.html

The other contest I love is the 10 lines of basic contest:
http://gkanold.wixsite.com/homeputerium/basic-10liners-2017

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Fri, 02 Mar 2018 06:02:14 GMT
View Forum Message <> Reply to Message

I've successfully ported the non-banked version of CPM3 to TinyZ280. In the process I also removed all "workaround" for Z280 "bugs", specifically the extra codes to fix "OUTJMP" bug are all removed and all is working well. The only bug I know of is that refresh register can not be turned off with a value of 0x0. I'm using the refresh register for the DRAM so it is not turned off anyway. I exercise the MMU to do RAMdisk and memory diagnostic and the MMU is performing as expected. I used the DMA to do UART bootstrapping, but I have not used it for memory-to-memory transfer.

I'm beginning to question Z280's reputation for being buggy. Certainly I've encountered problems during the hardware & software developments, but they are all bugs on my part, not the Z280. Z280 is more susceptible to ground noise when interfacing to DRAM, but a couple extra ground wires took care of that. One thing that may be different about my board is that it has zero wait state access to all devices such as memory and compact flash. I believe OUTJMP problem showed up when I/O access has wait states inserted.

Does anyone has specific sample codes that'll cause Z280 to behave abnormally? I like to try them on my setup.

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Fri, 02 Mar 2018 16:43:53 GMT
View Forum Message <> Reply to Message

Quote:I'm beginning to question Z280's reputation for being buggy.

I'm beginning to wonder myself. I have relied on and deferred to Tilmann Reh's experiences along with information from several users of the Z280, but that information is almost all pre-92, and I've seen datecodes now as late as 97. So I guess it's possible that some bugs actually were fixed, but since the bugs weren't documented neither were the fixes. I've not disassembled the Specialix firmware for their SI/XIO ISA card from the Linux kernel drivers; that might be interesting to see if any of the workarounds were used.

I know I'm questioning the quantity manufactured, or the wisdom that the Z280 never really was commercially successful. I mean, there are ten years' worth of production (according to datecodes) out there, and UTSource is claiming 30,000+ stock on chips. And every order I've placed has been filled, quickly, so they apparently have stock of some size to do that.

The CPU280 does some pretty significant wait state generation for I/O, but that's primarily because of the ECB interfacing.

I'm loving where you're taking this beast, Plasmo!

Subject: Re: TinyZZ, a Z280 SBC
Posted by tor on Fri, 02 Mar 2018 18:24:50 GMT
View Forum Message <> Reply to Message

This is getting more and more interesting. What I heard about the Z280 in the past was about a somewhat buggy processor that wasn't around anymore. Now it seems to be alive and well (not problematically buggy), and with 30000 chips in stock? I'm definitely taking notice. Memory management and CP/M (3).. this sounds very interesting.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sun, 04 Mar 2018 16:09:20 GMT
View Forum Message <> Reply to Message

lowen wrote on Fri, 02 March 2018 09:43Quote:I'm beginning to question Z280's reputation for being buggy.

I'm beginning to wonder myself. I have relied on and deferred to Tilmann Reh's experiences along with information from several users of the Z280, but that information is almost all pre-92, and I've seen datecodes now as late as 97. So I guess it's possible that some bugs actually were fixed, but since the bugs weren't documented neither were the fixes. I've not disassembled the Specialix firmware for their SI/XIO ISA card from the Linux kernel drivers; that might be interesting to see if any of the workarounds were used.

The 15pcs of Z280 I have are from 1992 to 1997 so that's consistent with what you have. The earlier Z280 may have bugs, but Zilog had a number of years to correct that and it is hard to imagine they'll leave the more significant bugs as they were during all 10 years of production.

I re-read Tilmann Reh's description of Z280 bugs (section 10 of CPU280 Software Manual):
* OUTJMP bug is associated with I/O-write with wait state. This applies to both CPU access as well as DMA access. I can confirm that without wait state in I/O-write, the CPU access is not subject to the OUTJMP bug. I have not tried I/O write with DMA so far.
* DIVUW gives wrong results.
* DMA may not release the bus after I/O write for up to 20uS. There is a software fix named "Stefan Nitschke Chaos" for that.
* CPU may have invalid carry flag when instruction is sandwiched between two EX AF,AF' instructions. That error is also influenced by whether cache is enabled or not.

I'm ready to turn on DMA for CF and RAM disk data transfer so I will soon find out whether OUTJMP bug applies to DMA transfer of zero-wait I/O and whether DMA hangs on to the bus for up to 20uS. Any one know what is "Stefan Nitschke Chaos"? I imagine that code is implemented in UZI280?
I can understand the race of flag register updating its condition while it is been swapped out. Running with cache enabled will aggrevate that race condition even more. So turning off cache is a simple & quick test when encountering strange results.

I have not look into CPU280's BIOS nor UZI280. I imagine the specific bug fixes were documented in the codes. It may be interesting to remove these bug fixes and see if CPU280 with later date code Z280 still work or not. It is also interesting to port UZI280 and CPU280's BIOS to TinyZ280 with or without the bug fixes and see what happened.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Tue, 06 Mar 2018 21:29:39 GMT
View Forum Message <> Reply to Message

I have the official G step errata list (1987)

It lists only two known bugs but I'm not clear if those are just 'new' ones or all the others got fixed and also Tilman claims Zilog didn't even acknowledge some of those he found.

1. A cache corruption bug in Z80 mode

2. Burst mode does not work in X2 and X4 bus clock

H step was released late 1987 early 1988 some time

It would be interesting to know if DIVUW works correctly on your CPU.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Wed, 07 Mar 2018 03:13:56 GMT
View Forum Message <> Reply to Message

This was what Tilmann Reh said about DIVUW bug:

The DIVUW gives wrong results in certain cases. Such a case is the MSB (bit15) of the divisor being set.

So I tried divide 0x20000000 by 0x8888 and get 0x3c00 quotient and 0x2000 remainder. Try 0x20000000 divide by 0x4888 get 0x70F1 quotient and 0x37F8 remainder. These are the correct answers according to my calculator. I tried three different addressing modes: divuw[dehl],(addr), divuw[dehl],bc, and divuw[dehl],ix, all give the same answers.

Fritz had got Tilmann's board and has picture of it here:
https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp
;th=93&goto=4162&#msg_4162
The date code of Tilmann's Z280 is 8735. The date code of my Z280 that I just did the divuw tests is 9332. So perhaps the divuw bug is fixed since mid 1987.

I remember someone had said somewhere that 'step J' has the bugs fixed. I don't know how to identify the step. The parts I have all have a 2-letter designation on the same line as the date code, but all 15 Z280 have different 2-letter designations.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Wed, 07 Mar 2018 12:23:53 GMT
View Forum Message <> Reply to Message

If you know a given divuw fails on Tillman's board then to an extent you do know how to identify the stepping I've not seen any way to tell Z280 steps apart that is documented, nor any documented register that exposes it. The Z80 is similar - you can pull tricks to tell NMOS from

CMOS but that is about it.

If you think the Z280 is buggy you should really read the 80286 errata, older versions of that were dire, after that the original 386 couldn't multiply, the pentium couldn't divide

---

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Wed, 07 Mar 2018 14:18:34 GMT
View Forum Message <> Reply to Message

What I had thought about doing was writing a small routine that was the equivalent of DIVUW but using only Z80 instructions and then iteratively comparing the results of that routine with the results of DIVUW and see if there was a pattern. Work has me way too busy right now to be able to concentrate on it, though.

I'd love to see the equivalent of ZEXDOC and ZEXALL for Z280, but it will be a while before I will have time to try my hand at writing it.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Wed, 07 Mar 2018 16:05:46 GMT
View Forum Message <> Reply to Message

For the 680x0 family Motorola would designate the parts with 'XC' prefix before going full production and there usually is a cryptic line of text indicating the mask level and production facility. I guess date code on Z280 is all we have to go on.

Anyone who has a pre-1988 Z280, I'll trade you with a post-1992 Z280.

Never heard of ZEXDOC & ZEXALL until now. It is amazing how much Z80 knowledge are out there.

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Wed, 07 Mar 2018 16:13:15 GMT
View Forum Message <> Reply to Message

One of the best sources of information on Z80 is Rodnay Zaks' book. Rodnay has given permission for a PDF to be posted online, and you can grab it from the links at the end of this page. For that matter, the whole 'www.z80.info' site is a gold mine, but it needs more Z280, Z380, and eZ80 information. (Z380 and eZ80 are current production, but since both are only available in surface-mount packages, they haven't been as popular with hobbyists)

Subject: Re: TinyZZ, a Z280 SBC

Posted by etchedpixels on Wed, 07 Mar 2018 20:08:59 GMT
View Forum Message <> Reply to Message

The other problem with them was that Zilog made it near enough impossible to use anything but their own proprietary cable and tools for programming the eZ80. Nowdays its documented and there are tools (eg https://hackaday.io/project/9483-ez80-open-source-programmer

There was an ez80 sbc project years ago (ez80sbc) - see
https://web.archive.org/web/20100604065529/http://www.ez80sb c.com:80/

Subject: Re: TinyZZ, a Z280 SBC
Posted by fritzeflink on Fri, 16 Mar 2018 05:47:34 GMT
View Forum Message <> Reply to Message

plasmo wrote on Wed, 07 March 2018 04:13This was what Tilmann Reh said about DIVUW bug:

The DIVUW gives wrong results in certain cases. Such a case is the MSB (bit15) of the divisor being set.

So I tried divide 0x20000000 by 0x8888 and get 0x3c00 quotient and 0x2000 remainder. Try 0x20000000 divide by 0x4888 get 0x70F1 quotient and 0x37F8 remainder. These are the correct answers according to my calculator. I tried three different addressing modes: divuw[dehl],(addr), divuw[dehl],bc, and divuw[dehl],ix, all give the same answers.

Fritz had got Tilmann's board and has picture of it here:
https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp ;amp ;amp ;th=93&goto=4162&#msg_4162
The date code of Tilmann's Z280 is 8735. The date code of my Z280 that I just did the divuw tests is 9332. So perhaps the divuw bug is fixed since mid 1987.
.

The buggy old Z280 is mine. Very interesting info from you reading just as I can't sleep.
The CPU280 with the old Z280 stop after the ram test. I hope I got help next month from a 'developer' as I'm only a mechanic.

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Fri, 16 Mar 2018 09:40:19 GMT
View Forum Message <> Reply to Message

plasmo wrote on Tue, 30 January 2018 06:57I'm leaning toward porting CP/M because it may test the board more thoroughly and also let me try Hector Peraza (hperaza) Z280 assembler (https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp ;amp ;amp ;th=93&goto=3700&#msg_3700). Right now I'm hand assembling the few Z280 extended instructions I needed.

You can use the Z280 assembler on Unix/Linux with a tool like John Elliott's ZXCC. There is a Windows port available somewhere as well.

The advantage of ZXCC over other CP/M emulators like e.g. z80pack is that with ZXCC you can run any CP/M utility directly from the command line (or from a Makefile), and access files on the host filesystem without having to put everything first in a virtual disk. ZXCC is used, for example, to cross-build CP/M 3 for the CPU280; that's the tool I used as well to develop and test the assembler.

BTW, I love your TinyZ280 board; I just added it to my TODO list . The CPLD is a clever addition that makes it more versatile while keeping the component count down.

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Fri, 16 Mar 2018 10:12:38 GMT
View Forum Message <> Reply to Message

[quote title=plasmo wrote on Sun, 04 March 2018 08:09]lowen wrote on Fri, 02 March 2018 09:43I'm ready to turn on DMA for CF and RAM disk data transfer so I will soon find out whether OUTJMP bug applies to DMA transfer of zero-wait I/O and whether DMA hangs on to the bus for up to 20uS. Any one know what is "Stefan Nitschke Chaos"?
It's a section of code inside the DskIO routine, diskio.280 file. Basically, the DMA bug causes the data transfer from the FDC controller to fail with a DMA underrun error. When that happens, the code "fixes" the bug by writing four null words to four consecutive I/O addresses starting at FFxxE8. The mystery being that those addresses do not correspond to anything (documented, at least), that the fix has to be applied only when the bug is triggered, and that it needs to be applied only once (at least according to the code logic). So a second FDC DMA underrun error is supposed to be a floppy or FDC fault. The fix is never applied for hard disk (GIDE) DMA transfers, since IDE data transfers are not time-critical and there is no easy way to tell if the DMA bug happened in that case.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Fri, 16 Mar 2018 12:59:47 GMT
View Forum Message <> Reply to Message

FFxxE8 is the refresh controller (section 9.3 of Preliminary technical manual). I don't know what the next three I/O address after FFxxE8 are. Writing zero to FFxxE8 will disable the refresh controller but that will cause my board to crash, but I have not find out why exactly. To turn off the refresh controller, I would clear bit 7 (enable bit), but write non-zero values to remaining 7 bits.

Since I do use the refresh counter for the DRAM, it is enabled and programmed to 16uS refresh.

I begin to understand why the unexpected 20uS delay: The refresh counter keeps track of how many refresh cycles it missed when Z280 bus mastership is relinquished to alternate bus masters (such as DMA). When Z280 regained the bus mastership, the refresh counter will take over immediately and make up for the refresh cycles it missed. This could be quite a number of cycles so Z280 appears to stalled during that period. At reset the refresh counter defaults to a refresh every 32 CPU clocks (about 2.6uS) so if DMA takes away 200uS from CPU bus mastership, the refresh counter will take 20uS to make up for the missing refresh cycles. Turning off the refresh counter will stop it from taking over the Z280 bus, but only until next reset.

CPU280 have DRAM as well, so it must have a different way of refreshing the memory and not relying on the internal refresh counter.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Fri, 16 Mar 2018 13:22:10 GMT
View Forum Message <> Reply to Message

---

hperaza wrote on Fri, 16 March 2018 03:40plasmo wrote on Tue, 30 January 2018 06:57I'm leaning toward porting CP/M because it may test the board more thoroughly and also let me try Hector Peraza (hperaza) Z280 assembler (https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp ;amp ;amp ;amp ;th=93&goto=3700&#msg_3700). Right now I'm hand assembling the few Z280 extended instructions I needed.

You can use the Z280 assembler on Unix/Linux with a tool like John Elliott's ZXCC. There is a Windows port available somewhere as well.

The advantage of ZXCC over other CP/M emulators like e.g. z80pack is that with ZXCC you can run any CP/M utility directly from the command line (or from a Makefile), and access files on the host filesystem without having to put everything first in a virtual disk. ZXCC is used, for example, to cross-build CP/M 3 for the CPU280; that's the tool I used as well to develop and test the assembler.

BTW, I love your TinyZ280 board; I just added it to my TODO list . The CPLD is a clever addition that makes it more versatile while keeping the component count down.

Thank you for the pointer to ZXCC. The wonderful thing about the Z80 community is there are so many resources for every development environment and the bad thing about the Z80 community is there are so many resources for every development environment.   Without someone points the way, a newbie like me can get bogged down trying every different tools.

CPLD is the heart of the design. I've learned my lesson from Tiny68K where CPLD is completely used up without any spare pins and logic. Booting from CF is a simpler design that gives me a dozen spare pins and 25% spare logic thus more room to play!

Subject: Re: TinyZZ, a Z280 SBC
Posted by Andrew B on Fri, 16 Mar 2018 23:59:45 GMT
View Forum Message <> Reply to Message

The windows port of zxcc is part of Wayne's RomWBW repository:

https://github.com/wwarthen/RomWBW/tree/master/Tools/zx

A very useful tool indeed.

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Sat, 17 Mar 2018 12:53:27 GMT
View Forum Message <> Reply to Message

plasmo wrote on Fri, 16 March 2018 05:59FFxxE8 is the refresh controller (section 9.3 of Preliminary technical manual). I don't know what the next three I/O address after FFxxE8 are. Writing zero to FFxxE8 will disable the refresh controller but that will cause my board to crash, but I have not find out why exactly. To turn off the refresh controller, I would clear bit 7 (enable bit), but write non-zero values to remaining 7 bits. Since I do use the refresh counter for the DRAM, it is enabled and programmed to 16uS refresh.
Indeed, just consulted the manual and is there. I was quoting the source code comments, and it looks like either the register was not documented at the time, or Stefan Nitschke did not have access to the full documentation. Interestingly, there is an entry for the register in the z280equ.mac file.

Your explanation of the DMA delay makes full sense, and the 'bug' starts looking now more like a design 'feature' than a real bug. And since the workaround was found "by accident", the additional writes to the FFxxE9..EB addresses are very likely superfluous.

Subject: Re: TinyZZ, a Z280 SBC
Posted by fritzeflink on Sat, 17 Mar 2018 13:31:54 GMT
View Forum Message <> Reply to Message

lowen wrote on Wed, 07 March 2018 17:13One of the best sources of information on Z80 is Rodnay Zaks' book. Rodnay has given permission for a PDF to be posted online, and you can grab it from the links at the end of this page. For that matter, the whole 'www.z80.info' site is a gold mine, but it needs more Z280, Z380, and eZ80 information. (Z380 and eZ80 are current production, but since both are only available in surface-mount packages, they haven't been as popular with hobbyists)

For german people you can get Zaks_Programmierung_des_Z80.pdf at

http://oldcomputers.dyndns.org/public/pub/manuals/ and scroll down for download.

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Sat, 17 Mar 2018 14:00:54 GMT
View Forum Message <> Reply to Message

plasmo wrote on Sun, 04 March 2018 08:09* CPU may have invalid carry flag when instruction is sandwiched between two EX AF,AF' instructions. That error is also influenced by whether cache is enabled or not.

You can find the original bug report here.

My CPU (date code 9528) has the bug, and I can reproduce it consistently with any variations of the same sequence, e.g.:

```
scf
ex  af,af'
xor a
ex  af,af'
```

It is not that the carry bit disappears, but that the second ex af,af' instruction copies the flags register instead of swapping it (the accumulators are swapped as expected). For example, after the following sequence:

```
xor a      ;clear sign and carry
ex  af,af'
or  0FFh   ;set sign flag
ex  af,af'
```

both F and F' are identical: Carry and Zero bits clear, Sign and Even parity bits set. The pattern seems to be: one single arithmetic or logical instruction that uses the ALU sandwiched between two ex af,af' instructions. Thus, if the instruction is a load operation, or a rotate, nop, scf, etc. the bug will not be triggered. Neither it will if the "sandwich" consists of two or more instructions. Those are my observations so far.

How can it depend on the specific memory location (for me it happens when the starting address of the sequence is e.g. 100h, 1000h, 2000h, 8000h, etc. but not if it is e.g. 9E75h), or if cache is enabled/disabled is unclear to me. An easy fix is to add at least a nop after the arithmetic instruction, easy to do when developing new software, but difficult to fix for existing applications.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sun, 18 Mar 2018 14:59:05 GMT
View Forum Message <> Reply to Message

The xxx0 requirement doesn't surprise me - presumably it's dependant in part upon the four instructions being in the same DRAM burst (and probably that there is nothing much else midflight when it happens).

So long as there is an address and cache setting it reliably happens at then it's easy enough to test for. It also doesn't look too scary to write an assembler fixup for either.

I am curious if your second examle actually works or whether the requirement is 1 byte instruction

?

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sun, 18 Mar 2018 18:56:39 GMT
View Forum Message <> Reply to Message

I can observe the exaf bug on my machine. Using a test code based on hperaza's 2nd example, my hardware will get identical F, F' if the first ex af,af' instruction is located on an even address. I wrote a program that relocates the test code by one byte every iteration; and the error occurs every other iteration. What's interesting is that to observe the problem, the test code must not be in the cache. If the test code is in the cache, the error will occur every time regardless of the location. To flush the test code out of the cache, I'd call a routine that has more than 256 NOP before returning to the test code again.

One difference between my hardware and Tilmann Reh's CPU280 is that I've not implemented the burst mode memory transaction.

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Mon, 19 Mar 2018 17:09:06 GMT
View Forum Message <> Reply to Message

plasmo wrote on Sun, 18 March 2018 14:56I can observe the exaf bug on my machine. Using a test code based on hperaza's 2nd example, my hardware will get identical F, F' if the first ex af,af' instruction is located on an even address. I wrote a program that relocates the test code by one byte every iteration; and the error occurs every other iteration. What's interesting is that to observe the problem, the test code must not be in the cache. If the test code is in the cache, the error will occur every time regardless of the location. To flush the test code out of the cache, I'd call a routine that has more than 256 NOP before returning to the test code again.

You can also disable the cache for instructions for testing. I would find it interesting to make sure that register A is also being exchanged not copied if the first ex af,af' is on an odd address; if an internal data bus collision is occurring for F and F' on even, I would be suspicious of A and A' on odd. Ken Shirriff's blog has a great discussion of the Z80's register implementation at the silicon level; I would suspect the Z280, even though it's CMOS and not NMOS, might use some of the same features. His blog entry is
http://www.righto.com/2014/10/how-z80s-registers-are-impleme nted-down.html

Quote:
One difference between my hardware and Tilmann Reh's CPU280 is that I've not implemented the burst mode memory transaction.

I was going to ask if you were using burst mode.

It is very nice to see some of these bugs being investigated again, and one (delay after DMA) being put to rest as 'not a bug' instead. Nice!

Oh, almost forgot:
plasmo wrote on Fri, 16 March 2018 08:59...
CPU280 have DRAM as well, so it must have a different way of refreshing the memory and not relying on the internal refresh counter.

CPU280 doesn't use the refresh counter as far as I know, but uses CAS-before-RAS refresh. This isn't well documented in the English manual; the German Hardware Handbuch shows the timing diagrams on page 22 (PDF page number).

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Mon, 19 Mar 2018 21:57:18 GMT
View Forum Message <> Reply to Message

My test shows that when errors occurred, only flag registers are incorrect. Reg A and reg A' retain their correct values in all cases.

The timing diagrams are quite useful, but now I'm confused. The CAS-before-RAS refresh cycle is triggered by the RFSH signal which is generated by decoding Z280's status lines ST[3..0] for value of 0001. That status value can only be generated by the refresh controller. Another word, if the refresh controller is turned off, then RFSH will also turned off and no more CAS-before-RAS refresh cycle. My refresh logic is very similar which also uses CAS-before-RAS refresh cycle and depends on the refresh controller to activate the refresh access once every 16uS.

Subject: Re: TinyZZ, a Z280 SBC
Posted by fritzeflink on Mon, 19 Mar 2018 23:54:40 GMT
View Forum Message <> Reply to Message

lowen wrote on Mon, 19 March 2018 18:09

CPU280 doesn't use the refresh counter as far as I know, but uses CAS-before-RAS refresh. This isn't well documented in the English manual; the German Hardware Handbuch shows the timing diagrams on page 22 (PDF page number).

Hi...

I did a new scan of the papers for the linked z280_reh.pdf including some history pages and will save the pdf at oldcomputers.
Here I post a screen hard copy of page 17 - timing CPU280, 12.5MHz for information.

File Attachments

1) hc_2955.jpg, downloaded 1351 times

Subject: Re: TinyZZ, a Z280 SBC
Posted by lowen on Tue, 20 Mar 2018 00:12:32 GMT
View Forum Message <> Reply to Message

Yeah, it does sound a bit confusing. The Z280 manual does say that if an external bus transaction doesn't occur within the number of cycles defined in the refresh control register, that a refresh transaction will be inserted anyway, even if refresh is disabled. This means that an idle loop, running entirely in cache, would trigger refresh cycles even with refresh disabled.

But I begin to wonder about the advisability of the snc fix. Tilmann's doc does say that software development had hit a snag due to some of the issues; without digging deeper, I wonder if the cure could have been worse than the disease.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Tue, 20 Mar 2018 01:03:49 GMT
View Forum Message <> Reply to Message

lowen wrote on Mon, 19 March 2018 18:12Yeah, it does sound a bit confusing. The Z280 manual does say that if an external bus transaction doesn't occur within the number of cycles defined in the refresh control register, that a refresh transaction will be inserted anyway, even if refresh is disabled. This means that an idle loop, running entirely in cache, would trigger refresh cycles even with refresh disabled.

This is the last paragraph of section 9.3 Refresh Controller

Thanks for the reminder. I remember reading it but not understanding it and moved on. Didn't worry about it since I turned on the refresh controller anyway. Reading between the lines it basically says the refresh rate register can't have a value of zero (refresh disable and rate = 0). This is why it crashed when I write 0x0 to the refresh rate register. This also tells that SNC (Stefan Nitschke Chaos) can't work because it supposedly wrote 0's to the refresh controller. Or perhaps it did worked with older date code parts, but now it won't work. I wonder whether this is part of the UZI280 problems?

Edit: Thinking about it more, even if SNC had set the rate field to non-zero value it can create insidious bugs because if program is idling in cache, the refresh mechanism would work properly, but if program starts running some tasks then the refresh cycles won't start and parts of the memory will become corrupted gradually. This is insidious because the tasks that run regularly will get refreshed, but the tasks that seldom get called will become corrupted and then crashed in random ways. DRAM can actually retain its data for many seconds without refresh but it is highly variable and strongly temperature dependent--a nightmarish problem to debug.

File Attachments

1) Refresh_Z280_MPU_(noocr_bw_400).jpg, downloaded 1332 times

---

Subject: Re: TinyZZ, a Z280 SBC
Posted by fritzeflink on Tue, 20 Mar 2018 10:03:24 GMT
View Forum Message <> Reply to Message

lowen wrote on Mon, 19 March 2018 18:09
CPU280 doesn't use the refresh counter as far as I know, but uses CAS-before-RAS refresh. This isn't well documented in the English manual; the German Hardware Handbuch shows the timing diagrams on page 22 (PDF page number).

My new scan of the german manual for CPU280 and some history information are at:

1.)
Collection of information by Tilmann Reh send to us including hardwarebook, short
info for the operating system and historie informations. (german text as below)

http://oldcomputers.dyndns.org/public/pub/rechner/zilog/z280/CPU280_Handbuch_und_Historie_(ger_bw_ocr).pdf

2.)
Zusammenstellung der Schreiben von Tilmann Reh - Hardware-Handbuch (210690),
Kurz-Information zum
Betriebssystem (231090) und aeltere Texte.

http://oldcomputers.dyndns.org/public/pub/rechner/zilog/z280/CPU280_Handbuch_und_Historie_(ger_gray_ocr).pdf

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Tue, 20 Mar 2018 14:23:47 GMT
View Forum Message <> Reply to Message

plasmo wrote on Mon, 19 March 2018 18:03I wonder whether this is part of the UZI280 problems?
Is there a list somewhere of UZI280 problems? Haven't had the time to look at the sources in detail, but I don't think it has the SNC fix (floppies are not supported at all). OTOH, OUTJMP fixes can be found in e.g. the IDE code.

Quote:Thinking about it more, even if SNC had set the rate field to non-zero value it can create insidious bugs because if program is idling in cache, the refresh mechanism would work properly, but if program starts running some tasks then the refresh cycles won't start and parts of the memory will become corrupted gradually. This is insidious because the tasks that run regularly will get refreshed, but the tasks that seldom get called will become corrupted and then crashed in random ways. DRAM can actually retain its data for many seconds without refresh but it is highly variable and strongly temperature dependent--a nightmarish problem to debug.
I was wondering about the same. The DRAM on the CPU280 relies on the Z280 CPU for refresh, and after the SNC fix the software cannot longer be expected to work reliably.

Just wanted to do a quick test to see how long it would take before memory contents begin to fade, but since the BIOS enables protected mode I could not access the refresh register. Also, it is not a trivial test, as the memory will get refreshed just by reading it or by instruction fetches, etc.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 22 Mar 2018 12:03:41 GMT
View Forum Message <> Reply to Message

Did a couple experiments with turning off refresh logic to DRAM on TinyZ280. The test condition is at room temperature (70F). I filled 52K of DRAM with test patterns that's unique for each 16-bit word; turned off the refresh controller (the program itself is constantly refreshed by code fetches); wait for some period of time then check the memory for number of mismatches:
DRAM #1 (TI TMS417409)

time without refresh (seconds)	error count
5	no error
10	no error
15	1
20	7
25	10
30	17
35	25

DRAM #2 (NANYA NT511740)

2.5	no error
5	1
10	8
15	21

20	50
25	61

Even at room temperature, DRAM's data retention time is orders of magnitude greater than what manufacturer's datasheet required (32mS). Literature showed that retention time is dramatically increased at colder temperature. Retention is also different for different manufacturers.

So depending on the operating temperature and SIP DRAM manufacturers, idling processor with SNC fix will remain operational indefinitely; processor that's actively running with SNC fix may remain operational for many seconds, possibly minutes depending on where the data corruption had occurred.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sat, 24 Mar 2018 18:15:55 GMT
View Forum Message <> Reply to Message

The DRAM retention time is very dependant upon what else is happening to cause disturbance and leakage. Try repeatedly changing physically close memory sells and see how the time looks. On pure idle though your numbers look irght. Folks used to switch SIMMs and RAM chips between devices live for nefarious purposes and it usually worked OK.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Thu, 05 Apr 2018 03:12:38 GMT
View Forum Message <> Reply to Message

With Spring growing season upon us, I find myself not able to spend much time on the Z280 exploration. Fortunately the hardware development is at a stable point. Other than the EX AF,AF' bug, many other bugs that were rumored did not materialized on my hardware. I went as far as built up all 6 remaining pc boards and checked them out. They all worked, and worked the same. I have a working monitor, CPM2.2 and CPM3 (non banked). I wanted very much to port CPM3 banked version, but that'll have to wait to late Fall. UZI280 is but a dream, I had no idea how even to approach it.

I'm pleased with the simplicity of the hardware (and low cost, too). It consists of 5 active components: Z280, Altera EPM7128 CPLD, SIMM72 memory (either 4Meg or 16 Meg), CF disk and 24MHz oscillator, There is an optional RTC based on DS12887. There are plenty of room for other devices even on a small 10cm*10cm pc board. The CPLD is about 75% utilized with a dozen spare pins.

The design is posted on my builderpages starting with this page:
https://www.retrobrewcomputers.org/doku.php?id=builderpages: plasmo:tinyz280:final_step
I welcome ideas, recommendations and constructive criticisms, but they'll have to wait til the Fall to be realized.

Bill

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Wed, 13 Jun 2018 18:29:14 GMT
View Forum Message <> Reply to Message

I couldn't stay away from electronics and continue to work on Z280 albeit at a lower level of efforts. TinyZ280 had served its purpose as a learning platform. I gave a couple of them away and put the remaining boards on eBay hoping to recoup my development cost but was shocked to find how high people bid them up. With TinyZ280 winding down, I returned to the original idea of TinyZZ where a Z280 SBC with connector can be plugged into solderless breadboard to help me with experimentation. Along the way I discovered a line of Z80 family modules called RC2014. Its simplicity appears to me and its single-in-line bus connector can also plug into a solderless breadboard like what I envisioned for TinyZZ. So I redesigned the TinyZ280 to have a RC2014 bus interface and renamed it Z280RC. Like TinyZ280, Z280RC is a single board computer but with only 2 meg of DRAM and an expansion port that can interface to RC2014 I/O modules. Here are the features of Z280RC:

Z280 CPU running at 14.7MHz with bus speed of 7.37 MHz
2 megabyte of DRAM
RAM-only system, shadow ROM is stored in CF disk and loaded in RAM at power on or reset
Bus connected 16-bit wide CF interface
RTC based on DS1302
Four 8-meg CF drives (A: to D:)
1.5-meg RAM drive (E:)
One internal UART at 115200 baud, odd parity, no handshake
bootstrap from CF disk to ZZMon, a simple monitor
CP/M 2.2
CP/M 3 non-banked
A standalone single-board computer with I/O expansion bus compatible with RC2014 I/O bus

The wiki page for Z280RC is here:
https://www.retrobrewcomputers.org/doku.php?id=builderpages: plasmo:z280rc

The hardware appears to be quite stable and producible even with the 12MHz CPU overclocked to 14.7MHz. I built up 8 boards and 7 of them worked up to 17MHz. One board is flaky at 14.7MHz but stable at 12MHz.

Still have quite a bit of hardware/firmware/software I need to do. I like to turn on the burst mode memory access but my CPLD is low on available logic. A board revision is needed to fix the engineering changes and shuffle the CPLD assignments to free up some logic. I also want to port the banked version of CP/M3 to it. It'll be a busy summer in many ways.

Subject: Re: TinyZZ, a Z280 SBC

Posted by etchedpixels on Fri, 15 Jun 2018 22:44:51 GMT
View Forum Message <> Reply to Message

Only 2MB... but how will anyone fit software in that ! ;-)

Nice - and I like the idea of using the RC2014 - even if it's a bit crude as a bus it has some handy I/O cards.

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Tue, 26 Jun 2018 17:27:19 GMT
View Forum Message <> Reply to Message

I'm more or less done with the documentation of the Z280RC.
https://www.retrobrewcomputers.org/doku.php?id=builderpages: plasmo:z280rc

I still have 4 assembled/tested Z280RC boards for $50 each listed in the board inventory page (toward the bottom of the page):
https://www.retrobrewcomputers.org/doku.php?id=boardinventor y

Bill

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Wed, 20 May 2020 12:50:08 GMT
View Forum Message <> Reply to Message

I think I am still finding more Z280 bugs 8)

This fails even with all the caches off and a cache flush in the middle

```
        ld hl,0   ; a ROM address
        ld a,(hl)
        inc (hl)
        cp (hl)
```

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Thu, 21 May 2020 18:26:29 GMT
View Forum Message <> Reply to Message

etchedpixels wrote on Wed, 20 May 2020 05:50I think I am still finding more Z280 bugs 8)

This fails even with all the caches off and a cache flush in the middle

```
        ld hl,0   ; a ROM address
        ld a,(hl)
        inc (hl)
        cp (hl)
```

Alan

Can't try it on the Z280RC (no ROM) and on the CPU280 can't access the ROM without some code juggling... I suppose the bug is that it behaves like if it was RAM? On RAM addresses it seems to work fine. Z80 or Z280 bus mode?

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Fri, 22 May 2020 18:40:58 GMT
View Forum Message <> Reply to Message

On RAM it works fine. I suspect it is keeping (hl) internally as an optimization and not flushing it. There is other stuff going on too, when I try and do ordinary bank switching weirdness happens (again cache on or off) in Z80 bus mode if I execute code, flip banks, do some I/O into memory in the new bank, flip back and then do stuff. Again caches on or off, prefetch hazards accounted for as far as I can see. Unfortunately I don't see any way to debug it without a fairly fancy logic analyzer.

Subject: Re: TinyZZ, a Z280 SBC
Posted by hperaza on Sat, 23 May 2020 07:29:30 GMT
View Forum Message <> Reply to Message

etchedpixels wrote on Fri, 22 May 2020 11:40On RAM it works fine. I suspect it is keeping (hl) internally as an optimization and not flushing it. There is other stuff going on too, when I try and do ordinary bank switching weirdness happens (again cache on or off) in Z80 bus mode if I execute code, flip banks, do some I/O into memory in the new bank, flip back and then do stuff. Again caches on or off, prefetch hazards accounted for as far as I can see. Unfortunately I don't see any way to debug it without a fairly fancy logic analyzer.

That behavior is actually documented: see page 7-6 of the Z280 Technical Manual, last paragraph near the bottom-right corner of the page, which warns about the CPU pipeline not being flushed after a MMU register is changed. It also has that phrase in parenthesis "(However, no data accesses are pre-fetched.)" where the parenthesis do not make that much sense until you remember that the docs says "Preliminary" (like most Zilog's docs do, BTW).

I've respected that caveat in RSX280, and never ran into any problems.

I'd stay away of the Z80 bus mode as well, as Zilog itself discouraged its use.

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sun, 24 May 2020 00:49:25 GMT
View Forum Message <> Reply to Message

This isn't the Z280 MMU - its plugged into an RC2014 backplane with standard RAM/ROM so it's stuck with Z80 mode. I do have the various bits of pipelining accounted for - on a bank switch I flush the cache and then ret which forces a pipeline stall for data. In fact I can put other stuff there as well and it doesn't help.

I suspect you are right about the Z80bus mode being involved - it took them until the H stepping to even make Z80 bus usable, but my CPU is post 1988 so ought to be ok (and the cache disable doesn't fix it either).

Alan

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sun, 24 May 2020 03:15:02 GMT
View Forum Message <> Reply to Message

Alan,
Could you provide a snippet of Z280 code operating in Z80 mode so I can try it? I do have a fairly good logic analyzer and I can take a close loop at the bus transaction.
Bill

Subject: Re: TinyZZ, a Z280 SBC
Posted by etchedpixels on Sun, 24 May 2020 13:53:45 GMT
View Forum Message <> Reply to Message

For the crash case I don't have a snippet - it's the CP/M 3 loader issuing a disk read the bios pages in the rom (which holds the BIOS), pages out the ROM via a helper in high common memory, does the two inirs, pages the ROM back in and then restores everything and pages the ROM back out, then the CP/M code execution goes nonsensical. It's a fairly long sequence of instructions that would need capturing therefore even if capturing from the completion of the inir I/O transfers.

For the ROM test case it's just

```
ld a,(hl)
inc (hl)
purge
cp (hl)
```

and I am fairly sure you'll see a fetch of the LD, a prefetch of the INC a fetch of (HL) and a prefetch of the CP, and there won't be a second read of (HL). That one is just an annoyance where I need to work out a way to defeat the CPU being clever.

Subject: Re: TinyZZ, a Z280 SBC
Posted by plasmo on Sun, 24 May 2020 18:31:48 GMT
View Forum Message <> Reply to Message

OK, I believe your ROM case is for Z280 operating in Z80 compatible mode, and the MMU and cache are disabled? I'll try it.
  Bill
