

EEVblog Electronics Community Forum



A Free & Open Forum For Electronics Enthusiasts & Professionals

Welcome, **Guest**. Please login or register.
Did you miss your activation email?

Forever

Login with username, password and session length

This topic

[Home](#) [Help](#) [Search](#) [About us](#) [Links](#) [Login](#) [Register](#)

EEVblog Electronics Community Forum » Electronics » Beginners » Z80 single board memory bank switching



Used N9917A FieldFox Handheld Microwave Analyzer, 18 GHz
Limited offer **55%** off, options addable, calibrated

SHOP NOW >>

KEYSIGHT TECHNOLOGIES

5 PCBs for \$2, any color

Production of PCB, SMT, Stencil is back to normal

J@LC JLCPCB

« previous next »

Pages: Prev 1 [2] 3 Next All **Go Down**

PRINT SEARCH

Author

Topic: Z80 single board memory bank switching (Read 33645 times)

0 Members and 4 Guests are viewing this topic.

☐ **grumpydoc**

Super Contributor



Posts: 2705

Country:

Re: Z80 single board memory bank switching

« **Reply #25 on:** July 28, 2013, 01:14:57 pm »

Quote

Nice, I've never seen that technique used before but it looks good to me.

It's a fairly well known trick, there are circumstances where it's **required** - eg 8080 interrupt handling but, TBH, it's not something I'd implement unless there were no choice.

Logged

☐ **MasterOfNone**

Regular Contributor



Posts: 123

Re: Z80 single board memory bank switching

« **Reply #26 on:** July 28, 2013, 01:37:18 pm »

Quote from: grumpydoc on July 28, 2013, 01:14:57 pm

Quote

Nice, I've never seen that technique used before but it looks good to me.

It's a fairly well known trick, there are circumstances where it's **required** - eg 8080 interrupt handling but, TBH, it's not something I'd implement unless there were no choice.

Apart from writing device drivers I never had much to do with the level workings of the Intel chips, I can't remember seeing that technique used on the Z80 and I've

certainly never seen it used on designs for things like Transputers, PowerPC, and SHARC DSPs. So I guess I just missed out on that one.

 Logged

☐ **grumpydoc**

Super Contributor



Posts: 2705

Country: 
 **Re: Z80 single board memory bank switching**

« Reply #27 on: July 28, 2013, 04:23:32 pm »

Quote

I can't remember seeing that technique used on the Z80 and I've certainly never seen it used on designs for things like Transputers, PowerPC, and SHARC DSPs. So I guess I just missed out on that one.

The only CPU that I have actually used where the interrupting device places an **instruction** on the bus is the 8080, although I assumed the intel designers probably weren't the first to think of this but always felt it was a bit of a kludge. However it's pretty common for the interrupting device to have to place some data on the bus which is then used to calculate a service routine address by table look-up - Z80 mode 2 interrupts work this way as do 8086 interrupts as do some other processors (eg the NS32032).

I've no idea how the transputer, power PC or any DSPs do it.

 Logged

☐ **MasterOfNone**

Regular Contributor



Posts: 123

 **Re: Z80 single board memory bank switching**

« Reply #28 on: July 28, 2013, 07:23:15 pm »

Quote from: grumpydoc on July 28, 2013, 04:23:32 pm**Quote**

I can't remember seeing that technique used on the Z80 and I've certainly never seen it used on designs for things like Transputers, PowerPC, and SHARC DSPs. So I guess I just missed out on that one.

The only CPU that I have actually used where the interrupting device places an **instruction** on the bus is the 8080, although I assumed the intel designers probably weren't the first to think of this but always felt it was a bit of a kludge. However it's pretty common for the interrupting device to have to place some data on the bus which is then used to calculate a service routine address by table look-up - Z80 mode 2 interrupts work this way as do 8086 interrupts as do some other processors (eg the NS32032).

I've no idea how the transputer, power PC or any DSPs do it.

OK Hardware modified interrupt vectors (and lookups) were very common. When the CPU is fetching the interrupt vector you usually have a pin you can use to enable the circuit that places the vector (part of the vector, or lookup) on the bus. You may have to combine a couple of pins to get this output signal. Usually you can just use the signal to enable the output of a buffer and which places the correct pattern on the bus (I can't remember seeing a Mux used, but same principle).

What was confusing me is normally when accessing memory you have no output pin that could be used to enable a special pattern that will be placed on the bus. So I just couldn't see how you could distinguish between the pattern and RAM without doing an address decode.

So amyk suggestion of generating the 'special pattern enable' signal by using M1 to clock the D-Types was something I've never seen before, not placing the Vectors on the bus. So when you said this technique is used on the 8080 for interrupts I was still thinking you meant the D-Types, not placing the vectors on the bus.

 Logged

☐ **grumpydoc**
Simple Z80 I/O interfacing pt 2

Super Contributor



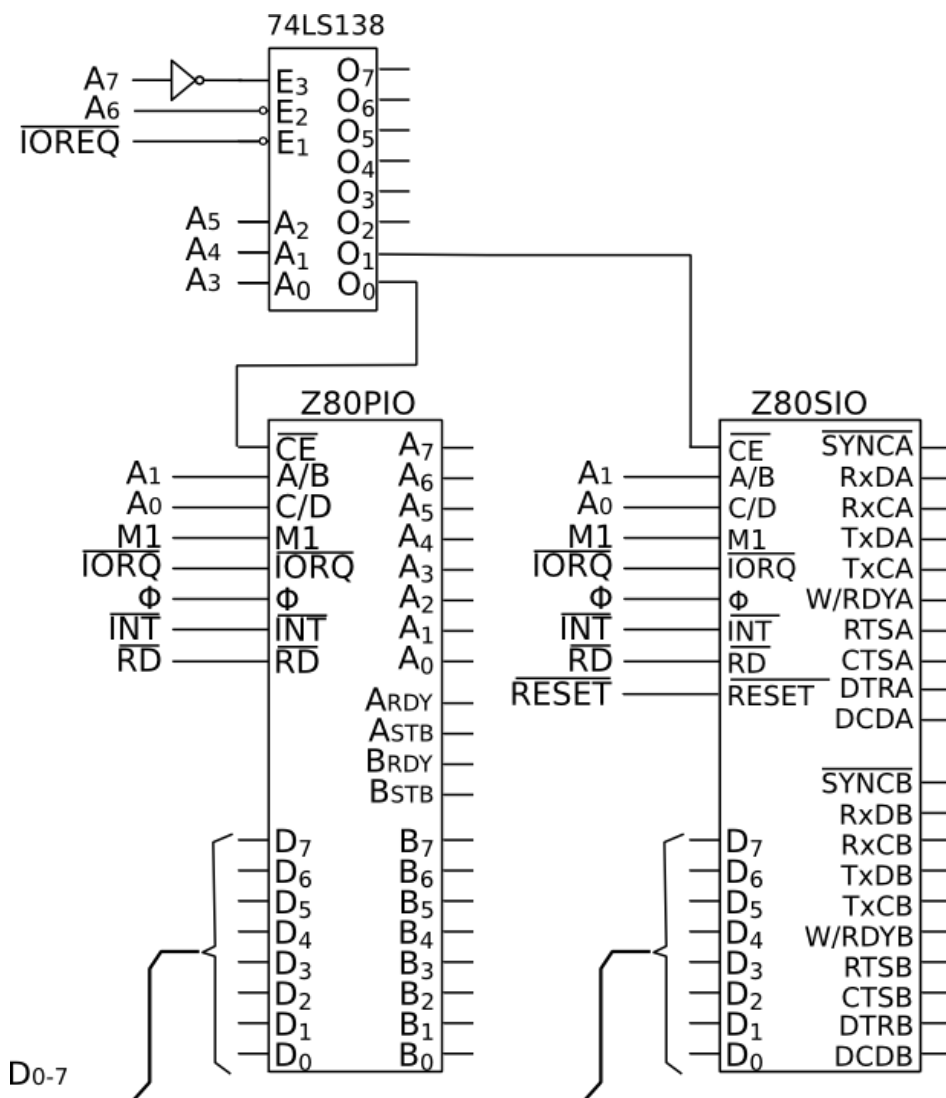
Posts: 2705

Country:



« **Reply #29 on:** July 28, 2013, 07:30:35 pm »

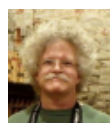
The final "simple" circuit snippet shows how to interface to a couple of typical Z80 peripheral chips. Hardware-wise this is extremely simple - you just need some address decode which provides a chip select for some valid I/O port range - the example uses a 74LS138 as previously.



One thing not shown is the IEI/IEO daisy chain - you need to connect these lines together if you wish to use the mode 2 interrupt support so that the peripheral chips know what priority they are in for generating interrupts.

Each chip occupies 4 I/O ports so the port address decode is still incomplete since each device can be accessed at it's base address and at base_address+4

Super Contributor



Posts: 3162

Country:



Re: Z80 single board memory bank switching

« **Reply #30 on:** July 28, 2013, 09:37:07 pm »

Quote

make the first 3 bytes read be [a jump to ROM addresses]

Or you can make your first three (or N) accesses decode to your ROM space (put a counter in your address decode), and actually put the jump instruction (and however many more you need) there.

A reason people don't remember the 8080's (or 8086) weird interrupt mechanism is that Intel did all the work for you inside the 8259 Interrupt controller. An 8086 interrupt "controller" was part of my BSEE "senior project" (back when the 8086 was a new chip.) It was pretty easy...

 Logged **grumpydoc**

Super Contributor



Posts: 2705

Country: **Advanced Z80 memory interfacing
pt 1**« **Reply #31 on:** July 28, 2013, 09:53:56 pm »

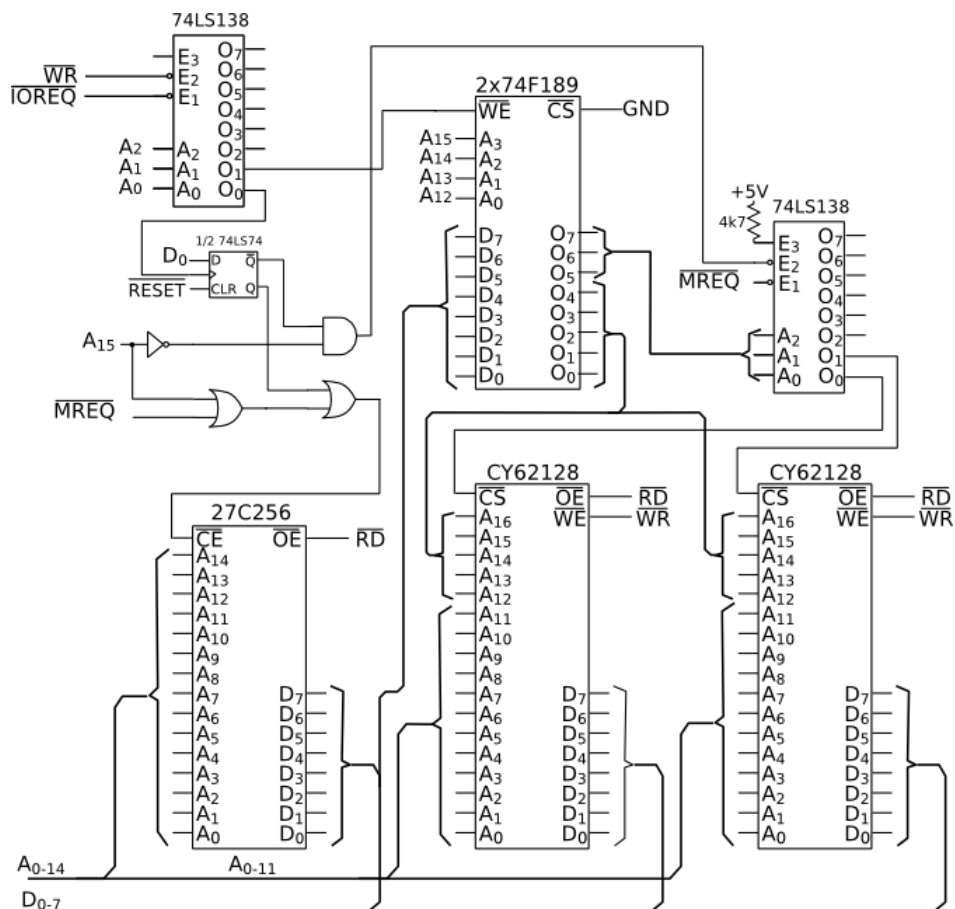
Something a little more fun - how to build a system with more than 64k of RAM¹.

I built a couple of systems like this when I was younger - mid 80's I think. Partly to see if I could and partly with the intent to develop a multi-tasking OS. That bit never happened 😊 but the hardware worked.

I've lost the original circuit diagrams and haven't got either of the boards to hand so I'm not sure this is *quite* how I did it - in fact I know it isn't because the originals used DRAM but these days you can get 128k x 8 static rams which would make it easier to build. Of course the down side is that 74F189's are not that easy to come by any more. There are still a few listed on ebay so it wouldn't be totally impossible to construct.

You *can* get fast static RAM which you could use to do the mapping although everything I can find is total overkill - about the smallest 10ns SRAM I can see is 256kbit. Also the separate data in and data out of the 74F189 make the circuit a lot simpler to implement. A 74F89 can be substituted with output pull-up resistors. 74LS parts can probably be used for a 4MHz Z80 but the access time of the RAM eats into the address set-up time so for a Z80B you really need 74F series. A Z80H probably won't run like this unless you delay MREQ to allow some address set-up time and avoid glitches on the chip select outputs.

The circuit as it stands is a little incomplete - you need to provide an I/O address range selector to the active high enable of the first 74LS138.



The Z80 memory map is divided into 16 4k byte pages - each of these pages can be mapped onto one of 256 4k pages in a total memory space of 1M byte. This is done by using a small fast scratch pad RAM consisting of two 74F189 16x4bit RAM to make a 16x8 bit RAM.

The outputs of the RAM form an extended 20-bit address bus. The top three bits drive a 74LS138 which is used to produce chip selects for up to eight CY62128 8x 128kbyte SRAM chips. The bottom 5 bits of the 74F189 outputs are then combined with the first 12 address bits from the Z80 to form the 17 address lines needed to drive the CY62128's

The bottom 32k of the Z80 address space is switchable between ROM and RAM. Following reset the D-type flip flop is cleared which allows A15 low to select the 27C256 when MREQ is also active. At the same time the complementary output from the flip flop is high this is ANDed with the complement of A15 and the result used to drive one of the active low enables on the address decode 74LS138. Thus for addresses where A15 is low that 74LS138 is disabled and no SRAM chip will be selected. When A15 is high (addresses from 0x8000 to 0xFFFF) the 74LS138 enable pin will be driven low.

Once a "1" is written to the flip flop the high signal will block the 27C256 from being selected and the 74LS138 enable line will be low regardless of A15's state.

With the ROM switched in all 1M byte of the expanded address range can still be accessed but only if mapped into the top half of the Z80 address space.

To set up a mapping we make use of the fact that the "OUT (C), A" instruction is really "OUT (BC), A" so the top four bits of B should be the page in the Z80 address space we wish to map and the accumulator should be the complement of the expanded address range page we want (the complement because the 74F189 outputs are inverted).

Edit: Oh, yes, while I think of it - as far as I can see this scheme would work just fine for bank switched CP/M 3 with suitable BIOS code to achieve the bank switching

as it's a much more flexible scheme than CP/M needs.

[1] Yes, I know, build a system which uses a processor that has a bigger address bus. At the time I wanted to build a 68000 based system but didn't have any form of OS to run on one and didn't fancy writing one. Also 68000's were a bit expensive.

« Last Edit: July 28, 2013, 10:12:00 pm by grumpydoc »

 Logged

The following users thanked this post: TomS_

 **MasterOfNone**

Regular Contributor



Posts: 123



Re: Z80 single board memory bank switching

« Reply #32 on: July 28, 2013, 10:21:48 pm »

Quote from: westfw on July 28, 2013, 09:37:07 pm

Quote

make the first 3 bytes read be [a jump to ROM addresses]

Or you can make your first three (or N) accesses decode to your ROM space (put a counter in your address decode), and actually put the jump instruction (and however many more you need) there.

Sorry but I'm not sure how that was different to what was suggested on the previous page?

Also one last point on the multiplexer idea, i'm pretty sure it wasn't common/popular because you would need something that is bidirectional, or you would have to use two (one for read the other for write), rather than just enabling the output of a buffer.


 Logged

 **MrAureliusR**

Supporter



Posts: 366

Country: 

frozenelectronics.ca



Re: Z80 single board memory bank switching

« Reply #33 on: July 28, 2013, 10:23:01 pm »

Holy smokes, I go to bed and come back and ... information overload! I've got to slowly go over all the info in this thread.

grumpydoc -- thank you so much for your help -- it's invaluable. Especially the part hooking up the PIO and SIO. I'm not sure I understand it fully yet but I'm going to spend the next hour or so looking at the schematics and trying to extrapolate what's going on. I tend to think of things in a frame-by-frame mentality (much like a logic analyzer) so I like to see what everything is doing at any given moment. That makes it a lot easier for me to understand. Like, I can read the schematic just fine but it doesn't mean anything until I know the timings between everything. I know that people who have been working with Z80's for a long time just know that sort of thing innately but this is my first crack at it and it's definitely not easy!

By the way, I'm using one of the newer Z80's -- the Z84C0020PEC, which can run up to 20MHz. I've got it clocked at about 12Hz for testing, is there a problem with that speed? I know since there's no DRAM and all the internal registers are also static it shouldn't matter, I just wanted to make sure I don't have something simple wrong that will throw a wrench into everything down the line.

So, the next big step is hooking up the SIO. I want to have I/O of course so I can see if this monitor is even being loaded properly. Of course, as you mentioned, this opens up a whole new can of worms. But I need to wrap my head around all of this if I hope to get anywhere. grumpydoc -- did you happen to look at the z80mon.asm file? Does it look like it's fairly well-written? To me, at least, it looks incomplete, but what is there is very well-implemented.

Anyway, if I run into any problems I'll ask here for help. Thanks guys!!

 Logged

Amateur Radio operator VA3XMR

westfw

Super Contributor



Posts: 3162
Country:

Re: Z80 single board memory bank switching

« Reply #34 on: July 28, 2013, 10:42:52 pm »

Grumpy's memory mapper is pretty similar to the scheme used on the SUN-1 68000 cpu card, except that it had two levels of page-mapping memory, and managed to reduce performance impact by overlapping the percolation of the upper address bits with the the DRAM RAS/CAS sequencing. (Although on the SUN, it wasn't for memory expansion, it was JUST for multitasking support.) You can find papers describing the SUN-1 online.

Quote

I'm not sure how that was different to what was suggested on the previous page?

I'm not sure which previous page, but I think the primary difference is that the "reset handler" hardware no longer has to tie into the databus, only into the address decoding. For the 32k/32k division, you'd have:
ROMSELECT = (A15 | RESETSEQ) & MREQ ; select ROM for high addresses, or for reset
RAMSELECT = (!A15 & !RESETSEQ) & MREQ ; select RAM for low addresses, unless resetting.

Logged

MasterOfNone

Regular Contributor

Posts: 123

Re: Z80 single board memory bank switching

« Reply #35 on: July 28, 2013, 11:18:32 pm »

Quote from: westfw on July 28, 2013, 10:42:52 pm

Quote

I'm not sure how that was different to what was suggested on the previous page?

I'm not sure which previous page, but I think the primary difference is that the "reset handler" hardware no longer has to tie into the databus, only into the address decoding. For the 32k/32k division, you'd have:
ROMSELECT = (A15 | RESETSEQ) & MREQ ; select ROM for high addresses, or for reset
RAMSELECT = (!A15 & !RESETSEQ) & MREQ ; select RAM for low addresses, unless resetting.

Ok got it. After the reset the ROM is at 0x0000 but only for one instruction plus data (I.e. RESETSEQ), the first instructions jumps to another location in ROM using the high memory address.

Logged

westfw

Super Contributor



Posts: 3162
Country:

Re: Z80 single board memory bank switching

« Reply #36 on: July 28, 2013, 11:26:40 pm »

yes; exactly.

Logged

ignator

Z80 developement board clock speed

Regular Contributor



Posts: 205

Country:

« **Reply #37 on:** July 29, 2013, 12:24:20 am »**Quote from: MrAureliusR on July 28, 2013, 10:23:01 pm**

By the way, I'm using one of the newer Z80's -- the Z84C0020PEC, which can run up to 20MHz.

20MHz may be difficult with your breadboard. What will get you is fast edges, and long connection wires. They are very inductive, the $V=L \cdot di/dt$, can make large lenz's law induced voltages. What might save you is very little mutual capacitance (if the wires are far apart and away from a power plane), may be small, which may reduce the di current pulse. The result is the control, address, and data lines will have lots of overshoot and undershoot. The undershoot causes current to be sucked out of the connected parts, and may cause upsets. From memory the 74LS138 has 7-8nano second fall time. But that's a max, and from experience current production may be sub-nanosecond. This dt is problematic in the real world. You can call them reflections if you want, but intuitively that's a misnomer to me. My experience was a 1991 PWB design that was marginal (very poor signal integrity) then, in 2004 new production parts, resulted in field failures. It was a data switch product with serial UARTs, every time the processor polled the status register to see if there was data in the data buffer, it just happens the 74138 control line went 5 volts negative (pulling current from a chip select input), the UART set the parity bad register, and it slew (threw away) the current shifting input data. So it dropped data. This was avionics, which have a 20-30 year product cycle life. And there was no specification change as it (fall time) exceeded the spec. You may have zero problems, but if you get bus cycles that cause the processor to get lost, you want to suspect timing signal integrity as possible cause.

« Last Edit: July 29, 2013, 02:54:40 am by ignator »

Logged

westfw

Super Contributor



Posts: 3162

Country:

**Re: Z80 single board memory bank switching**« **Reply #38 on:** July 29, 2013, 02:32:56 am »

Also, you'd need to watch your memory access times. IIRC, a 4MHz z80 required 250ns memory chips (which used to be "fast".) Your EPROM is probably not any faster than 100ns, which means you'd almost certainly have to generate some "wait states" to run much faster than 8MHz.

(obviously, none of this is a problem when running as 12Hz!)

Logged

grumpydoc

Super Contributor



Posts: 2705

Country:

**Re: Z80 single board memory bank switching**« **Reply #39 on:** July 29, 2013, 06:04:11 am »**Quote**

Also, you'd need to watch your memory access times. IIRC, a 4MHz z80 required 250ns memory chips (which used to be "fast".) Your EPROM is probably not any faster than 100ns, which means you'd almost certainly have to generate some "wait states" to run much faster than 8MHz.

The tightest timing is the M1 cycle which notionally takes 1.5 clocks for the read - however when you look at the timings you get about 260ns for your read cycle on a 4MHz part. For an 8MHz part not all of the timings scale with clock speed so you actually get 97.5ns for the cycle rather than the "expected" 125ns. This is without counting any extra delays for generating the chip selects (8ns per gate for LS-TTL) so a 100ns ROM will **definitely** need wait states.

I haven't looked at the timings for the 20MHz part but it's probably down at 30ns or tighter for the M1 cycle. Fortunately it's easy to find fairly fast RAM chips - the CY62128 is available as 70ns and 50ns parts (so probably good enough for a Z80H) and you can get 1mbit 10ns SRAMS (though not in DIP) which should mean it's possible to build a wait state free system even at 20MHz, though I don't fancy trying

that on a breadboard.

12Hz is 1-3 instructions per second. It would be neat to connect LEDs to the control signals and watch it run.

 Logged

 **westfw**

Super Contributor



Posts: 3162

Country: 



Re: Z80 single board memory bank switching

« **Reply #40 on:** July 29, 2013, 06:29:20 am »

Quote

12Hz is 1-3 instructions per second. It would be neat to connect LEDs to the control signals and watch it run.

I agree completely, BTW. Running at human-visible speeds is an excellent way to see what is going on. Just remember that at some point when you turn up the clock speed, you WILL run into limitations!

 Logged

 **grumpydoc**

Super Contributor



Posts: 2705

Country: 



Re: Z80 single board memory bank switching

« **Reply #41 on:** July 29, 2013, 06:45:50 am »

Quote

Grumpy's memory mapper is pretty similar to the scheme used on the SUN-1 68000 cpu card, except that it had two levels of page-mapping memory, and managed to reduce performance impact by overlapping the percolation of the upper address bits with the the DRAM RAS/CAS sequencing. (Although on the SUN, it wasn't for memory expansion, it was JUST for multitasking support.) You can find papers describing the SUN-1 online.

Ah, interesting - I didn't know that. Burying the propagation delays in the RAS/CAS strobe is a neat idea, not sure whether I'd have thought of that.

I didn't especially need to though - address-valid to MREQ is 65ns on a 4MHz Z80 which is what I was using - loosing 20ns in the page RAM is no big deal. Even on a Z80B (6MHz) it's 35ns so probably wouldn't have to do much hard thinking. A Z80H, however only provides 20ns of set-up time so you would need to think a bit about signal delays for it all to work without glitching. Again, no idea about 20MHz timing - anyone know of a suitable sub-5ns RAM that is currently available?

« *Last Edit: July 29, 2013, 07:02:06 am by grumpydoc* »


 Logged

 **grumpydoc**

Super Contributor



Posts: 2705

Country: 



Re: Z80 single board memory bank switching

« **Reply #42 on:** July 29, 2013, 07:03:47 am »

Quote

However, I would have thought wait states would be needed for the Z80 to handle **slower** memory devices. ie the Z80 inserts wait states until the device responds with valid data.

Yes, that's exactly what happens.

Edit: Actually I can see where some confusion might come in. If the data isn't ready the Z80 will delay in 1 clock cycle increments (the "wait state") until it becomes ready - but it needs to be **told** to do this via the WAIT input.

« *Last Edit: July 29, 2013, 08:02:14 am by grumpydoc* »

 Logged

 **MasterOfNone**

Regular Contributor



Posts: 123

**Re: Z80 single board memory bank switching**

« Reply #43 on: July 29, 2013, 07:45:08 am »

Quote from: wilfred on July 29, 2013, 06:56:48 am

I've been following this thread with interest too. I find the discussion of wait states so far a bit counter intuitive. It reads to me as if wait states are needed for faster memories. However, I would have thought wait states would be needed for the Z80 to handle **slower** memory devices. **ie the Z80 inserts wait states until the device responds with valid data.**

I've been thinking of a project similar to the OP's

Sorry but to me that reads like the Z80 will know when valid data isn't available, and will automatically wait for the data to become valid (I.e. handshaking or pre-programmed delay). Whilst you actually need to add external hardware to generate wait states long enough for the data to become valid.

« Last Edit: July 29, 2013, 07:51:07 am by MasterOfNone »

Logged

MrAureliusR

Supporter



Posts: 366

Country:

frozenelectronics.ca

**Re: Z80 single board memory bank switching**

« Reply #44 on: July 29, 2013, 12:11:37 pm »

Quote from: grumpydoc on July 29, 2013, 06:04:11 am

12Hz is 1-3 instructions per second. It would be neat to connect LEDs to the control signals and watch it run.

That's what I was doing in the beginning, just fed the Z80 NOP's and watched the address bus increment to make sure it was a good chip. Since then I throw in LEDs but I've been finding that depending on the LED sometimes it seems to draw so much that it interferes with the signals. I certainly wasn't expecting that to happen but for example when I put an LED on the /CE on the RAM, all of a sudden it was asserted even though the other end of the wire (coming from the OR gate) was high, or not asserted. I suppose it depends on the choice of LED, if I used my smallest 1.8V ones maybe...

Anyway, I haven't made a lot of progress yet, I will hopefully have some sort of logic analyzer in the near future which will really help me out, as I explained I like to see things in a state-by-state way, so I will hand-step the Z80 while watching the analyzer (or maybe run it at 2-4 Hz), in order to better understand everything. There's nothing like watching it do it's thing to learn about it! Reading the datasheet helps a bit but I find it hard to keep track of everything in my head.

I was actually thinking of building a debug board (like the ALICE II project), with LEDs for every address and data line, plus control lines, just so I can always see what's going on. Again, the trick is building that without interfering with the lines. I guess I could use latches or something?

Logged

Amateur Radio operator VA3XMR

**Re: Z80 single board memory bank switching**

« Reply #45 on: July 29, 2013, 12:31:03 pm »

LEDs will draw way too much current; the CMOS Z80 can only source 1.6mA and sink 2mA on each pin for the output voltage to be within spec (NMOS version is even less, 250uA source). You need to buffer the buses.

Logged

amyk

Super Contributor



Posts: 6763

☐ **SeanB**

Super Contributor



Posts: 15375

Country:

☐ **grumpydoc**

Super Contributor



Posts: 2705

Country:

Re: Z80 single board memory bank switching

« **Reply #46 on:** July 29, 2013, 07:27:36 pm »

If you are wanting to drive LED's use a bus buffer to drive them. Any octal bus buffer will do, unidirectional or bidirectional, inverting or non inverting. Better though to use an inverting buffer, and sink the LED current into the bus buffer, as they have a better defined current sink voltage of under 0.8V while high will just be over 2V4. Comes from only using NPN transistors internally.

 Logged

Advanced Z80 memory interfacing pt 2

« **Reply #47 on:** July 31, 2013, 10:33:35 pm »

I thought I'd finish my trip down memory lane with a look at interfacing dynamic RAM.

The systems that I built with large (i.e. > 64k) RAM had a DRAM interface for 2 banks of 256k x 1-bit DRAMs for up to 512k bytes of memory though I ended up with two systems which had 256k as I could only afford 16 DRAMs.

I confess here that this is not my original circuit - it's a fresh design based on a couple of things that I remember about the original - mostly that I only used a couple of flip-flops and I'm sure that I implemented CAS before RAS refresh to get around the fact that the Z80 only has a 7 bit refresh counter. It "feels about right" though 😊

This does mean I can't guarantee that it would work if implemented. I don't view that as too much of a big deal though since I don't think I'd bother these days. At the time 8k x 8 was a huge SRAM so to get 512k would have taken 64 chips - just not practical and much more price-wise than 16 DRAMs. Since you can now get much larger SRAMs I don't think I would bother with the complexity for a small Z80 SBC

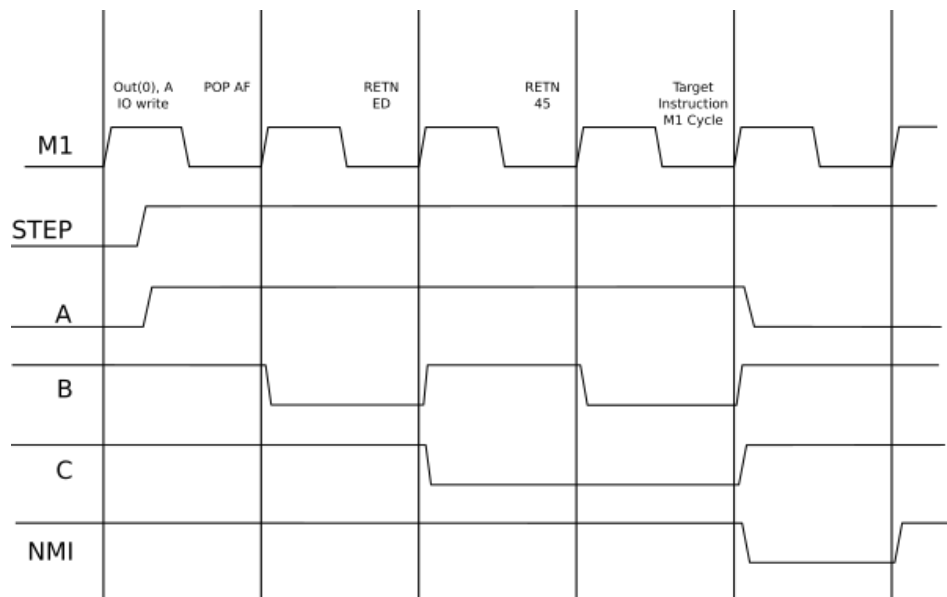
That said it's not all that complicated, what complexity there is comes from the fact that DRAMs have a multiplexed address bus and need to be periodically refreshed, fortunately the Z80 does some of the work for us by providing a refresh signal and a row counter for refresh addresses. Unfortunately anything bigger than a 16k DRAM tends to have more row address bits than the Z80 provides.

There is an excellent document produced by Mostek regarding dynamic memory which you can read [here](#)

Pretty much all the designs I've come across are based on the 74LS157 quad 2-input multiplexor.

The first thing to look at is the timing diagram - the Z80 has two memory read patterns and one memory write. Of these the opcode fetch or M1 cycle has the tightest timing so most people design to that and the longer memory read/write cycles should then take care of themselves.

Here's the timing (mostly without taking gate delays into account).



One thing to notice is that RAS is just a copy of MREQ (that's handy!). Using RAS before CAS means that the multiplexing signal which drives the 74LS157 just needs to be copy of MREQ which is not active during a refresh cycle - easy to drive with a couple of gates (the slight delay this produces helps to ensure that the row address hold times should be easily met).

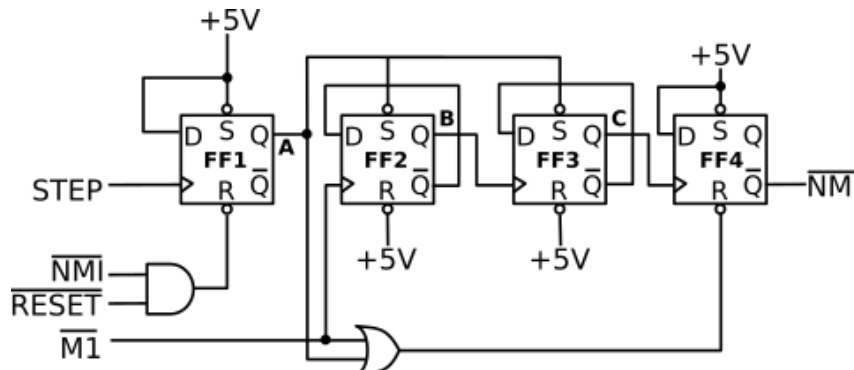
That just leaves CAS - this was generated in two halves. The first is a copy of MREQ which is delayed until the start of T2 (CASm on the diagrams). For a 4MHz Z80 MREQ is guaranteed to be not more than 85ns after the falling clock in T1 so that gives us about 40ns of RAS to CAS delay which should be OK.

The second part of the CAS signal is active during refresh - this is generated as a shortened version of the RFSH signal. In this case we start CAS with the falling edge of RFSH but deactivate it with the next rising clock edge. This is CASr on the diagrams.

The two signals are then combined with an AND gate,

Finally RAS is not quite an unmodified MREQ - we will probably want to supply an enable signal from the address decode logic, however we do not want this to interfere with RAS during the refresh cycle so we gate it with a select plus RFSH to generate the actual RAS circuit.

The circuit ends up looking like this.



EDIT: Looking at something else I chanced upon the fact that the above circuit is nothing to do with the memory interface - it is the Nascom 2 single step circuit. I obviously overwrote the old image my error. I'll try to restore the correct diagram.

I think it should work with most 64k x 4 DRAM chips. Clock to CAS should be about 40ns with LSTTL, for a 4MHz Z80 there's a 35ns data-in set-up time so the RAM

needs a CAS access time of 250-(40+35) or 175ns, give-or-take. As even the venerable 4116-25 had a T_{CAC} of 165ns I don't foresee any problems with anything faster. For a 6Mhz system the figure will be about 100ns which should be do-able with a "150ns" DRAM.

The only thing I can see that might be iffy is whether there's enough CAS precharge time for the refresh cycle. One problem is that there is no specification for the interval between MREQ going inactive and RFSH going active - however it's probably 40 or 50ns on a 4Mhz part which *should* be enough, 6 or 8Mhz parts might become marginal here. Wiser heads might spot other mistakes I made.

If anyone does implement it, let me know if it works 😊

Edit: Re-reading the Mostek document I was reminded of one small tip which had eluded my memory - it's wise to latch any address bits which are used to generate chip select signals with the falling edge of MREQ. The Z80 doesn't guarantee that the address bus will be valid for the whole of the time that MREQ is low. Thus it is possible to get short glitches on chip select signals which can cause problems - especially when used to generate RAS signals for DRAM.

« Last Edit: August 06, 2017, 11:16:10 am by grumpydoc »


 Logged



Super Contributor



Posts: 1345

Country: 



Re: Z80 single board memory bank switching

« Reply #48 on: August 03, 2013, 11:15:54 am »

MrAureliusR

Before you get too far you might want to take a quick look at the TurboDos Z80 Implementors Guide

As I said TurboDos lets you run CPM-80 programs while also letting Z80 machines network. TurboDos can be a standalone system like CPM-80 or be a network of Z80's.

You should be able to find all the needed files for TurboDos online. You will just need to add the bios part which they give examples of..

The neat part is that the Z80 Implementors Guide makes this software part of bios easy to understand and create | modify. And there is enough information there that you could have a file on a pc act like a disk drive to a z80 turbodos system with a small program running on a PC.

One connection to a PC would allow console, printer & drives to be via the PC. Worth a quick look to save you time and hardware needed on the Z80 to function as a computer system.

If you would like to see the schematic of a system that pushed the limits back then, you may want to look at the Tandy Radio Shack TRS model II or TRS model 12 these are online with a description of how it works.

While not stated in the description of this system, this system has bank switched memory hardware and DMA connected IO chips. The TRS model II was also the foundation to the model 16 which is really just a add on slave card with a 68000 processor on the slave card. When running a 68K operating system on a model 16 the Z80 was the smart IO processor to the 68K leaving the 68K with more time to work on user tasks.

C

 Logged

 netdudeuk

Frequent Contributor



Re: Z80 single board memory bank switching

« Reply #49 on: November 23, 2013, 11:23:14

pm »

Posts: 409

Country: 

I've been working on a new Z80 system and I am planning on using a 74LS138 as per post #11.

I have two 8K EEPROMs which will fit nicely off the first two outputs. I also have a 32K RAM chip which needs to sit in the top 32K of address space. I'm thinking about using a 74LS21 (dual four input AND gate) across the 138's O4-O7 pins with the output going to the active low chip enable on the RAM chip. The theory is that if none of the upper four 8K blocks are accessed, the output from the gate will be high, disabling the RAM chip. The RAM chip should be enabled if any of the inputs to the LS21 goes low.

My only concern is what happens with the LS21 if the MREQ on the LS138 is high. If the outputs from it go high impedance, could the LS21 end up in an unpredictable state ? Would it be better to have pull-ups on the LS138/LS21 connections ?

Thanks

 LoggedPages: Prev 1 [**2**] 3 Next All **Go Up**PRINT SEARCH
« previous next »

Share me



EEVblog Electronics Community Forum » Electronics » Beginners » Z80 single board memory bank switching

Jump to: => Beginners ▼ go

[EEVblog Main Site](#)[EEVblog on Youtube](#)[EEVblog on Twitter](#)[EEVblog on Facebook](#)[EEVblog on Library](#)

SMF 2.0.15 | SMF © 2017, Simple Machines

[Simple Audio Video Embedder](#)[SMFAds for Free Forums](#)[XHTML](#) [RSS](#) [Mobile](#) [WAP2](#)