Subject: ECB-DMA
Posted by b1ackmai1er on Sun, 10 Nov 2019 08:16:55 GMT
View Forum Message <> Reply to Message

Hi guys,

I have been working on reimplementing Wolfgang Kabtazke's ECB-DMA board.

https://www.retrobrewcomputers.org/doku.php?id=boards:ecb:dm a:start

I beleive this was prototyped but never finalized.

Kicad files have been regenerated and I have just finished routing the board in 10x10 format.

Would anyone be interested in building and debugging this board if I order some?

Regards Phil.

File Attachments
1) ECB-DMA-R01A.jpg, downloaded 2013 times

Subject: Re: ECB-DMA
Posted by Wayne W on Tue, 12 Nov 2019 20:31:35 GMT
View Forum Message <> Reply to Message

Hi Phil,

I would be happy to help. I am severely constrained on time these days, so I won't be fast.

Not knowing much about DMA without reading up on it, I am curious what peripheral boards will support/benefit from DMA? What are you targeting for initial testing?

-Wayne

Subject: Re: ECB-DMA
Posted by jcoffman on Wed, 13 Nov 2019 00:14:59 GMT
View Forum Message <> Reply to Message

Phil,

I'm just going from memory, but I believe there was an error on the SBC v1 that prevented its use with DMA. SBC v2 supposedly corrected the error. It looks like I tried to support DMA on the

Z180 Mark IV, but that board uses >16 address lines.  SBC-188 does not support external DMA; it uses the on-chip DMA controller for floppy access.  Kiss-68030 does not support DMA.

IMHO, the DMA controller should be on the same motherboard as the CPU.  Take a very good look at SBC v2.  Address & data lines look okay; as do /MREQ, /IORQ, /RD, & /WR.  /M1 does not tri-state; I hope the CPU keeps it high.

==========
==========

None of the existing backplanes support DMA or Z80 interrupt priority arbitration out of the box.  All must have the /IEI-/IEO & /BAI-/BAO jumper wires soldered in place.  Also, the DMA board must be placed on the correct side of the CPU board, and adjacent to it, to get the priority chains to work.  The CPU board will have to be modified to ground /IEO & /BAO to initiate the priority chains.

The 3-slot, 8-slot, and 12-slot backplanes will need to have the priority chain jumpers soldered in place.  AFAIK, no one has ever done this, since there is no DMA board or board generating interrupts using the Z80 interrupt priority scheme.

--John

---

Subject: Re: ECB-DMA
Posted by etchedpixels on Wed, 13 Nov 2019 23:40:36 GMT
View Forum Message <> Reply to Message

On RC2014 the main uses I've found for the Z80 DMA so far have been
- CF card adapters (doing a burst transfer)
- Big memory copies with Fuzix (which is not really CP/M relevant in most cases)
- Sound - where I am using it together with a CTC and a simple DAC to try and build myself an 8bit DMA soundcard.

The CF adapter is the big win. As the device is a good order of magnitude faster than the CPU having stuff DMA directly into memory to run makes a difference you can feel. Doesn't help with 82C55 style IDE though.

Alan

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sat, 16 Nov 2019 04:34:20 GMT
View Forum Message <> Reply to Message

Hi everyone,

>>> The CF adapter is the big win.

I did not actually have anything particularily in mind for this but was wondering if it could be used for ROMWBW bank to bank memory copy. If dma is currently not used on the MKIV then maybe there is the opportunity to play with that (cp/m 3 disk buffers?). CF adapter is a good suggestion since I presume that is what most people are using. Been working off floppy disk drives the last couple of days and had forgotten how slow they are :)

>>> I'm just going from memory, but I believe there was an error on the SBC v1 that prevented its use with DMA. SBC v2 supposedly corrected the error.

Yes I remember reading about this as well.

>>> IMHO, the DMA controller should be on the same motherboard as the CPU.

Agreed, I really like the idea of dropping the 8255 off the SBC V2 (connecting IDE directly to the bus as Sergeys suggestion) and replacing with DMA chip. That might be within my capability.

>>> Address & data lines look okay; as do /MREQ, /IORQ, /RD, & /WR. /M1 does not tri-state; I hope the CPU keeps it high.

Had assumed the SBC V2 would be good to go. Would not know how implement tri-state on /M1 as there is no more room for another buffer/driver chip. Wonder what S100 designs do.

>>> None of the existing backplanes support DMA or Z80 interrupt priority arbitration out of the box.

I had installed all links on the boards I had constructed not really understanding the full use. Fortunately easy to install for those haven't. Now I'm wondering if they should only we installed in locations where the ecb-board is not using /IEI-/IEO & /BAI-/BA

>>> Also, the DMA board must be placed on the correct side of the CPU board, and adjacent to it, to get the priority chains to work.

I wasn't aware of this requirement. I think some of Wolgangs design have  /IEI-/IEO & /BAI-/BA linked on the ecb-board. It seems to me that that is the correct way to go to make them position independant and removes the need for the links on the backplane?

>>> The CPU board will have to be modified to ground /IEO & /BAO to initiate the priority chains.

Not really understanding this yet ... will get the datasheet out and read up more. Thank you for your insightful comments.

At the moment feeling like I have bitten off more than I can chew ...

Maybe a good place to start would be with the Data Gear DMA add on board
https://velesoft.speccy.cz/data-gear.htm

regards Phil.

PS I have gone ahead and ordered 5xData Gear DMA boards.

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sat, 16 Nov 2019 12:56:01 GMT
View Forum Message <> Reply to Message

Thinking aloud ...

>>> /M1 does not tri-state; I hope the CPU keeps it high.

So looking at the data sheet, /M1 is an input only for the z80 dma and it does not state that it is expecting this to be tri-state like some other signals. /M1 high is use to qualify I/O selection - if it were low then the dma would not be able to access any ports. It makes sense that /M1 would remain high during when it is not in control of the bus and goes against it's purpose to be active at any other time then when it has control of the bus.

... So I think this is ok.

---

Subject: Re: ECB-DMA
Posted by etchedpixels on Sat, 16 Nov 2019 13:23:08 GMT
View Forum Message <> Reply to Message

Take a look at Marten's Z80DMA for RC2014. It's a much simpler bus so rather easier to understand IMHO.

IEI/IEO was always a problem with backplanes. S100 systems generally don't use it - S100 after all has its own set of vectored interrupts. Cromemco systems did but they use flying leads for it so that you can chain interrupts neatly without having to worry about exact slot orders and magic jumpers in unused slots. The 16x50 on the board doesn't support IEI/IEO anyway so you can't really use Z80 IM2 properly. What you can do though is to use the spare modem pins on the 16x50 as an interrupt controller as it will let you generate interrupts on modem line changes. That (plus a 10Hz clock board) is how I run Fuzix on SBCv2.

S100 also solved the DMA access problems its own unique way and supports priority bus mastering so is a whole chunk of extra logic you probably wouldn't want to copy.

The Z80DMA isn't actually much use for bank to bank copying unless you've got a better MMU design. On a Zeta v2 style MMU you can map any pair of 16K banks and DMA between them. On the SBCv2 you'd need to have some kind of separate DMA bank select logic and latches - like the Altos 5 did. CP/M 3 bank to bank copying is not usually much of a win anyway. It gets you a slightly bigger TPA but at a measurable performance hit over not having XMOVE and ending up with a 512 byte buffer per CF drive (drive in CP/M sense) in the common.

CP/M 2.2 it's hard to use the Z80DMA effectively. CP/M 3 for I/O you can at least do larger I/O directly using the DMA (in particular the CCP loading applications which is one of the most obvious 'feels fast' points) but not really use it for buffering, for MP/M it fits much better as MP/M expects your driver transfer code to be in common space and you switch bank to the target, do the I/O then switch back. That plus the fact you know about multi-block I/O fits the DMA better.

For floppy the biggest use IMHO is that you can leave interrupts on during transfers even on 1.44/2.88MB media and you'll not drop anything. Very useful in MP/M.

I'm wary of just blindly going to a raw CF adapter. The number of incompatibilities and timing issues with that is proving inconveniently large in other projects.

---

Subject: Re: ECB-DMA
Posted by jcoffman on Sat, 16 Nov 2019 15:45:19 GMT
View Forum Message <> Reply to Message

b1ackmai1er wrote on Fri, 15 November 2019 20:34Hi everyone,

Agreed, I really like the idea of dropping the 8255 off the SBC V2 (connecting IDE directly to the bus as Sergeys suggestion)

====Yes.  This was the approach on the Mark IV, and the idea worked.

>>> Address & data lines look okay; as do /MREQ, /IORQ, /RD, & /WR. /M1 does not tri-state; I hope the CPU keeps it high.

Had assumed the SBC V2 would be good to go. Would not know how implement tri-state on /M1 as

====With CPU & DMA on the same board, you don't need to worry about the chip signals.  DMA knows to keep hands off of /M1.  BTW:  S-100 get the board priority done the right way.  Each board is jumpered for its priority, and board may be plugged in anywhere on the backplane.

>>> None of the existing backplanes support DMA or Z80 interrupt priority arbitration out of the box.

I had installed all links on the boards I had constructed not really understanding the full use. Fortunately easy to install for those haven't. Now I'm wondering if they should only we installed in locations where the ecb-board is not using /IEI-/IEO & /BAI-/BA

====Yes they should.

>>> Also, the DMA board must be placed on the correct side of the CPU board, and adjacent to it, to get the priority chains to work.

I wasn't aware of this requirement. I think some of Wolgangs design have /IEI-/IEO & /BAI-/BA linked on the ecb-board. It seems to me that that is the correct way to go to make them position independant and removes the need for the links on the backplane?

>>> The CPU board will have to be modified to ground /IEO & /BAO to initiate the priority chains.

==== Don't forget, /BAO goes out from the CPU board; in on the next backplane slot on /BAI; hence the need for the soldered jumper on the backplane. [BAO and BAI should actually be on Axx & Cyy, respectively, where xx == yy. Similarly for IEO & IEI. However, the two are backwards on the Kontron backplane. IEO is on pin Caa and IEI on Abb (aa == bb). Note: BAI->BAO goes from row A->C; IEI->IE0 goes from row C->A. The backplane wiring would be easier if these signals came into a board from one side, and both left from the other side, and each change on the SAME NUMBERED PIN; i.e., Axx->Cxx and Ayy->Cyy respectively. Kontron really missed the mark on the chaining scheme. They require soldered jumpers, or more than a 2 layer backplane.

Not really understanding this yet ... will get the datasheet out and read up more. Thank you for your insightful comments.

====Wolfgang published the schematics of a couple of Kontron boards. Reading the schematic would tell you immediately how this is done.

--John

Subject: Re: ECB-DMA
Posted by jcoffman on Sat, 16 Nov 2019 15:56:09 GMT
View Forum Message <> Reply to Message

I couldn't find my Z80 manual, but I do know that the CPU and DMA are designed to hand-shake properly.

--John

Subject: Re: ECB-DMA
Posted by jcoffman on Sat, 16 Nov 2019 16:03:23 GMT
View Forum Message <> Reply to Message

I applaud S-100 for getting the DMA priority resolution done right.

CF cards directly on the Z180 Mark IV bus have not shown any problems. The major problem I see is the incompatibility between certain pairs of Master/Slave IDE cards. But this problem crops up on MF/PIC PPIDE, SBC-188 with 8255-to-ppide daughter card, and elsewhere.

--John

Subject: Re: ECB-DMA
Posted by wsm on Sat, 16 Nov 2019 17:28:23 GMT
View Forum Message <> Reply to Message

I use DMA extensively in my Z180 designs and etchedpixels post confirms many of the reasons why I quit using the Z80 and only ever used the Z80 DMA in one design a VERY LONG time ago. Of course the Z180 also has the convenience of integrated peripherals and higher speed :) Since the HD64180 came out in 1985, I still consider it and it's faster cousins to be "retro".

The Z180 DMA has a major enhancement, namely 20-bit addresses which allows it to fully address 1MB of memory independent of any MMU settings. Thus it is straightforward to do I/O to/from a bank and also bank-to-bank transfers such as buffer blocking. Designs using the Z80 DMA and more than 64K could greatly benefit from using a similar technique by using external registers for DMA address bits higher than A15. The one negative is that DMA transfers wouldn't be able to span a 64K boundary but that can be overcome in SW.

I also agree that many of the ATA designs I've seen don't pay attention to the ATA specifications about timing and would be prone to failure. Interfaces like GIDE and one recently promoted on VCF simply rely on having a slow processor that doesn't exceed the PIO mode 0 timing of 290ns rd/wr pulses and 600ns cycle times ... unrealistic on something like a 20MHz Z80 without external wait logic. Note that the ATA specification calls for all devices to be initialized in PIO mode 0 before possibly being configured for higher rates, including DMA.

The only reliable way I've found to efficiently implement ATA on fast processors is to use a CPLD or FPGA to control the timing.  In it's simplest form it can assert a processor wait signal for register access and PIO modes plus take over the bus for the entire duration of a DMA block.  DMA mode is most efficient in an FPGA using a dual-ported buffer between the two different clock domains and Ultra DMA adds a LOT of extra complexity.

I would suspect the reason that some [many?] CF cards work with non-compliant ATA interfaces is due to them not forcing the specification timings.  I've definitely seen CF cards that will respond properly to PIO mode 4 timing even though they're configured for the much slower PIO mode 0.  This may also explain why only certain brands of CF cards may work on a particular interface and also why physical disks might not.  I've experienced hard drives that were picky about being properly configured with timing that matched the selected mode.

---

Subject: Re: ECB-DMA
Posted by jcoffman on Sat, 16 Nov 2019 21:09:47 GMT
View Forum Message <> Reply to Message

wsm wrote on Sat, 16 November 2019 09:28
I would suspect the reason that some [many?] CF cards work with non-compliant ATA interfaces is due to them not forcing the specification timings.  I've definitely seen CF cards that will respond properly to PIO mode 4 timing even though they're configured for the much slower PIO mode 0.  This may also explain why only certain brands of CF cards may work on a particular interface and also why physical disks might not.  I've experienced hard drives that were picky about being properly configured with timing that matched the selected mode.

Thank you for the explanation about CF card timing.  SD cards have some slow timing requirements on start-up, although they are of a different nature.

--John

---

Subject: Re: ECB-DMA
Posted by wsm on Sat, 16 Nov 2019 22:02:04 GMT
View Forum Message <> Reply to Message

Quote:Thank you for the explanation about CF card timing.  SD cards have some slow timing requirements on start-up, although they are of a different nature.
You're welcome.  As to SD, yes there's the 400KHz initialization pulses and there can also be data transfer timing issues.  A 25MHz HW SPI can be slower than a 33MHz Z180 INIR instruction and I now use DMA rather than processor timing.  This allows the transfer to occur as soon as the SPI interface can handle the data transfer and moves timing issues to within my CPLD.

I know I've wandered a bit off-topic but for those that don't want to dig through the ATA specification, I'm including some timing comments from an ATA CPLD that I developed and has been used successfully with CF cards, ATA hard disks and an ATA to SATA interface.

```
//              cycle   a_setup  rd/wr(reg)  rd/wr  wr_hold
//      PIO 0 :  600      70      290         165     30
//      PIO 1 :  383      50      290         125     20
//      PIO 2 :  330      30      290         100     15
//      PIO 3 :  180      30           80          10
//      PIO 4 :  120      25           70          10
//   CF PIO 5 :  100      15            65           5
//   CF PIO 6 :   80      10            55           5

//                 cycle   r_setup  rd/wr  wr_hold  rd_neg  wr_neg
//      DMA 0 :      480      150     215      20      50      215
//      DMA 1 :      150       60      80      15      50       50
//      DMA 2 :      120       50      70      10      25       25
//   CF DMA 3 :      100       50      65       5      25       25
//   CF DMA 4 :       80       45      55       5      20       20
```

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sun, 17 Nov 2019 13:50:26 GMT
View Forum Message <> Reply to Message

Just working through all the great feedback and discussion, thanks.

God bless JLCPCB - Data Gear DMA boards are on the way.

John, got your message re 24 bit address mode - I gather the extra 8 bits would be set up as an i/o port that latches the address on the bus the same time as the A0-A15 is by ~BAI. Think this would still be limited to a 64K range within the 24 bit address space as I cannot see how you could differentiate between a source and destination transfer.

Regards Phil.

---

Subject: Re: ECB-DMA
Posted by jcoffman on Sun, 17 Nov 2019 15:06:39 GMT
View Forum Message <> Reply to Message

What I suggest is the same thing IBM did using the Intel 8237 DMA controller. On the PC/XT, addressing was extended from 16 bits to 20 bits, with the restriction that DMA could not cross a 64K boundary. I have never looked at the PC/AT schematics, but I imagine the extension was from 16 bits to 24 bits using two 74LS670's instead of one.

I would imagine that you are aware of the two types of DMA transfers. Simplest, as use on the N80188 and Z80DMA, is the 2 bus cycle "read source; save data in DMA controller buffer; write to destination." The 8237 & other controllers use the "fly-by" transfer where both the source and

destination are enabled simultaneously, one being read and the other being written. Memory to memory transfers are not possible, but everything is accomplished in a single bus cycle. The former scheme, using the DMA chip buffer does allow memory-to-memory transfers.

The 2 cycle DMA is easier to implement.

Intel app note AP-116 on the use of the 8237 is here:
http://www.intel-vintage.info/intelotherresources.htm#889713 809
Fly-by transfers are not easy ...

--John

---

Subject: Re: ECB-DMA
Posted by etchedpixels on Sun, 17 Nov 2019 15:41:22 GMT
View Forum Message <> Reply to Message

For the simple case all you need to do is ensure that you can load different MMU translations for read and write cycles. You don't even need to do anything special for the DMA side if you also support it for CPU (which is useful anyway). Then \RD or \WR selects your translation.

With the 16K banking ZetaV2 style the problem goes away because you can always map the banks you need into RAM together and do the transfer.

@wsm: I don't think its necessarily modes and timings that are the problem. I see the same problems with the Memotech MTX CFX-II and with the backplane based CF cards in RC2014. In both cases getting the CF adapter closer to the CPU helps which to me suggests it also involves signal quality. Likewise on some buffered adapters people have found that 74LS or 74ALS works and 74HC/HCT does not for certain cards.

---

Subject: Re: ECB-DMA
Posted by wsm on Sun, 17 Nov 2019 18:06:49 GMT
View Forum Message <> Reply to Message

@jcoffman : Having implemented both "fly-by" and "flowthrough" DMA controllers in CPLDs, I agree that "flowthrough" is somewhat easier to implement and understand but would respectfully disagree that "fly-by" transfers are not easy. The system bus address and control signals, such as RD and WR, reflect memory signals. The trick is to have unique peripheral control signals such as ATA's CSx, DAx, DIOR and DIOW. The potential problem becomes bus loading issues. Of course there's also the issue of whether a generalized DMA controller supports "fly-by" and can interface in this way.

@etchedpixels : Perhaps there are signal quality issues but when I see different logic families behave differently, I tend to first look at factors like propogation delays, enable/disable times and bus loading. For example, the HCT245 propogation delay is longer than the LS245. One thing to

note with ATA is the relatively long address setup times before the DIOR/DIOW signal.

I would recommend keeping the DMA extended addressing completely independent of the MMU. After all, the basic concept of DMA is to allow simultaneous I/O and processor activity, especially in a multi-tasking environment. There simply needs to be two address latches for the DMA extended addressing, one for reads and one for writes. The only relatively insignificant DMA restriction then becomes the 64K boundary since it would take a lot of extra logic for the DMA controller to be able to increment the external address latches.

Phantom memory which replaces parts of the address space can create a different set of problems compared to purely linear extended addressing since it requires the address decoding and chip select logic to be aware of processor versus DMA activity.

---

Subject: Re: ECB-DMA
Posted by wsm on Sun, 17 Nov 2019 22:12:56 GMT
View Forum Message <> Reply to Message

Since we've been discussing Z80s, DMA, CF and ATA, there is an ambiguity that I think needs to be clarified.

Many of the CF interfaces that I've seen for 8-bit processors utilize the CF-unique SET FEATURES command to enable 8-bit transfers. This feature is ONLY useable in PIO mode and was retired for ATA compliant devices such as hard disks in ATA-3 (1997). CF cards in 8-bit mode can work with an 8-bit DMA controller BUT the CF card actually needs to be configured for PIO mode, not DMA mode. Compliant ATA-3+ devices and all CF / ATA devices configured for DMA mode always use 16-bit data transfers.

The first ATA CPLD I built was for an 8-bit system and fully supported PIO and DMA modes (not Ultra) on CF cards and hard disks. This required extra logic for high byte versus low byte but did increase throughput since it only required one ATA access per two memory accesses. On my NYOZ system with it's 8-bit Z180 I chose to use an 8/16-bit memory organization in order to further increase ATA throughput and due to DMA "fly-by" each ATA DMA access also includes the main memory access.

---

Subject: Re: ECB-DMA
Posted by jcoffman on Mon, 18 Nov 2019 05:28:41 GMT
View Forum Message <> Reply to Message

etchedpixels wrote on Sun, 17 November 2019 07:41
@wsm: I don't think its necessarily modes and timings that are the problem. I see the same problems with the Memotech MTX CFX-II and with the backplane based CF cards in RC2014. In both cases getting the CF adapter closer to the CPU helps which to me suggests it also involves signal quality. Likewise on some buffered adapters people have found that 74LS or 74ALS works and 74HC/HCT does not for certain cards.

I have twice had bad experiences with 74ACT logic; i.e., flaky to non-operational boards. First, for low power, I used a 74ACT373 address latch. Flaky operation was cured by switching to 74ALS373. Similar problem on the Z280 design from Tilmann Reh. Getting rid of 74ACT logic in favor of 74ALS / 74LS solved the problem.

My take on the situation is as follows: both of these boards distribute power & ground via wide traces, not power & ground planes. 74ACT probably has micro power use when not switching, but large current surges when it switches. 74LS and 74ALS logic, although using more power, do not produce these surges. I notice that DRAM strips (KISS-68030 & 80386EX) have rather large electrolytic, or tantalum, caps right on the strips to provide the high current surges which occur when the chips switch.

I'm surprised to hear that HC & HCT may exhibit a similar problem, although there may be another explanation. Many of the buffer driver chips in the 74LS series have hysteresis on their inputs. ACT, HCT, HC, F, AS, ALS do not.

Chips with 74LS hysteresis: '240, '241, '244, '245
Chips with hysteresis in all chip families: '13, '14, '132

--John

---

Subject: Re: ECB-DMA
Posted by rcini on Fri, 13 Dec 2019 01:52:47 GMT
View Forum Message <> Reply to Message

Just FYI, I put together a schematic for a SCSI host adapter which uses DMA. It has not been tested, but is based on a reference design I found on-line, although it was for a 6502-based system. So, I had to adapt it a bit for the ECB. The schematic is attached, I feel that the DMA part of it needs some work, and maybe the chip-select logic. Other than that, it's a simple design.

When looking at it, if anyone sees something that needs to be changed, let me know.

Rich

File Attachments
1) ECB SCSI-1.0-000A.pdf, downloaded 478 times

---

Subject: Re: ECB-DMA
Posted by wsm on Fri, 13 Dec 2019 05:24:40 GMT
View Forum Message <> Reply to Message

I don't have any of the ECB cards so won't comment on that interface. However, I did develop a SCSI interface board for my NYOZ system using the Z53C80 which is the CMOS version of the

Z5380 / NCR5380.  A couple of comments:

Termination takes a LOT of current and is one more potential configuration problem, especially when devices are added / removed from the chain or powered down.  I chose to use active termination based on the UCC5606 which also has the advantage of offering 2.5K Ohm versus 110 Ohm termination for short cable runs.

I'm not sure which processor you intend to use but the 53*80 requires a relatively long data hold after the IOW* goes inactive: 30/40 ns for the 5380 and 20/25 ns for the 53C80 (CPU write/DMA write).

Note that the datasheet discusses block-mode DMA restrictions and does not recommend using this mode due to the difficulty in detecting when there are SCSI bus errors.

If using the Z180's DMA channels, it should be noted that on DMA reads they generate the TEND signal on the last DMA memory write, and not on the I/O read as the peripheral would expect. Although I see the XFER_END# signal going into your 5380's EOP#, it's not obvious to me where it's actually coming from.

## File Attachments

1) SCSI.png, downloaded 421 times

---

Subject: Re: ECB-DMA
Posted by rcini on Fri, 13 Dec 2019 15:22:12 GMT
View Forum Message <> Reply to Message

I like some of the features of the design you have such as the terminator power. The design I did is not finished -- I left the DMA stuff open because I didn't know how to handle it -- lots of "placeholder" kind of stuff. So, I'm kind of hoping someone can take this design as a starting point and build on it.

---

Subject: Re: ECB-DMA
Posted by mikesmith on Fri, 10 Jan 2020 02:36:11 GMT
View Forum Message <> Reply to Message

wsm wrote on Sat, 16 November 2019 09:28...
I also agree that many of the ATA designs I've seen don't pay attention to the ATA specifications about timing and would be prone to failure.  Interfaces like GIDE and one recently promoted on VCF simply rely on having a slow processor that doesn't exceed the PIO mode 0 timing of 290ns rd/wr pulses and 600ns cycle times ... unrealistic on something like a 20MHz Z80 without external wait logic.  Note that the ATA specification calls for all devices to be initialized in PIO mode 0 before possibly being configured for higher rates, including DMA.

The only reliable way I've found to efficiently implement ATA on fast processors is to use a CPLD or FPGA to control the timing.  In it's simplest form it can assert a processor wait signal for register

access and PIO modes plus take over the bus for the entire duration of a DMA block. DMA mode is most efficient in an FPGA using a dual-ported buffer between the two different clock domains and Ultra DMA adds a LOT of extra complexity.

I would suspect the reason that some [many?] CF cards work with non-compliant ATA interfaces is due to them not forcing the specification timings. I've definitely seen CF cards that will respond properly to PIO mode 4 timing even though they're configured for the much slower PIO mode 0. This may also explain why only certain brands of CF cards may work on a particular interface and also why physical disks might not. I've experienced hard drives that were picky about being properly configured with timing that matched the selected mode.
As someone that's been tinkering with GIDE in the RC2014 context recently (waiting for Plasmo to sell me one of his PC90 boards) I'd be super interested in reading more about your experiences here. I went looking over at VCF to see if I could find the thread(s) you allude to here without any success. Any pointers (in general) would be gratefully appreciated...

---

Subject: Re: ECB-DMA
Posted by wsm on Fri, 10 Jan 2020 04:28:36 GMT
View Forum Message <> Reply to Message

Quote:As someone that's been tinkering with GIDE in the RC2014 context recently (waiting for Plasmo to sell me one of his PC90 boards) I'd be super interested in reading more about your experiences here. I went looking over at VCF to see if I could find the thread(s) you allude to here without any success. Any pointers (in general) would be gratefully appreciated...
The VCF project I was referring to is the uIDE interface for the Z80 and there's also a wiki with a LOT of generalized information. However, I take exception to the use of the word "universal" since there's no discussion of IDE (actually ATA) timing and it's guaranteed to fail on fast Z80's with devices that are ATA compliant.

I haven't done a lot of write-ups here about my projects. Buried in a verbose document <here> is some discussion about an ATA interface that I've used with PIO and DMA on a Z180 at 33MHz.

Although I need to get back to my FPGA version of the ATA card, recently I've been playing around with microSD cards on my MinZ-cased system. With a 25MHz SPI interface it has a slightly higher transfer rate than ATA PIO mode 0 in 8-bit mode.

---

Subject: Re: ECB-DMA
Posted by lynchaj on Mon, 26 Apr 2021 16:03:44 GMT
View Forum Message <> Reply to Message

Hi (possible necro-post)

This question of DMA for the ECB has been kicking around in one form or another for several years on this forum and the preceding mailing list. I'd like to suggest a community project to determine if the Wolfgang Kabtazke's ECB-DMA board works or not. However the issue of no DMA capable peripheral boards is a limiting factor. So my first question is "are there any DMA

capable ECB boards" as part of RBC wiki or elsewhere? Do we need a DMA capable peripheral or would an ECB memory board suffice for testing the ECB-DMA?

I suspect there are no DMA capable peripherals (please correct me if I am wrong). Testing the ECB-DMA would also require a purpose-built peripheral that's DMA capable. Is that correct?

Maybe the way forward is to design a second DMA capable peripheral board specially designed to verify the ECB-DMA board. Once the ECB-DMA & DMA peripheral pair is working, the concept could be extended to other peripherals such as the ECB-SCSI and/or other peripherals like serial, parallel boards, memory, video boards, etc.

Does that sound like a reasonable approach? Otherwise I don't see a way to test the ECB-DMA and ensure it works. This could open up a whole bunch of new possibilities for RBC

Thanks, Andrew Lynch

---

Subject: Re: ECB-DMA
Posted by lynchaj on Tue, 27 Apr 2021 11:41:34 GMT
View Forum Message <> Reply to Message

Hi
The FDC circuit on the N8 supports DMA operations with the Z8S180 CPU in theory. Has anyone gotten the N8's FDC DMA mode working? This seems like a logical starting point for a DMA peripheral since it already has most of the logic filled out already. Also might be a good test bed to debug some software unless it's already been written. Not quite the WK ECB-DMA board but a step in the right direction.

Thoughts? Ideas?

---

Subject: Re: ECB-DMA
Posted by lynchaj on Wed, 28 Apr 2021 15:00:38 GMT
View Forum Message <> Reply to Message

I looking over the DMA controller and some DMA peripherals and am now thinking the IEI/IEO, BAI/BAO, RDY/TC signals are so rarely used that they really don't belong on the backplane bus. They are different than the CPU data, address, and other control lines. Rather instead use flying leads "over the top" for those few boards that actually use interrupt and/or DMA daisy chaining. I think they are only used in rare cases and even in those it would be mostly a single DMA controller and a single DMA peripheral. Possibly more for interrupts but still not a lot.

At least with interrupts, having multiple devices sharing a single interrupt line can usually be dealt with in software in the ISR by doing a round-robin query once the CPU gets the interrupt to determine what sent it and then deal with it appropriately. Maybe a bit slower but not a lot since there aren't all that many devices generating interrupts. Maybe IEI/IEO helps with some prioritization but it seems to be pretty cumbersome dealing with this on the backplane. I am

thinking better to use flying leads and set up the chaining on top of the boards where its visible and easily accessible.

Discussion? Thoughts?

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Thu, 17 Jun 2021 14:14:39 GMT
View Forum Message <> Reply to Message

Finally build :)

1983 DMA chip :)

This will be a real challenge figuring out if this works.

File Attachments
1) ecb-dma-r01a.jpg, downloaded 1184 times

---

Subject: Re: ECB-DMA
Posted by lynchaj on Thu, 17 Jun 2021 14:29:20 GMT
View Forum Message <> Reply to Message

Great news! I am looking forward to finally getting to the bottom of this mystery board. Does it work? If so, what can it do? Very interesting!

Good luck!

Thanks, Andrew Lynch

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sat, 19 Jun 2021 12:33:51 GMT
View Forum Message <> Reply to Message

Findings so far:

I have managed to do a small memory to memory transfer using dma.

The data bus direction logic is not correct. I added an inverter on the 74LS245 direction pin and that was sufficient to getting the memory transfer to work but am unsure if the problem is deeper than that.

Memory Transfer fails after 5MHz clock speed (Using 10Mhz crystal with clock divider)

Memory Transfer fails on copy size greater than ~800 bytes.

Activity LED works.

The board decodes two ports. Base port is for programming the DMA chip. Second port triggers the transfer after programming.

I'm honestly surprised to have gotten this far :)

---

Subject: Re: ECB-DMA
Posted by lynchaj on Sat, 19 Jun 2021 14:11:25 GMT
View Forum Message <> Reply to Message

Hi
Well that's a good start. Most new boards require a generation or two to work out the kinks. That's pretty normal. That you've gotten a memory to memory transfer working is really great though!

Keep plugging away and the bugs will fall out eventually. It would be a great addition to have a DMA board available. I think it would transform a lot of our IO devices

Thanks, Andrew Lynch

---

Subject: Re: ECB-DMA
Posted by lynchaj on Sat, 19 Jun 2021 14:30:15 GMT
View Forum Message <> Reply to Message

I have a question for you on the ECB-DMA. To do the memory to memory transfers, what signals is the board using? Are the IEI-IEO and BAI-BAO being used? Is it the normal Z80 control bus, data lines, and address lines? Are there any special signals being used?

Thanks, Andrew Lynch

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Mon, 21 Jun 2021 14:31:48 GMT
View Forum Message <> Reply to Message

Hi Andrew,

Interrupts are currently not being used for memory/memory copy so there is no activity on IEI-IEO.

However, interrupts can be used as part of memory/memory copy by programming a counter to generate a Pulse/Interrupt signal or generate a pulse / interrupt at end of transfer.

SW2 facilitates connecting CPU BUSAK line to the BAI input and consequently to the BAO ouput. In a single board system the BAI/BAO bud line does not really appear to be needed (IMO)

In a multi board DMA system SW2 would connect configured as above on the first board connect to BUSAK but other boards connect to BAI, so this is where the daisy chain and mainboard links come into play.

Still trying to ger my head around it.

Just the normal Z80 data, address and control signals are used. I can see that using flying lead for dma daisy chain signals would be a reasonable compromise if mainboard lines were scarce.

It's pretty clever, I am surprised they got this right first time. Would be nice if it was multichannel :)

Still so much to learn.

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sun, 27 Jun 2021 02:10:57 GMT
View Forum Message <> Reply to Message

Quick update.

memory transfer issues resolved - unsoldered address line.
some dma setup issues resolved - bad data sheet
system locks up most times when reset requiring power cycling.

going to try plugging in some code into the romwbw hbios.

todo: try some i/o writing.

Regards Phil.

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Fri, 02 Jul 2021 09:06:14 GMT
View Forum Message <> Reply to Message

Hi Everyone,

i/o read and writes work through dma.
have build a dma initialization driver and helper routines for romwbw.
implemented dma block move for sector transfers in memory disk driver.
implemented dma block i/o read and write for ramfloppy driver.

not seeing any performance increase as previous posters predicted but it has added another blikenlight to to system :)

this does appear to be a suitable starting point for dma based projects.

todo: submit drivers to romwbw dev build
see if higher speed dma chips are available
update and post kicad development files and gerbers
consider changing dma to plcc footprint

Best Wishes Phil

---

Subject: Re: ECB-DMA
Posted by lynchaj on Fri, 02 Jul 2021 11:49:00 GMT
View Forum Message <> Reply to Message

Congratulations Phil

This is really great news. I think the memory to memory sector transfers could help all sorts of device drivers. the RAM and ROM drives use a pseudo DMA like transfer with the CPU, LDIR if I recall. A real DMA option is could really open things up.

The IO transfers I think are the big gain though. It will require new drivers to exploit. Possibly new peripheral boards as some may not be suitable for DMA at all. I'd like to see an IDE especially CF that could use DMA and also an FDC. I think the current FDCs are limited to DD only at 4 MHz but a DMA capable FDC could do HD as well.

Is there any hope of using DMA with PPIDE or is the nature of the device just not able to use DMA?

I would certainly like to add DMA to the Z80 MBC. Very, very cool. Great work Phil, much appreciated

Thanks, Andrew Lynch

PS, I would stick with the DIP-40 DMA chip because those are a lot more common than the PLCC version. Is there a CMOS DMA chip?

---

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Fri, 02 Jul 2021 14:23:05 GMT
View Forum Message <> Reply to Message

Thanks Andrew.

Code references:

https://github.com/b1ackmai1er/RomWBW/blob/dev/Source/HBIOS/ dma.asm
https://github.com/b1ackmai1er/RomWBW/blob/dev/Source/HBIOS/ rf.asm
https://github.com/b1ackmai1er/RomWBW/blob/dev/Source/HBIOS/ md.asm

I do not know much about IDE but Bill has some experience
https://www.retrobrewcomputers.org/forum/index.php?t=msg&amp
;th=430&goto=6913&#msg_6913 File links broken unfortunately.

Regards Phil.

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sun, 04 Jul 2021 15:09:52 GMT
View Forum Message <> Reply to Message

Hello,

I have posted revised gerbers and kicad files on the wiki if anyone is interested in experimenting with this board:

https://www.retrobrewcomputers.org/doku.php?id=boards:ecb:dm a:start

I have 3 spare boards which I am happy to post to anyone in australia free of charge if they intend to build it. Requires minor modification.

Best Wishes Phil.

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Mon, 05 Jul 2021 14:35:40 GMT
View Forum Message <> Reply to Message

Well after all the celebration of finalizing all that, I though I would go back and understand and verify the data bus direction logic.

To my embarressment I found that the data bus direction wasn't changing.

So I looked deeper into the issue and compared the circuity against the data sheet. I believe there has been an incorrect logic substitution.

I have patched in a correction using the spare gates and it seems to work, I can see direction changes now. I will confirm if I can do register reads which I hadn't tried before.

If all successful I will post an update.

Best Wishes Phil

File Attachments
1) ecbdmabug.png, downloaded 956 times

Subject: Re: ECB-DMA
Posted by lynchaj on Wed, 07 Jul 2021 10:27:27 GMT
View Forum Message <> Reply to Message

Hi Phillip
Awesome, this is great work and I am looking forward to robust testing of the new DMA board design. I would very much like to port the design over to the Z80 MBC and my plan is the DMA will be the next board after the Z80 FDC. Speaking of Z80 FDC, I am designing the Z80 FDC to be PIO and DMA compatible with the important DMA signals (TEND# and FDC_DMA) being exported as "flying signals" over-the-top. The Z80 DMA board will be an adaptation of the ECB-DMA board with a corresponding connector.

Probably this is not the greatest DMA-FDC design ever but the goal is to demonstrate it working and if successful we can make further adaptations in the inevitable PCB respin cycle. There are already a bunch of DMA related signals in the Z80 FDC design so I am hoping everything is covered. If someone is knows more about DMA and FDCs would please take a look at these features of the Z80 FDC and also the ECB-DMA and post feedback or send me an email.

Again, this is wonderful and its great to see this DMA issue finally getting its due after literally years of discussion. Great job and keep going!

Thanks, Andrew Lynch

PS, oh yeah, what I originally meant to say is these complicated direction logic circuits can drive you nuts trying to figure them out and get it straight. For me it was the bus direction on the original Z80 SBC with all the rules for internal and external IO, memory, etc. More recently, on the Z80 MBC it has been the chip select logic on the Z80 ROM and Z80 RAM were an odd combination that even on two separate boards had to work together. What I did was write up logic truth tables in my notes to get all the scenarios straight and ensure the outcome is what I was expecting and wanted to happen. Actually on the Z80 ROM and Z80 RAM boards I ended up putting the truth tables on the schematics because it was so difficult to explain how those two boards interact. You might consider a similar approach to make your design logic more visible

and enable others to help you.  Just a thought meant to help

Subject: Re: ECB-DMA
Posted by lynchaj on Wed, 07 Jul 2021 10:44:18 GMT
View Forum Message <> Reply to Message

Hi Phillip

Another thought I meant to add but forgot was there are other IO peripherals which would benefit from a working DMA system.  Somewhere earlier in this thread someone posted a partial ECB-SCSI-1 design with DMA support or something which could be adapted for DMA support.  I looked at it a while ago and thought it needed some work but the point is once we have working DMA it will be worthwhile to revive and complete these designs for both PIO and DMA capabilities.

In addition, another project I'd like to see is a revival of the IDE from DiskIO board but this time modified for DMA support.  I think that means a single IO port address so its probably a fairly significant modification but if we had IDE CF with DMA I think it would be dramatically better than PPIDE.  Don't get me wrong, I think PPIDE is great but I don't think it can ever be anything but PIO only unless I am missing something.  Please someone set me straight if I am wrong!  However, I think a DMA capable CF-IDE would much better than current solutions.

Not a project we can do right now but something to consider down the road after the proof-of-concept are demonstrated to work successfully.

Thanks, Andrew Lynch

Subject: Re: ECB-DMA
Posted by etchedpixels on Thu, 08 Jul 2021 15:22:36 GMT
View Forum Message <> Reply to Message

b1ackmai1er wrote on Fri, 02 July 2021 02:06
not seeing any performance increase as previous posters predicted but it has added another blikenlight to to system :)

Raw I/O speed or through the CP/M ? If you are going through CP/M you won't see much because CP/M doesn't scale to modern I/O speeds - not that you can blame DRI for that! CP/M 3 is a bit better than 2.2 and at least tries to do direct to user I/O on bigger transfers if the program knows to do big transfers.

Alan

Subject: Re: ECB-DMA
Posted by b1ackmai1er on Sat, 10 Jul 2021 05:08:53 GMT
View Forum Message <> Reply to Message

Hi,

I have managed to verify that register reads now work and this has facilitated improving the driver to do hardware detection at initialization and return succcess/fail status on transfers.

Have submitted the driver to ROMWBW development build.

In regard to the circuit, is it ok to do this change? Is there anything special about B_IEI that makes this unwise to do?

Best Wishes Phil.

File Attachments
1) dmaieibuffer.png, downloaded 561 times

Subject: Re: ECB-DMA
Posted by lynchaj on Tue, 13 Jul 2021 18:44:49 GMT
View Forum Message <> Reply to Message

Hi
I've stared at these circuit and they seem equivalent to me. The top U1A AND gate is essentially just an input buffer which is the same function as the 74ls244. the U1B AND gate is literally unchanged. Possibly the 74ls244 input has schmitt trigger inputs and the 74ls08 does not? I think if any difference the right hand circuit would perform better with a signal from the bus than the left hand circuit.

If they are not equivalent then they're pretty close, AFAIK. I'd like to see some of the other people's thoughts & opinions on the subject. I learn something most every time.

Thanks, Andrew Lynch

Subject: Re: ECB-DMA
Posted by lynchaj on Thu, 29 Jul 2021 19:15:16 GMT
View Forum Message <> Reply to Message

Hi Phillip
OK, so the Z80 FDC seems to be working from my testing so that's good. Also drew up a Z80 DMA board based on the latest ECB-DMA. A few minor modifications for but nothing that should

affect operations of the DMA logic.  Mostly the "over the top" signal connector from the Z80 FDC and splicing in a few diagnostic LEDs.  I'll post the design over on the Z80 MBC thread.

Thanks, Andrew Lynch

---

Subject: Re: ECB-DMA
Posted by lynchaj on Mon, 14 Mar 2022 10:48:15 GMT
View Forum Message <> Reply to Message

b1ackmai1er wrote on Sun, 10 November 2019 03:16Hi guys,

I have been working on reimplementing Wolfgang Kabtazke's ECB-DMA board.

https://www.retrobrewcomputers.org/doku.php?id=boards:ecb:dm a:start

I beleive this was prototyped but never finalized.

Kicad files have been regenerated and I have just finished routing the board in 10x10 format.

Would anyone be interested in building and debugging this board if I order some?

Regards Phil.

Hi

Well the good news is Phil realized Wolfgang's vision of an ECB-DMA board and it works.  I ported the design over to Z80 MBC and it seems to work there as well.  RomWBW supports DMA and even uses it to a limited extent with the Flash memory drive.  It is nice to see the DMA board blinking away in my Z80 MBC systems.

However, there are still a lot of mysteries associated with this board and several things certainly would benefit from a more detailed explanation.  I have a theory that Wolfgang designed this board based on a magazine article or a book.  I don't see any references in the RBC wiki but this looks like something from the old C'T magazine (German) or something like that.  Maybe an old hobbyist book?

Does anyone recognize this design from something they've read years ago?  Does it look familiar in some way?  It is possible that Wolfgang came up with this design entirely on his own so I don't discount that possibility either but I am guessing he was inspired by something.  I think it would help if we knew what it was and could explain better how this board works.

Any ideas or thoughts on the topic?  Thanks, Andrew Lynch

---

Subject: Re: ECB-DMA
Posted by lynchaj on Wed, 23 Mar 2022 17:05:03 GMT
View Forum Message <> Reply to Message

Hi
Good news regarding the ECB DMA board.  A very close derivative is the Z80 DMA board for Z80 MBC.  Recently we found that the Z80 DMA board is able to do IM2 interrupts and coexist peaceably with other IM2 peripherals.  This is excellent news because if it works on the Z80 DMA board it will likely work with the ECB DMA board as well.  Wolfgang really designed an excellent DMA board.  RomWBW supports Z80 MBC including the Z80 DMA board including IM2 interrupt support.  The debugging software tool (DMAMON) was updated to include interrupt test function.  Big Thanks to Wayne for his excellent work and support for RomWBW with the Z80 MBC

Thanks, Andrew Lynch