G'day Gwyllym,

>I will employ all changes suggested and modify the schematic accordingly.

Great! I presume you are using the PDP-8 method of memory expansion with 256 byte banks. You might want to consider adding a data field (DF) latch in the PDP-8. This allows instructions in one bank being executed while data in another bank are indirectly accessed. I recommend the following format. All use the LDB 1 instruction (write data and reset), where OD0 and OD1 have been swapped. This implies that ID0 and OD0 are the input/output devices, while OD1 is used to program the latches.

```
MA7 MA6
0   0  Writes to input device (ID) latch
1   0  Writes to output device (OD) latch
0   1  Writes to data field (DF) latch
1   1  Writes to address buffer and sets Q FF.
```

I suggest using MA6 instead of MA3 as that allows easier future compatibility. For example, bits MA5..MA0 could be used to address 64 banks of RAM (16 KB total) or 64 I/O. Thus

EL = T0.5*inv(JMP_*JMS_)*Q

Note that inv(JMP_*JMS_) is available from the Decoder board. When EL goes high the address buffer data is written to the instruction field (IF) latch (latch 3), where we define Q as

```
If inv(OD1_RESET_)*MA7*MA6 = 1
 then Q := 1
 else if \EXECUTE
        then Q := 0
        else Q := Q;
```

where \EXECUTE means the falling edge of EXECUTE (I changed this from EL as will be shown later). A multiplexer is used to select between the DF and IF latches using signal SDF (select data field) to give the memory field (MF).

```
If SDF = 0
 then MF := IF
 else MF := DF;
```

SDF would only go high during EXECUTE and for AND I, TAD I, ISZ I and DCA I instructions. That is

SDF = EXECUTE*(AND+TAD+ISZ+DCA)*I

As the value of I is not available during EXECUTE, we need a FF called P. That is

```
If FETCH*T23*(AND+TAD+ISZ+DCA)*I = 1
 then P := 1
 else if \EXECUTE
        then P := 0
        else P := P;
```

We have

FETCH*T23*(AND+TAD+ISZ+DCA)*I = P14*inv(MB7)*MB4

SDF = EXECUTE*P

where P14 is from the Memory board. Alternatively, we can define SIF (select instruction field)

```
SIF := inv(EXECUTE*P);
If SIF := 0
 then MF := DF
 else MF := IF;
```

You can implement the Q and P registers using a 7474 (dual D-Type FF) where

```
CP1 = CP2 = inv(EXECUTE)
CD1 = CD2 = reset signal from Timing board or Multiplexer board
D1 = D2 = 0
Q1 = Q
SD1 = inv(inv(OD1_RESET_)*MA7*MA6)
SD2 = 0
RD2 = inv(P14*inv(MB7)*MB4)
```

For the multiplexer, I realised that an SR FF with two NAND gates could be used. We have for an SR FF

```
Q := inv(Q_*S_)
Q_ := inv(Q*R_*MR_)
```

where S_ is an active low set, R_ is an active low reset, and MR_ is an active low master reset, which we represent this by the function

Q := SR(S_,R_,MR_)

Thus, for the Q and P flip flops, we have

```
Q := SR(inv(inv(OD1_RESET_)*MA7*MA6),inv(EXECUTE*T22.5),MR_)
P := SR(inv(FETCH*T22.5*inv(MB7)*MB4),inv(EXECUTE*T22.5),MR_)
```

where EXECUTE, T22.5, inv(FETCH*T22.5), MB4 and MB7 are from the Decoder Board and MR_ is from the Timer board. Note that

FETCH*T23*(AND+TAD+ISZ+DCA)*I = P14*inv(MB7)*MB4

won't work, as MB4 and MB7 get written over during an indirect address during T23. Also, I forgot to invert P14. If using MB4 and MB7, then the input needs to be

FETCH*T22.5*inv(MB7)*MB4

For Page Zero operation, using inv(P14) = inv(FT23*inv(OPR+IOT)) is OK to use with MB4 to MB7, since the 7495 register is loaded only on a high to low transition of MCP.

```
1) Each SR FF requires a 2-input NAND and a 3-input NAND.
2) inv(inv(EXECUTE*T22.5) uses a 2-input NAND.
3) inv(inv(OD1_RESET_)*MA7*MA6) uses and a 3-input NAND, with inv(OD1_RESET_) already available.
4) inv(FETCH*T22.5*inv(MB7)*MB4) uses and a 3-input NAND, an inverter for inv(FETCH*T22.5) and an inverter for MB7
5) EL := T0.5*inv(JMP_*JMS_)*Q uses and 3-input NAND and an inverter, with inv(JMP_*JMS_) already available.
6) SIF := inv(EXECUTE*P) uses a 2-input NAND.

Total count is

  NAND2 NAND3 INV
1)  2    2    0
2)  1    0    0
3)  0    1    2
4)  0    1    2
5)  0    1    1
6)  1    0    0
 ------------------
    4    5    3
```

The equation for the Q FF is incorrect. It should be

Q := SR(inv(inv(OD1_RESET_)*MA7*MA6),inv(EXECUTE*inv(JMP_*JMS_)*T22.5),MR_)

That is we only reset Q for a JMP or JMS instruction. Thus we have

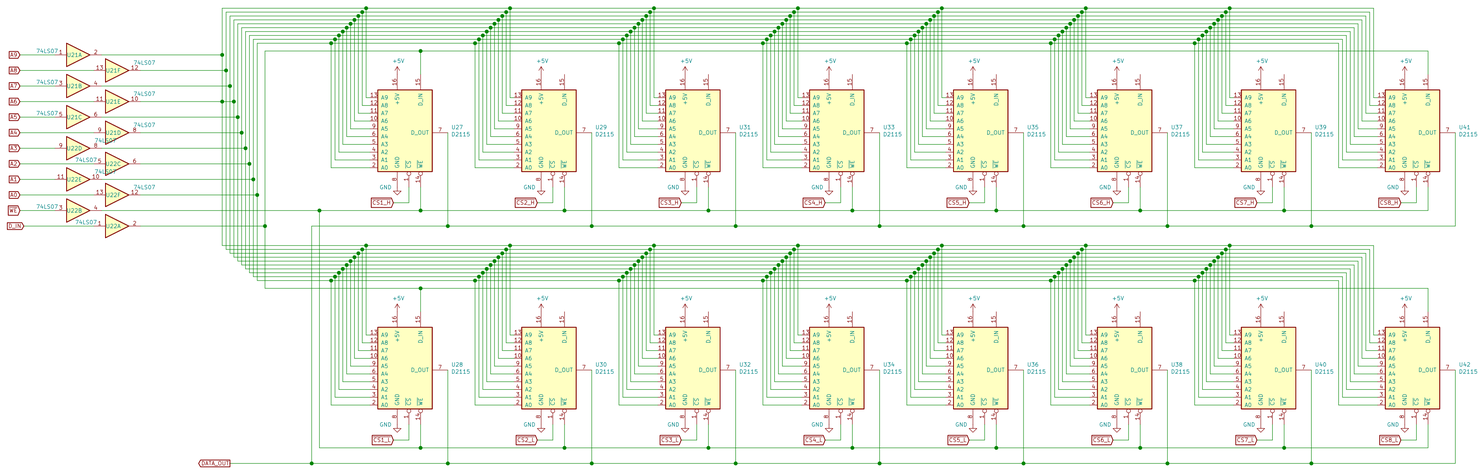
7) inv(EXECUTE*inv(JMP_*JMS_)*T22.5) uses a 2-input NAND, a 2-input NAND and inverter for EXECUTE*inv(JMP_*JMS_), with inv(JMP_*JMS_) already available.

Also, EL can only go high during EXECUTE. That is

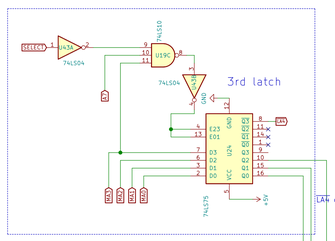
5) EL := EXECUTE*inv(JMP_*JMS_)*0.5*Q uses and 3-input NAND and an inverter, with EXECUTE*inv(JMP_*JMS_) already available from 7).

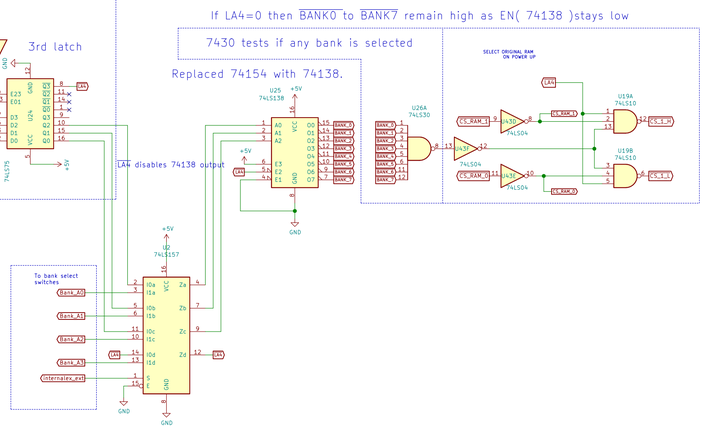
```
  NAND2 NAND3 INV
1)  2    2    0
2)  1    0    0
3)  0    1    0
4)  0    1    2
5)  0    1    1
6)  1    0    0
7)  2    0    1
 ------------------
    6    5    4
```

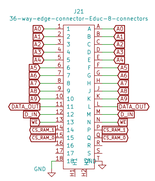
All the best, Steven.



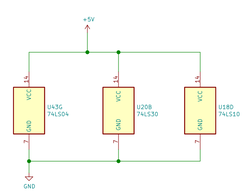
8 X 256 BYTE EXTENDED RAM







INPUT CONECTION
FROM E8/M



2K RAM

Sheet: /RAM/
File: RAM.sch

Title:

Size: A1	Date:	Rev:
KiCad E.D.A. kicad (5.1.10-1-10_14)		Id: 2/2