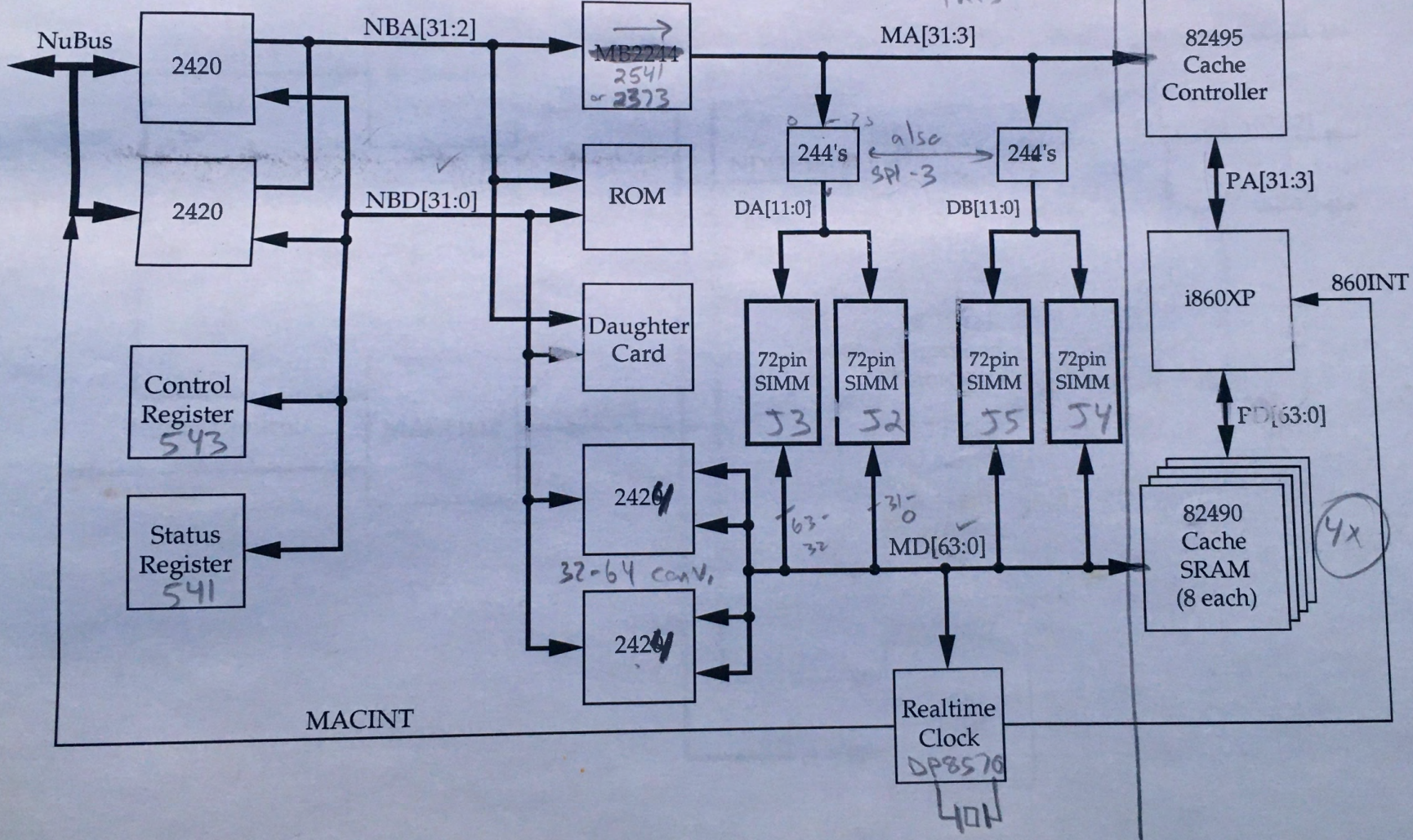
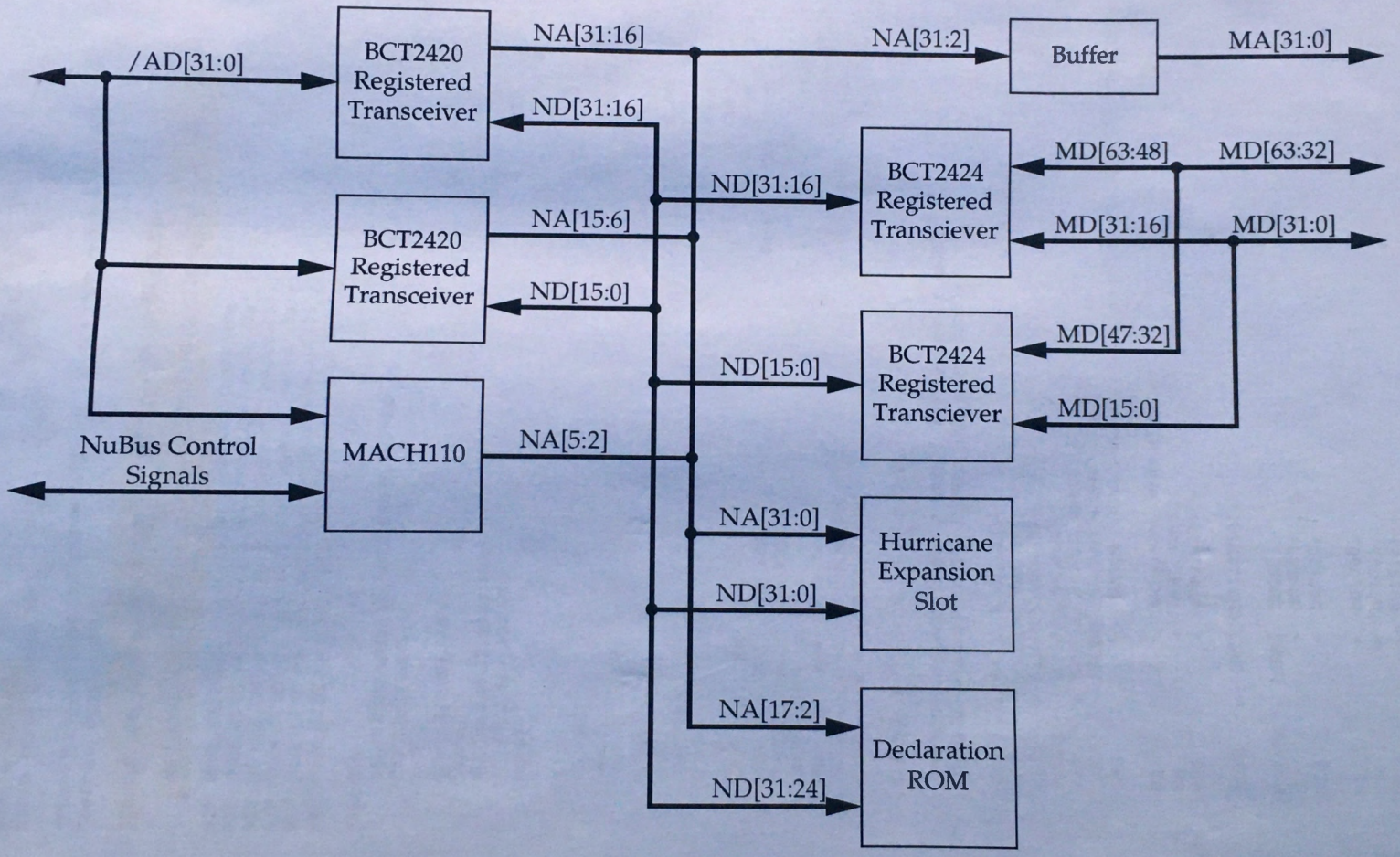
8 March 1993

NuBus Hurricane Block Diagram



NuBus Hurricane
NuBus Interface w/ DMA



"DUMB"

```
module u13dumb
title 'DRAM Controller'

"This PAL generates RAS and WE signals for the DRAM modules and controls
"refresh operations.

"Change MEM_SIZE here to correct memory size in Megabytes!

    MEM_SIZE = 16;                  "Choices are: 4, 8, 16, 32, 64, 128

@IF (MEM_SIZE ==   4) {u13_4m   device 'mach220a';}
@IF (MEM_SIZE ==   8) {u13_8m   device 'mach220a';}
@IF (MEM_SIZE ==  16) {u13_16m  device 'mach220a';}
@IF (MEM_SIZE ==  32) {u13_32m  device 'mach220a';}
@IF (MEM_SIZE ==  64) {u13_64m  device 'mach220a';}
@IF (MEM_SIZE == 128) {u13_128m device 'mach220a';}

"Inputs
    CLK,RESET,MADS,DRAMSEL,NBAEN,REFREQ                 pin 15,51,50,17,54,3;
    CWR,NBRD,NENE                                           pin 37,36,20;
    LCDTS                                                   pin 62;
    MCACHE,MKEN,LLEN                                    pin 10,16,49;
    LCACHE                                                  pin 63;
    MA3,MA4,MA22,MA24,MA26                              pin 29,30,31,32,33;
    PD1,PD2,PD3,PD4,PD5                                     pin 12,14,65,67,6;
"   BURST12                                                     pin 47;

"Outputs
    RREFREQ                                                         pin;
    EARLY                                                       pin 55;
"   TOP                                                             node;
"   DUALBANK                                                    node;
"   LAST                                                        node;
    BURST1P                                                         node;
    BURST2                                                      node;
    BURST4                                                      node;
    CNT0,CNT1,CNT2                                              node istype 'reg_t,buffer';
    DSB0,DSB1,DSB2,DSB3,DSB4                            node istype 'reg_d';
    ROWEN,COLEN,GOCASA,GOCASB                           pin 21,22,56,13;
    CBRREF,DRDY,MA4B                                    pin 57,59,60;
    WEA1,WEA2,WEB1,WEB2                                     pin 26,25,24,23;
    RASLA1,RASLA2,RASHA1,RASHA2                         pin 7,5,4,2;
    RASLB1,RASLB2,RASHB1,RASHB2                         pin 43,45,46,48;

    C,L,H,Z,X     = .C.,0,1,.Z.,.X.;

"   AMDMACH property 'GROUP F RASLB1 RASLB2 RASHB1 RASHB2';
"   AMDMACH property 'GROUP C WEA1 WEA2 WEB1 WEB2';

    RASLA   = [RASLA1,RASLA2];
    RASLB   = [RASLB1,RASLB2];
    RASHA   = [RASHA1,RASHA2];
    RASHB   = [RASHB1,RASHB2];
    RASL    = [RASLA1,RASLA2,RASLB1,RASLB2];
    RASH    = [RASHA1,RASHA2,RASHB1,RASHB2];
    RAS         = [RASLA1,RASLA2,RASLB1,RASLB2,RASHA1,RASHA2,RASHB1,RASHB2];

    WE          = [WEA1,WEA2,WEB1,WEB2];
"   WE          = [WEA1,WEB1];
"   WE2         = [WEA2,WEB2];

    RCNT    = [CNT2,CNT1,CNT0];

    DSTATE  = [DSB4..DSB0];
    S0          = ^b00000;
    S24         = ^b10110;

    S1          = ^b01000;
    S2          = ^b01001;
    S3          = ^b01011;
    S4          = ^b01010;
```

```
      S5            = ^b01110;
      S6            = ^b01100;
      S7            = ^b01101;
      S8            = ^b01111;

      S9            = ^b00100;
      S21           = ^b00101;
      S22           = ^b00111;
      S23           = ^b00010;
      S10           = ^b00011;

      S11           = ^b10010;
      S12           = ^b10011;
      S13           = ^b11011;
      S14           = ^b11010;
      S15           = ^b11110;
      S16           = ^b11100;
      S17           = ^b11101;
      S18           = ^b11111;
      S19           = ^b10111;
      S20           = ^b10101;

      S25           = ^b10100;
      S26           = ^b10000;
      S27           = ^b00001;
      S28           = ^b11000;
      S29           = ^b10001;
      S30           = ^b00110;
      S31           = ^b11001;

"     M512K   =  PD1 & !PD2 &  PD5;         Use this for 'smart' memory controller
"     M2M           =  PD1 &  PD2 &  PD5;
"     M8M           =  PD1 &  PD2 & !PD5;
"     M512K   =  0;                                 Comment out above declarations or use these to
      'fix' memory size
"     M2M           =  0;
"     M8M           =  0;

@IF (MEM_SIZE ==   4) {
      BOTTOM =  MA22 &  MA24 &  MA26              "decode top 4M
          # !MA22 & !MA24 & !MA26;                "and bottom 4M
      TOP    =  0;
      }
@IF (MEM_SIZE ==   8) {
      BOTTOM = !MA22 &  MA24 &  MA26              "decode top 8M
          # !MA22 & !MA24 & !MA26;                "and bottom 8M
      TOP    =  MA22 &  MA24 &  MA26
          #  MA22 & !MA24 & !MA26;
      }
@IF (MEM_SIZE ==  16) {
      BOTTOM =  MA24 &  MA26
          # !MA24 & !MA26;
      TOP    =  0;
      }
@IF (MEM_SIZE ==  32) {
      BOTTOM = !MA24 &  MA26
          # !MA24 & !MA26;
      TOP    =  MA24 &  MA26
          #  MA24 & !MA26;
      }
@IF (MEM_SIZE ==  64) {
      BOTTOM =  1;
      TOP    =  0;
      }
@IF (MEM_SIZE == 128) {
      BOTTOM = !MA26;
      TOP    =  MA26;
      }

      DUALBANK      = (MEM_SIZE ==   8)
                    # (MEM_SIZE ==  32)
```

```
                # (MEM_SIZE == 128);

"    BOTTOM  = M512K & !MA22
"            # M2M   & !MA24
"            # M8M   & !MA26;

     BURST1  =  NBAEN & !LLEN & !CWR & (MCACHE # MKEN)
             #  NBAEN & !LLEN &  CWR &  MCACHE
             # !NBAEN;

     LAST    = 1;

equations

     LCDTS = PD1 & PD2 & PD3 & PD4 & PD5;                    "Dummy equation to make PD[1:5]
     inputs!!!
     LCDTS.OE=0;

     MA22.OE=0;
     MA24.OE=0;
     MA26.OE=0;

"    TOP                  = M512K &  MA22
"                  # M2M   &  MA24
"                  # M8M   &  MA26;

     BURST1P              =  BURST1;
     BURST2          =  NBAEN &  LLEN   & !CWR & (MCACHE # MKEN)
                     #  NBAEN &  LLEN   &  CWR &  MCACHE;
     BURST4          =  NBAEN & !MCACHE &  CWR
                     #  NBAEN & !MCACHE & !CWR & !MKEN;

     DSTATE.clk      = CLK;

     !RREFREQ       := RESET &  !REFREQ
                     # RESET & !RREFREQ & !CBRREF;
      RREFREQ.clk = CLK;

      CBRREF             := RESET & (DSTATE == S0) & !RREFREQ
                          # RESET & (DSTATE == S1)
                     # RESET & (DSTATE == S2)
                     # RESET & (DSTATE == S3)
                     # RESET & (DSTATE == S4)
                     # RESET & (DSTATE == S5)
                     # RESET & (DSTATE == S6)
                     # RESET & (DSTATE == S7)
                     # RESET & (DSTATE == S8) & !LAST;
      CBRREF.clk     = CLK;

      MA4B          :=  MA4  & (DSTATE == S0)
                          #  MA4  & (DSTATE == S24)
                          #  MA4  & (DSTATE == S9)
                          #  MA4  & (DSTATE == S21)
                          #  MA4  & (DSTATE == S22)
                          #  MA4  & (DSTATE == S10)
                          #  MA4  & (DSTATE == S11)
                          #  MA4  & (DSTATE == S12)
                          #  MA4  & (DSTATE == S13)
                          # !MA4  & (DSTATE == S14)
                          # !MA4  & (DSTATE == S15)
                          # !MA4  & (DSTATE == S16)
                          # !MA4  & (DSTATE == S17)
                          # !MA4  & (DSTATE == S18)
                          # !MA4  & (DSTATE == S19)
                          # !MA4  & (DSTATE == S20);
      MA4B.clk       = CLK;

     !ROWEN         := RESET & (DSTATE == S0)
                     # RESET & (DSTATE == S9)                                "Page Miss or
     NuBus Transfer
                     # RESET & (DSTATE == S21);
```

```
ROWEN.clk     = CLK;

COLEN         := RESET & (DSTATE == S0)
                      # RESET & (DSTATE == S9)
                 # RESET & (DSTATE == S21);
COLEN.clk     = CLK;

!WE                 := RESET &  RREFREQ & !MADS & !DRAMSEL & (DSTATE == S0)  &  NBAEN & CWR
                 # RESET &  RREFREQ & !MADS & !DRAMSEL & (DSTATE == S24) &  NBAEN & CWR &
!NENE
                 # RESET &  RREFREQ & !MADS & !DRAMSEL & (DSTATE == S0)  & !NBAEN & NBRD
                 # RESET & !WE      & !MADS & !(DSTATE == S20);
WE.clk        = CLK;

RASL          := !RESET
                 # (DSTATE == S0)
                 # (DSTATE == S24) & !RREFREQ
                      # (DSTATE == S24) & !MADS & !DRAMSEL & (NENE # !NBAEN)
                 # (DSTATE == S1)
                 # (DSTATE == S6)
                 # (DSTATE == S7)
                 # (DSTATE == S8)
                 # (DSTATE == S9)
                 # (DSTATE == S12) & !NBAEN
                 # RASL & !CBRREF & !((DSTATE == S21) & BOTTOM);
RASL.clk      = CLK;

RASH          := !RESET
                 # (DSTATE == S0)
                 # (DSTATE == S24) & !RREFREQ
                      # (DSTATE == S24) & !MADS & !DRAMSEL & (NENE # !NBAEN)
                 # (DSTATE == S1)
                 # (DSTATE == S6)
                 # (DSTATE == S7)
                 # (DSTATE == S8)
                 # (DSTATE == S9)
                 # (DSTATE == S12) & !NBAEN
                 # RASH & !CBRREF & !((DSTATE == S21) & TOP)
                 # !DUALBANK;
RASH.clk      = CLK;

GOCASA        := RESET & (DSTATE == S0) & !RREFREQ
                 # RESET & (DSTATE == S1)
                 # RESET & (DSTATE == S2)
                 # RESET & (DSTATE == S22) & !MA3 & WEA1
                      # RESET & (DSTATE == S10) & !MA3
                      # RESET & (DSTATE == S11) &  MA3 & !BURST1P & WEA1
                      # RESET & (DSTATE == S12) &  MA3 & !BURST1P
                      # RESET & (DSTATE == S14) & !MA3 & !BURST2  & WEA1
                      # RESET & (DSTATE == S15) & !MA3 & !BURST2
                      # RESET & (DSTATE == S16) &  MA3 & WEA1
                      # RESET & (DSTATE == S17) &  MA3
                      # RESET & (DSTATE == S24) & !MA3 & !MADS & !DRAMSEL &  RREFREQ & !NENE
& NBAEN;
GOCASA.clk    = CLK;

GOCASB        := RESET & (DSTATE == S0) & !RREFREQ
                 # RESET & (DSTATE == S1)
                 # RESET & (DSTATE == S2)
                 # RESET & (DSTATE == S22) &  MA3 & WEA1
                      # RESET & (DSTATE == S10) &  MA3
                      # RESET & (DSTATE == S11) & !MA3 & !BURST1P & WEA1
                      # RESET & (DSTATE == S12) & !MA3 & !BURST1P
                      # RESET & (DSTATE == S14) &  MA3 & !BURST2  & WEA1
                      # RESET & (DSTATE == S15) &  MA3 & !BURST2
                      # RESET & (DSTATE == S16) & !MA3 & WEA1
                      # RESET & (DSTATE == S17) & !MA3
                      # RESET & (DSTATE == S24) &  MA3 & !MADS & !DRAMSEL &  RREFREQ & !NENE
& NBAEN;
GOCASB.clk    = CLK;
```

```
!DRDY          := RESET & (DSTATE == S24) & !MADS & !DRAMSEL & !NENE & NBAEN & CWR & RREFREQ
                  # RESET & (DSTATE == S10)
                  # RESET & (DSTATE == S12) & !BURST1
                  # RESET & (DSTATE == S15) & !BURST2
                  # RESET & (DSTATE == S17);
DRDY.clk        = CLK;

CNT0.T          = (DSTATE == S4);
CNT1.T          = (DSTATE == S4) & CNT0;
CNT2.T          = (DSTATE == S4) & CNT0 & CNT1;
RCNT.AR               = (DSTATE == S1);
RCNT.clk        = CLK;

!EARLY          := RESET & !MADS & !DRAMSEL & !NBAEN & (DSTATE == S22);
 EARLY.clk      = CLK;


state_diagram DSTATE

     State S0: "RAS precharge state for new accesses; S24 is idle state for page mode
            IF RESET & !RREFREQ THEN S1
                  ELSE IF !MADS & !DRAMSEL THEN S9
                  ELSE S0;

     State S24: "Idle State for page mode access, RAS lines are asserted
            IF !RESET THEN S0
                  ELSE IF !RREFREQ THEN S0
                  ELSE IF !MADS & !DRAMSEL & (NENE # !NBAEN) THEN S0
                  ELSE IF !MADS & !DRAMSEL & !NENE &  NBAEN & !CWR THEN S10
                  ELSE IF !MADS & !DRAMSEL & !NENE &  NBAEN &  CWR THEN S11
                  ELSE S24;

"RAS Precharge States

     State S9:  "RAS Precharge 2
            IF !RESET THEN S0
                  ELSE S21;

     State S21: "RAS Precharge 3
            IF !RESET THEN S0
                  ELSE S22;

     State S22: "Assert RAS
            IF !RESET THEN S0
                  ELSE S10;

"CAS States, Start here for Page Hits

     State S10: "Switch to Column Address
            IF !RESET THEN S0
                  ELSE S11;

     State S11: "CAS0, DRDY Asserted
            IF !RESET THEN S0
                  ELSE S12;

     State S12: "CAS0 Hold, MBRDY Asserted
            IF !RESET THEN S0
                  ELSE IF !NBAEN  THEN S0
                  ELSE IF      BURST1 THEN S20
                  ELSE S13;

     State S13: "CAS1, DRDY Asserted
            IF !RESET THEN S0
                  ELSE S14;

     State S14: "CAS1 Hold, MBRDY Asserted
            IF !RESET THEN S0
                  ELSE IF     BURST2 THEN S20
                  ELSE S15;
```

```
    State S15: "Toggle MA4B for burst accesses
            IF !RESET THEN S0
                  ELSE S16;

    State S16: "CAS2, DRDY Asserted
            IF !RESET THEN S0
                  ELSE S17;

    State S17: "CAS2 Hold, MBRDY Asserted
            IF !RESET THEN S0
                  ELSE S18

    State S18: "CAS3, DRDY Asserted
            IF !RESET THEN S0
                  ELSE S19;

    State S19: "CAS3 Hold, MBRDY Asserted
            IF !RESET THEN S0
                  ELSE S20;

    State S20: "Done
            IF !RESET THEN S0
                  ELSE S24;
"                 ELSE IF !RREFREQ THEN S0
"                 ELSE IF MADS THEN S24
"                 ELSE S20;

"Refresh States

    State S1: "Refresh, Load Refresh Counter
            IF !RESET THEN S0
                  ELSE S2;

    State S2: "Refresh State 1
            IF !RESET THEN S0
                  ELSE S3;

    State S3: "Refresh State 2
            IF !RESET THEN S0
                  ELSE S4;

    State S4: "Refresh State 3
            IF !RESET THEN S0
                  ELSE S5;

    State S5: "Refresh State 4
            IF !RESET THEN S0
                  ELSE S6;

    State S6: "Refresh State 5
            IF !RESET THEN S0
                  ELSE S7;

    State S7: "Refresh State 6
            IF !RESET THEN S0
                  ELSE S8;

    State S8: "Refresh State 7
            IF !RESET # LAST THEN S0
                  ELSE S2;

"Unused States
    State S23: GOTO S0;
    State S25: GOTO S0;
    State S26: GOTO S0;
    State S27: GOTO S0;
    State S28: GOTO S0;
    State S29: GOTO S0;
    State S30: GOTO S0;
    State S31: GOTO S0;
```

```
test_vectors ([CLK,RESET,!REFREQ,MADS,DRAMSEL,PD1,PD2] ->
    [DSTATE,!RREFREQ,RCNT,CBRREF,RAS,GOCASA,GOCASB])

"1. Refresh
    [C,0,X,1,1,1,1] -> [S0,0,X,0,^b11111111,0,0]; "Reset
    [C,1,0,1,1,1,1] -> [S0,0,X,0,^b11111111,0,0]; "Do Nothing
    [C,1,1,1,1,1,1] -> [S0,1,X,0,^b11111111,0,0]; "Do Nothing, Wait for !RREFREQ
    [C,1,1,1,1,1,1] -> [S1,1,X,1,^b11111111,1,1]; "Refresh State, reset RCNT

"5. Refresh
    [C,1,0,1,1,1,1] -> [S2,0,X,1,^b11111111,1,1]; "Refresh State 1
@IF (DUALBANK) {
    [C,1,0,1,1,1,1] -> [S3,0,X,1,^b00000000,1,1]; "Refresh State 2
    [C,1,0,1,1,1,1] -> [S4,0,X,1,^b00000000,0,0]; "Refresh State 3
    [C,1,0,1,1,1,1] -> [S5,0,X,1,^b00000000,0,0]; "Refresh State 4
    [C,1,0,1,1,1,1] -> [S6,0,X,1,^b00000000,0,0]; "Refresh State 5
}
@IF (!DUALBANK) {
    [C,1,0,1,1,1,1] -> [S3,0,X,1,^b00001111,1,1]; "Refresh State 2
    [C,1,0,1,1,1,1] -> [S4,0,X,1,^b00001111,0,0]; "Refresh State 3
    [C,1,0,1,1,1,1] -> [S5,0,X,1,^b00001111,0,0]; "Refresh State 4
    [C,1,0,1,1,1,1] -> [S6,0,X,1,^b00001111,0,0]; "Refresh State 5
}
    [C,1,0,1,1,1,1] -> [S7,0,X,1,^b11111111,0,0]; "Refresh State 6
    [C,1,0,1,1,1,1] -> [S8,0,X,1,^b11111111,0,0]; "Refresh State 7
    [C,1,0,1,1,1,1] -> [S0,0,X,0,^b11111111,0,0]; "Back to S0


test_vectors ([CLK,RESET,!REFREQ,MADS,DRAMSEL,NBAEN,CWR,NBRD,LCDTS,NENE,MCACHE,MKEN,LCACHE,LLEN,
               MA3,MA4,MA22,MA24,MA26,PD1,PD2,PD5]
        -> [DSTATE,DRDY,!RREFREQ,RCNT,CBRREF,WE,ROWEN,COLEN,MA4B,RAS,GOCASA,GOCASB])

"    C R R M D N C N L N M M L L M M M M M P P P  ->  D  D R R C   W    R C M      R     G G
"    L E E A R B W B C E C K C L A A A A A A D D D      S  R R C B   E    O O A      A     O O
"    K S F D A A R R D N A E A E 3 4 2 2 2 1 2 5      T  D E N R        W L 4      S     C C
"      E R S M E   D T E C N C N     2 4 6           A  Y F T R        E E B            A A
"      T E   S N     S   H   H                      T    R   E        N N              S S
"        Q   E           E   E                      E    E   F                         A B

"13. DRAM cycles from S0
    [C,1,0,1,X,X,X,X,X,X,X,X,X,0,X,X,X,X,X,X] -> [ S0,1,0,X,0,^b1111,0,1,0,^b11111111,0,0];
    "Do Nothing
    [C,1,0,1,X,X,X,X,X,X,X,X,X,X,1,X,X,X,X,X] -> [ S0,1,0,X,0,^b1111,0,1,1,^b11111111,0,0];
    "Do Nothing
    [C,1,0,0,1,X,X,X,X,X,X,X,X,X,0,X,X,X,X,X] -> [ S0,1,0,X,0,^b1111,0,1,0,^b11111111,0,0];
    "Do Nothing
    [C,1,0,1,0,X,X,X,X,X,X,X,X,X,0,X,X,X,X,X] -> [ S0,1,0,X,0,^b1111,0,1,0,^b11111111,0,0];
    "Do Nothing

"17. Cache Write, Burst of 4, Starting Offset = 0
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [ S9,1,0,X,0,^b0000,0,1,0,^b11111111,0,0];
    "RAS Precharge 2
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S21,1,0,X,0,^b0000,0,1,0,^b11111111,0,0];
    "RAS Precharge 3
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S22,1,0,X,0,^b0000,0,1,0,^b00001111,0,0];
    "Assert RASL
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S10,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "Switch Address
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S11,0,0,X,0,^b0000,1,0,0,^b00001111,1,0];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S12,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "CAS0
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S13,0,0,X,0,^b0000,1,0,0,^b00001111,0,1];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S14,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "CAS1
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S15,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
    "MA4B
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S16,0,0,X,0,^b0000,1,0,1,^b00001111,1,0];
    "Wait for valid data
```

```
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S17,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
    "CAS2
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S18,0,0,X,0,^b0000,1,0,1,^b00001111,0,1];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S19,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
    "CAS3
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S20,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
    "Done
    [C,1,0,1,0,1,X,X,X,X,X,X,X,X,0,0,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,1,^b00001111,0,0];
    "Idle State
    [C,1,0,1,0,1,X,X,X,X,X,X,X,X,0,0,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
    "Idle State

"32. Cache Read, Burst of 4, Starting Offset = 0, NENE=0
    [C,1,0,0,0,1,0,X,1,0,0,0,X,X,0,0,0,0,0,X,X,X] -> [S10,1,0,X,0,^b1111,1,0,0,^b00001111,1,0];
    "Switch Address
    [C,1,0,0,0,1,0,X,1,0,0,0,X,X,0,0,0,0,0,X,X,X] -> [S11,0,0,X,0,^b1111,1,0,0,^b00001111,1,0];
    "CAS0
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S12,1,0,X,0,^b1111,1,0,0,^b00001111,0,1];
    "CAS0
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S13,0,0,X,0,^b1111,1,0,0,^b00001111,0,1];
    "CAS1
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S14,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
    "CAS1
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S15,1,0,X,0,^b1111,1,0,1,^b00001111,1,0];
    "MA4B
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S16,0,0,X,0,^b1111,1,0,1,^b00001111,1,0];
    "CAS2
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S17,1,0,X,0,^b1111,1,0,1,^b00001111,0,1];
    "CAS2
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S18,0,0,X,0,^b1111,1,0,1,^b00001111,0,1];
    "CAS3
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S19,1,0,X,0,^b1111,1,0,1,^b00001111,0,0];
    "CAS3
    [C,1,0,0,0,1,0,X,1,X,0,0,X,X,0,0,0,0,0,X,X,X] -> [S20,1,0,X,0,^b1111,1,0,1,^b00001111,0,0];
    "Done
    [C,1,0,1,0,1,X,X,X,X,X,X,X,X,0,0,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,1,^b00001111,0,0];
    "Idle State
    [C,1,0,1,0,1,X,X,X,X,X,X,X,X,0,0,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
    "Idle State

"44. Cache Write, Burst of 4, Starting Offset = 18, NENE=0
    [C,1,0,0,0,1,1,X,1,0,0,0,X,X,1,1,0,0,0,X,X,X] -> [S11,0,0,X,0,^b0000,1,0,1,^b00001111,0,1];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S12,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
    "CAS3
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S13,0,0,X,0,^b0000,1,0,1,^b00001111,1,0];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S14,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
    "CAS2
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S15,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "MA4B
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S16,0,0,X,0,^b0000,1,0,0,^b00001111,0,1];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S17,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "CAS1
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S18,0,0,X,0,^b0000,1,0,0,^b00001111,1,0];
    "Wait for valid data
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S19,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "CAS0
    [C,1,0,0,0,1,1,X,1,X,0,0,X,X,1,1,0,0,0,X,X,X] -> [S20,1,0,X,0,^b0000,1,0,0,^b00001111,0,0];
    "Done
    [C,1,0,1,0,1,X,X,X,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
    "Idle State
    [C,1,0,1,0,1,X,X,X,X,X,X,X,X,0,0,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
    "Idle State

"55. Cache Write, Single Transfer, Starting Offset = 10, NENE=0
    [C,1,0,0,0,1,1,X,1,0,1,0,1,0,0,1,0,0,0,X,X,X] -> [S11,0,0,X,0,^b0000,1,0,1,^b00001111,1,0];
    "Wait for valid data
```

```
     [C,1,0,0,0,1,1,X,1,X,1,0,1,0,0,1,0,0,0,X,X,X] -> [S12,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
     "CAS0
     [C,1,0,0,0,1,1,X,1,X,1,0,1,0,0,1,0,0,0,X,X,X] -> [S20,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
     "Done
     [C,1,0,1,0,1,X,X,X,X,X,X,1,0,0,1,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
     "Idle State
     [C,1,0,1,0,1,X,X,X,X,X,X,1,0,0,0,0,0,0,X,X,X] -> [S24,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
     "Idle State

"    C R R M D N C N L N M M L L M M M M P P P  ->  D D R R C   W    R C M      R     G G
"    L E E A R B W B C E C K C L A A A A D D D      S R R C B   E    O O A      A     O O
"    K S F D A A R R D N A E A E 3 4 2 2 2 1 2 5    T D E N R       W L 4 B    S     C C
"      E R S M E   D T E C N C N     2 4 6          A Y F T R       E E B           A A
"      T E   S N     S   H   H                     T   R   E       N N             S S
"        Q   E           E   E                     E   E   F                       A B

"60. NuBus Read
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [ S0,1,0,X,0,^b1111,1,0,0,^b11111111,0,0];
     "NuBus Transfer
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [ S9,1,0,X,0,^b1111,0,1,0,^b11111111,0,0];
     "RAS Precharge 2
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [S21,1,0,X,0,^b1111,0,1,0,^b11111111,0,0];
     "RAS Precharge 3
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [S22,1,0,X,0,^b1111,0,1,0,^b00001111,0,0];
     "Assert RASL
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [S10,1,0,X,0,^b1111,1,0,0,^b00001111,0,1];
     "Switch Address
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [S11,0,0,X,0,^b1111,1,0,0,^b00001111,0,1];
     "CAS0
     [C,1,0,0,0,0,X,0,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [S12,1,0,X,0,^b1111,1,0,0,^b00001111,0,0];
     "Hold
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [ S0,1,0,X,0,^b1111,1,0,0,^b11111111,0,0];
     "Back to S0
     [C,1,0,1,0,1,X,1,1,X,X,X,X,X,1,0,0,0,0,X,X,X] -> [ S0,1,0,X,0,^b1111,0,1,0,^b11111111,0,0];
     "Stay in S0

"69. NuBus Write
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [ S9,1,0,X,0,^b0000,0,1,1,^b11111111,0,0];
     "RAS Precharge 2
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [S21,1,0,X,0,^b0000,0,1,1,^b11111111,0,0];
     "RAS Precharge 3
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [S22,1,0,X,0,^b0000,0,1,1,^b00001111,0,0];
     "Assert RASL
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [S10,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
     "Switch Address
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [S11,0,0,X,0,^b0000,1,0,1,^b00001111,0,1];
     "CAS0
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [S12,1,0,X,0,^b0000,1,0,1,^b00001111,0,0];
     "Hold
     [C,1,0,0,0,0,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [ S0,1,0,X,0,^b0000,1,0,1,^b11111111,0,0];
     "Back to S0
     [C,1,0,1,0,1,X,1,1,X,X,X,X,X,1,1,0,0,0,X,X,X] -> [ S0,1,0,X,0,^b1111,0,1,1,^b11111111,0,0];
     "Stay in S0


test_vectors ([NBAEN,MKEN,MCACHE,LCACHE,CWR,LLEN] -> [BURST1P,BURST2,BURST4])

"77. BURST signal generation
     [0,X,X,X,X,X] -> [1,0,0];
     [1,1,X,X,X,X] -> [1,0,0];
     [1,0,0,0,0,0] -> [0,0,1];
     [1,0,0,0,0,1] -> [0,0,1];
     [1,0,0,0,1,0] -> [0,0,1];
     [1,0,0,0,1,1] -> [0,0,1];
     [1,0,0,1,0,0] -> [0,0,1];
     [1,0,0,1,0,1] -> [0,0,1];
     [1,0,0,1,1,0] -> [0,0,1];
     [1,0,0,1,1,1] -> [0,0,1];
     [1,0,1,0,0,0] -> [1,0,0];
     [1,0,1,0,0,1] -> [0,1,0];
     [1,0,1,0,1,0] -> [1,0,0];
```

```
        [1,0,1,0,1,1] -> [0,1,0];
        [1,0,1,1,0,0] -> [1,0,0];
        [1,0,1,1,0,1] -> [0,1,0];
        [1,0,1,1,1,0] -> [1,0,0];
        [1,0,1,1,1,1] -> [0,1,0];
end
```

BF39 (not pfld)

```
module u1
title 'Memory Bus Controller'

"This PAL performs all memory bus transactions.  Whenever MADS is asserted,
"a new memory transfer begins.  If NBAEN=0, then it is a NuBus transfer,
"otherwise it is a cache transfer.  Accesses to DRAM are monitored here
"and BRDY's are supplied to the 860 if required.  Read/Write signals for
"the realtime clock are generated here as well as latch signals for NuBus
"transfers.

U1   device  'mach220a';

"Inputs
     CLK,RESET                                                    pin 15,51;
     MADS,MA27,CMIO                                               pin
     13,49,50;
     MCACHE,CWR,LLEN                                              pin 17,4,2;
     CDTS                                                         pin 9;
     NBREQ,DRDY                                                   pin 32,16;
     RDYSRC,NBAEN,NBRD,MKENDIS                                 pin 20,54,5,14;
     LCACHE,LBE1                                                  pin 47,48;
     CDC,FSIOUT,LBE3                                              pin
     11,22,25;
     SELFTEST                                                     pin 30;
     CAHOLD                                                       pin 57;
     EARLY                                                        pin 59;

"Outputs
     RMBRDY                                                       pin;
     BURST1,BURST2,BURST4                                      node;
     RMKENDIS,RMKENDIS1,RMKENDIS2                           node;
     RNBREQ,LCDTS                                              pin 37,67 istype
     'neg';
     FLUSH,SYNC                                                   pin 38,21;
     DRAMSEL                                                            pin
     41;
     MEOC,CNA                                                     pin 12,3
     istype 'neg';
     MDOE,MALE,MSEL,MZBT,MFRZ                               pin 39,28,40,26,43;
     NBRDY,MBRDY,BRDY,CRDY                                     pin 29,24,7,55;
     MKEN,SNPINV                                                  pin 36,46;
     CLKEN,CLKRD,CLKWR                                         pin 33,10,6;
     NBDLE,NBDEN                                                  pin 45,44;
     MSB3,MSB2,MSB1,MSB0                                          node istype
     'reg_d';
     BSB2,BSB1,BSB0                                               node istype
     'reg_d';
     RFSIOUT                                                            pin
     62;
     KWEND,SWEND                                                  pin 23,31;
     RESULT                                                       pin 63;

"    AMDMACH property 'GROUP A MEOC CNA';
"    AMDMACH property 'GROUP D MSB3 MSB2 MSB1 MSB0';

     C,L,H,Z,X     = .C.,0,1,.Z.,.X.;

     MSTATE  = [MSB3..MSB0];
```

```
        S0              = ^b0000;
        S1              = ^b0010;
        S2              = ^b0011;
        S3              = ^b0001;
        S4              = ^b0111;
        S5              = ^b0110;
        S6              = ^b0101;
        S7              = ^b0100;
        S8              = ^b1000;
        S9              = ^b1001;
        S10             = ^b1011;
        S11             = ^b1010;
        S12             = ^b1110;
        S13             = ^b1111;
        S14             = ^b1101;
        S15             = ^b1100;

        BSTATE  = [BSB2..BSB0];
        BS0             = ^b000;
        BS1             = ^b001;
        BS2             = ^b011;
        BS3             = ^b010;
        BS4             = ^b100;
        BS5             = ^b101;
        BS6             = ^b111;
        BS7             = ^b110;

        NUBUS   = !NBAEN & !MADS;
        CACHE   =  NBAEN & !MADS;

        KEN             =  MA27 & !RMKENDIS;
"       KEN             =  NBAEN &  MA27  & CMIO & !MKENDIS;
"       KEN             =  1;

        DRAM     = !NBAEN &  MA27
                 #  NBAEN &  CMIO;
        TIMESEL         = !NBAEN & !MA27
                 #  NBAEN & !CMIO & CDC;
        FLUSHSEL =  NBAEN & !MADS & !CMIO & !CDC;   " & CWR & (!LBE3 # !LBE1);
        Decode any Special Cycle

        RBURST1         =  !LLEN & (LCACHE # !KEN);
        RBURST2         =   LLEN & (LCACHE # !KEN);
        RBURST4         =  !CWR  & !LCACHE &  KEN
                #   CWR  & !LCACHE;

equations

        BURST1  =  NBAEN & !LLEN & !CWR & (MCACHE # !KEN)
                #  NBAEN & !LLEN &  CWR &  MCACHE
                # !NBAEN;
        BURST2  =  NBAEN &  LLEN & !CWR & (MCACHE # !KEN)
                #  NBAEN &  LLEN &  CWR &  MCACHE;
        BURST4  =  NBAEN & !MCACHE &  CWR
                #  NBAEN & !MCACHE & !CWR &  KEN;
"               #  NBAEN &  MCACHE &  CWR & !LCACHE;
```

```
     MALE          = 1;                "WWOR also, indicates strong/weak byte
    ordering mode, high=strong
    !MZBT          := !RESET;          "MX4/MX8 also, should be low during reset
    to indicate 8-bits per 82490
"   !MFRZ          := !RESET;           MEMLDR, low during reset to give high
    current drive for DATA
"   !SYNC          := !RESET;           MEMLDRV low during reset to give high
    current drive for ADDRESS
     MFRZ          := 1;                     "MEMLDRV high during reset to give
    low current drive for DATA
     SYNC          := 1;               "MEMLDRV high during reset to give low
    current drive for ADDRESS

     MZBT.clk      = CLK;
     MFRZ.clk      = CLK;
     SYNC.clk      = CLK;

    !DRAMSEL       := RESET & !NBAEN & MA27
                    # RESET &  NBAEN & CMIO & (!CDTS # !LCDTS) & MEOC;
     DRAMSEL.clk = CLK;

    !MDOE           = NBAEN & CWR
                    # MADS;             "Drive data bus during idle cycles

    MSTATE.clk      = CLK;
    BSTATE.clk      = CLK;

    RMKENDIS1      :=  MKENDIS;
    RMKENDIS2      :=  MKENDIS & RMKENDIS1;
    RMKENDIS       :=  MKENDIS & RMKENDIS1 & RMKENDIS2;

    RMKENDIS1.clk = CLK;
    RMKENDIS2.clk = CLK;
    RMKENDIS.clk  = CLK;

    !RNBREQ             := RESET & !NBREQ;
     RNBREQ.clk     = CLK;

"   !NBRDY         := RESET & !NBAEN & !DRDY
    !NBRDY         := RESET & !NBAEN & !EARLY
                    # RESET & !NBAEN & (MSTATE == S15)
                    # RESET & !NBRDY & !RNBREQ;
     NBRDY.clk      = CLK;

    !FLUSH         := !RESET
                    #  RESET & FLUSHSEL & MEOC;
     FLUSH.clk      = CLK;

    !LCDTS         := RESET & !CDTS
                    # RESET & !LCDTS & MEOC;
     LCDTS.clk      = CLK;

    !MKEN           =  KEN;
"    MKEN.clk       =  CLK;

    !KWEND         := !RESET
                    #  RESET & NBAEN & !MADS  & (MSTATE == S0)
                    #  RESET & NBAEN & !KWEND & MEOC & (MSTATE != S0);
```

```
KWEND.clk     = CLK;

SNPINV              = !NBAEN &  NBRD;

!SWEND        := !RESET
                 #  RESET & NBAEN & !MADS & (MSTATE == S0)
                 #  RESET & NBAEN & !SWEND & MEOC & (MSTATE != S0);
 SWEND.clk    = CLK;

!MSEL         :=  RESET & NBAEN & !MADS & MEOC;   "MTR4/MTR8 also, should
be high during reset to indicate 4 burst cache fills
 MSEL.clk     = CLK;

!CNA          :=  RESET  & NBAEN & !CDTS  & (MSTATE == S5) & !DRDY
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S5) & !DRDY
                 #  RESET &  NBAEN & !CDTS  & (MSTATE == S6) &  CNA
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S6) &  CNA
                 #  RESET &  NBAEN & !CDTS  & (MSTATE == S1) & BURST1 &
!DRDY
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S1) & BURST1 &
!DRDY
                 #  RESET &  NBAEN & !CDTS  & (MSTATE == S15)
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S15)
                 #  RESET &  NBAEN &          (MSTATE == S9) & CAHOLD;
 CNA.clk      = CLK;

!MEOC         :=  RESET &  NBAEN & !CDTS  & (MSTATE == S5) & !DRDY
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S5) & !DRDY
                 #  RESET &  NBAEN & !CDTS  & (MSTATE == S6) &  MEOC
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S6) &  MEOC
                 #  RESET &  NBAEN & !CDTS  & (MSTATE == S1) & BURST1 &
!DRDY
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S1) & BURST1 &
!DRDY
                 #  RESET &  NBAEN & !CDTS  & (MSTATE == S15)
                 #  RESET &  NBAEN & !LCDTS & (MSTATE == S15)
                 #  RESET &  NBAEN &          (MSTATE == S9) & CAHOLD;
 MEOC.clk     = CLK;

!MBRDY        :=  RESET &  NBAEN & !DRDY
                 #  RESET &  NBAEN & (MSTATE == S15)
                 #  RESET & !NBAEN & (MSTATE == S15)
                 #  RESET & !NBAEN & !DRDY;
 MBRDY.clk    = CLK;

!CLKEN        := RESET & (MSTATE == S11)
                 # RESET & !CLKEN  & (MSTATE != S6);
 CLKEN.clk    = CLK;

!CLKRD        := RESET & (MSTATE == S11) &  NBAEN & !CWR
                 # RESET & (MSTATE == S11) & !NBAEN & !NBRD
                 # RESET & !CLKRD  & (MSTATE != S6);
 CLKRD.clk    = CLK;

!CLKWR        := RESET & (MSTATE == S11) &  NBAEN &  CWR
                 # RESET & (MSTATE == S11) & !NBAEN &  NBRD
                 # RESET & !CLKWR  & (MSTATE != S6);
 CLKWR.clk    = CLK;
```

```
!NBDEN          = !NBAEN & NBRD;

!NBDLE         := RESET & !NBAEN & !NBRD & (MSTATE == S1) & !DRDY
                # RESET & !NBAEN & !NBRD & (MSTATE == S15);
 NBDLE.clk      = CLK;

!RMBRDY              := RESET & !MBRDY & (BSTATE == BS1)
                # RESET & !MBRDY & (BSTATE == BS2)
                # RESET & !MBRDY & (BSTATE == BS3);
 RMBRDY.clk     = CLK;

!BRDY          :=  RESET & (BSTATE == BS5) & !MEOC
  "When a FLUSH is requested
                #    RESET & (BSTATE == BS1) & !RMBRDY
  "During Burst of 4
                #  RESET & (BSTATE == BS3) & !RMBRDY
  "During Burst of 2
                #  RESET & (BSTATE == BS4) & !RMBRDY
  "During Burst of 1
                #  RESET & (BSTATE == BS7) & !RMBRDY;
  "At end of all transfers
"               #  RESET & (BSTATE == BS6) &  FSIOUT;
 End of a flush
 BRDY.clk       = CLK;

!CRDY          := !RESET & SELFTEST
        "Invoke self test during reset if SELFTEST asserted
                #  RESET & (BSTATE == BS0) & !MEOC & !RDYSRC
  "Provide CRDY when RDYSRC=0
                #  RESET & (BSTATE == BS5) & !MEOC
  "Provide CRDY during FLUSHes
                #  RESET & (BSTATE == BS6) & !MEOC
  "Provide CRDY during FLUSHes
                #  RESET & (BSTATE == BS4) & !MEOC
                #  RESET & (BSTATE == BS7);
 CRDY.clk       = CLK;

!RFSIOUT       := RESET & !FSIOUT;
 RFSIOUT.clk = CLK;

 RESULT              := RESET & CAHOLD & FSIOUT & !RFSIOUT & FLUSH
  "Latch CAHOLD after a RESET
                      # RESET & RESULT;
 RESULT.clk     = CLK;


state_diagram MSTATE

    State S0: "Idle State
                        IF RESET & !MADS & DRAM THEN S1
"                       IF RESET & !MADS & DRAM & BURST1 THEN S5
"                  ELSE IF RESET & !MADS & DRAM & BURST2 THEN S4
"                  ELSE IF RESET & !MADS & DRAM & BURST4 THEN S2
                   ELSE IF RESET & !MADS & TIMESEL & (!LCDTS # !NBAEN) THEN S11
                   ELSE IF RESET & !MADS & !FLUSH THEN S7
                   ELSE S0;
```

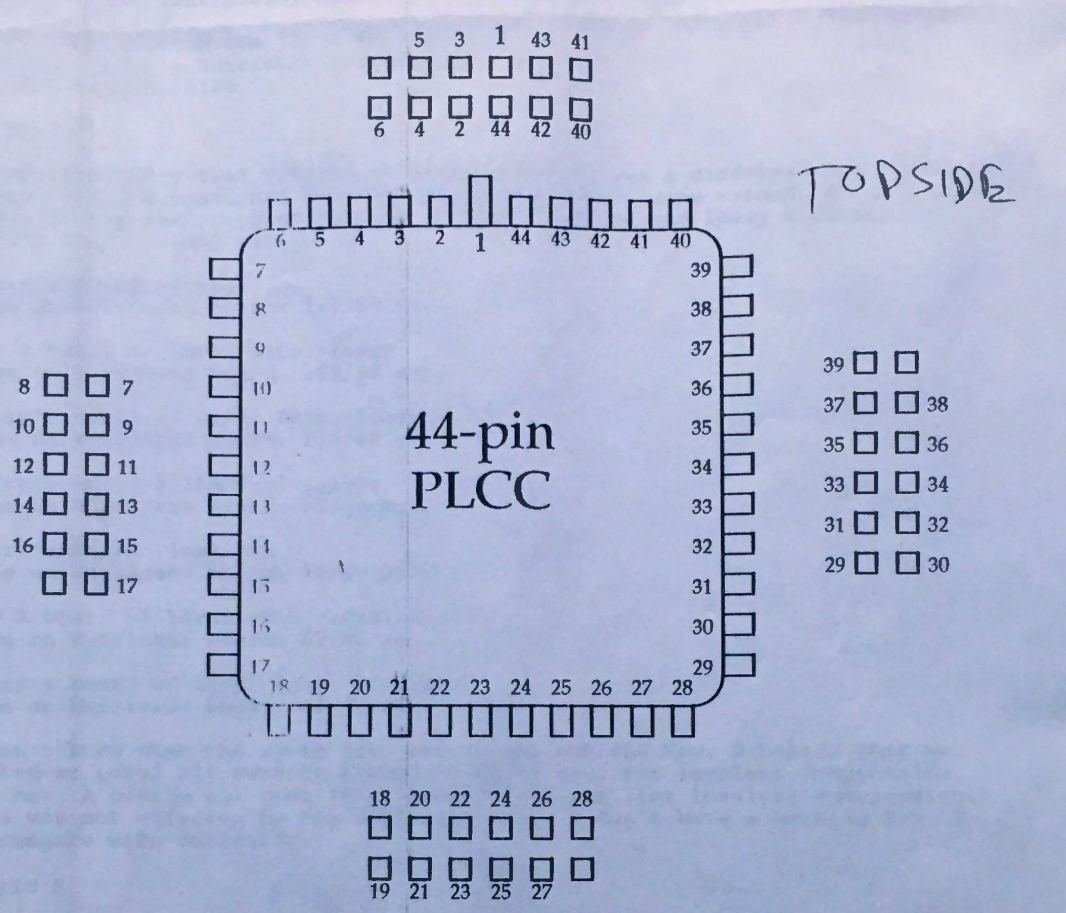
```
State S1: "DRAM Selected, 1st transfer
        IF !RESET THEN S0
              ELSE IF !DRDY & BURST4 THEN S2                    "Burst of 4
              ELSE IF !DRDY & BURST2 THEN S5                    "Burst of 2
              ELSE IF !DRDY & BURST1 THEN S6                    "Single
transfer
              ELSE S1;

State S2: "Burst of 4, 2nd transfer
        IF !RESET THEN S0
              ELSE IF !DRDY THEN S3
              ELSE S2;

State S3: "Burst of 4, 3rd transfer
        IF !RESET THEN S0
              ELSE IF !DRDY THEN S5
              ELSE S3;

State S4: "Burst of 2
              GOTO S0;
"       IF !RESET THEN S0
"             ELSE IF !MBRDY THEN S5
"             ELSE S4;

State S5: "Last transfer
        IF !RESET THEN S0
              ELSE IF !DRDY THEN S6
              ELSE S5;

State S6: "Assert MEOC, Wait for !LCDTS
        IF !RESET THEN S0
              ELSE IF !MEOC # !NBAEN THEN S0
              ELSE S6;

State S7: "Flush request, assert FLUSH
        IF !RESET THEN S0
              ELSE S8;

State S8: "Keep FLUSH Asserted
        IF !RESET THEN S0
              ELSE S9;

State S9: "WAIT FOR CAHOLD=1
        IF !RESET THEN S0
              ELSE IF CAHOLD THEN S6
              ELSE S9;

State S10: GOTO S0;

State S11: "Timer Selected
        IF !RESET THEN S0
              ELSE S12;

State S12: "Wait State 1
        IF !RESET THEN S0
              ELSE S13;

State S13: "Wait State 2
```

```
            IF !RESET THEN S0
                  ELSE S14;

      State S14: "Wait State 3
            IF !RESET THEN S0
                  ELSE S15;

      State S15: "Wait State 4
            IF !RESET THEN S0
                  ELSE S6;


"This state machine watches for MEOC asserted, end of memory cycle, for the
    cache.
"When MEOC is asserted, the state machine checks to see if BRDY's need to be
"provided, BRYSRC=1.  If so, then it provides the correct number of BRDY's to
    the
"CPU, based on BURST1, BURST2, or BURST4.  CDRY is asserted with the last
    BRDY.
"In addition, MEOC above is only asserted when this machine is in State BS0 to
"guarantee pipeline timing relationships, like BRDY,CRDY(N) > MEOC(N+1) + 2
    clocks, etc.
"This would probably only be violated if a cache fill was followed by a non-
    cacheable
"read.  BRDYs would be asserted for 4 cycles, and MEOC would come at the first
    MBRDY.
"If a FLUSH is going on, FSIOUT=0, then goto BS5 to check for the last
    transfer.
"If the last transfer, then assert BRDY.


state_diagram BSTATE

      STATE BS0: "Idle State, Assert CRDY if !RDYSRC and still in BS0
            IF !RESET THEN BS0
                  ELSE IF !MADS & NBAEN & FLUSHSEL          THEN BS5
            "FLUSH
                  ELSE IF !MADS & NBAEN & RDYSRC & RBURST4 THEN BS1
            "Provide 4 BRDY's
                  ELSE IF !MADS & NBAEN & RDYSRC & RBURST2 THEN BS2
            "Provide 2 BRDY's
                  ELSE IF !MADS & NBAEN & RDYSRC & RBURST1 THEN BS3
            "Provide 1 BRDY
                  ELSE BS0

      State BS1: "4 BRDY's, wait for !MEOC
            IF !RESET THEN BS0
                  ELSE IF !MEOC THEN BS7
                  ELSE BS1;

      State BS2: "Burst of 2, 1st BRDY
            IF !RESET THEN BS0
                  ELSE IF !MBRDY THEN BS3
                  ELSE BS2;

      State BS3: "Last BRDY
            IF !RESET THEN BS0
```

```
                ELSE IF !MBRDY & !MEOC THEN BS7                    "!MOEC here
means same length memory cycle
                ELSE IF !MBRDY &  MEOC THEN BS4                    "as BRDY's.
(1 or 2)
                ELSE BS3;

State BS4: "Wait for !MEOC on 4 cycle memory transfers, Assert CRDY when
leaving this state
        IF !RESET THEN BS0
                ELSE IF !MEOC THEN BS0
                ELSE BS4;

State BS7: "Assert CRDY when leaving this state
        IF !RESET THEN BS0
                ELSE BS0;

State BS5: "FLUSH requested, wait for CAHOLD=1
                IF !RESET THEN BS0
                ELSE IF !MEOC THEN BS0
                ELSE BS5;

State BS6:  GOTO BS0;


test_vectors
     ([CLK,RESET,MADS,NBAEN,CMIO,CWR,MA27,MCACHE,LCACHE,LLEN,CDTS,NBREQ,NBRD,D
     RDY,EARLY,RDYSRC,MKENDIS]
          ->
     [MSTATE,LCDTS,RNBREQ,MDOE,MALE,MSEL,MEOC,MBRDY,NBRDY,MKEN,KWEND,SWEND,SNP
     INV])


"     C R M N C C M M L L C N N D E R M  ->   M  L R M M M M M N M K S S
"     L E A B M W A C C L D B B R A D K      S  C N D A S E B B K W W N
"     K S D A I R 2 A A E T R R D R Y E      T  D B O L E O R R E E E P
"       E S E O   7 C C N S E D Y L S N      A  T R E E L C D D N N N I
"       T   N       H H     Q     Y R D      T  S E          Y Y   D D N
"                   E E             C S      E    Q                   V

"1. RESET and do nothing conditions
     [C,0,X,X,X,X,X,X,X,X,X,X,X,X,X,X] -> [ S0,1,1,1,1,1,1,1,1,1,0,0,X];
     "RESET
     [C,1,1,X,X,X,X,X,X,X,1,1,X,1,1,X,X] -> [ S0,1,1,0,1,1,1,1,1,1,1,1,X]; "Do
     Nothing
     [C,1,1,X,X,X,X,X,X,X,0,1,X,1,1,X,X] -> [ S0,0,1,0,1,1,1,1,1,1,1,1,X];
     "Assert LCDTS
     [C,1,1,X,X,X,X,X,X,X,1,0,X,1,1,X,X] -> [ S0,0,0,0,1,1,1,1,1,1,1,1,X];
     "Keep asserted until !MEOC
     [C,1,1,X,X,X,X,X,X,X,1,0,X,1,1,X,X] -> [ S0,0,0,0,1,1,1,1,1,1,1,1,X];
     "Assert RNBREQ
     [C,1,1,X,X,X,X,X,X,X,1,1,X,1,1,X,X] -> [ S0,0,1,0,1,1,1,1,1,1,1,1,X];
     "Keep RNBREQ asserted
     [C,0,X,X,X,X,X,X,X,X,X,X,X,X,X,X] -> [ S0,1,1,1,1,1,1,1,1,1,0,0,X];
     "RESET
     [C,1,1,1,X,X,X,X,X,X,1,1,X,1,1,X,X] -> [ S0,1,1,0,1,1,1,1,1,1,1,1,X]; "Do
     Nothing
```

```
     [C,1,1,X,X,X,X,X,X,X,1,1,X,1,1,X,X] -> [ S0,1,1,0,1,1,1,1,1,1,1,1,X]; "Do
     Nothing

"10. Posted Write, LLEN=0
     [C,1,0,1,1,1,1,1,1,1,0,1,1,X,1,1,0,0] -> [ S1,1,1,0,1,0,1,1,1,0,0,0,X];
     "Single transfer, wait for !DRDY
     [C,1,0,1,1,1,1,1,1,1,0,1,1,X,1,1,0,0] -> [ S1,1,1,0,1,0,1,1,1,0,0,0,X];
     "Wait for !DRDY
     [C,1,0,1,1,1,1,1,1,1,0,1,1,X,0,1,0,0] -> [ S6,1,1,0,1,0,1,0,1,0,0,0,X];
     "Done, wait for !LCDTS
     [C,1,0,1,1,1,1,1,1,1,0,0,1,X,1,1,0,0] -> [ S6,0,1,0,1,0,0,1,1,0,0,0,X];
     "Done
     [C,1,0,1,1,1,1,1,1,1,0,1,1,X,1,1,0,0] -> [ S0,1,1,0,1,1,1,1,1,0,1,1,X];
     "Back to S0
     [C,1,1,1,1,1,1,1,1,1,0,1,1,X,1,1,0,0] -> [ S0,1,1,0,1,1,1,1,1,0,1,1,X];
     "MADS deasserted

"16. Posted Write, LLEN=1
     [C,1,0,1,1,1,0,1,1,1,1,1,X,1,1,0,0] -> [ S1,1,1,0,1,0,1,1,1,1,0,0,X];
     "Burst of 2, Wait for !DRDY
     [C,1,0,1,1,1,0,1,1,1,1,1,X,1,1,0,0] -> [ S1,1,1,0,1,0,1,1,1,1,0,0,X];
     "Wait for !DRDY
     [C,1,0,1,1,1,0,1,1,1,0,1,X,0,1,0,0] -> [ S5,0,1,0,1,0,1,0,1,1,0,0,X];
     "First !DRDY, wait for !DRDY
     [C,1,0,1,1,1,0,1,1,1,1,1,X,1,1,0,0] -> [ S5,0,1,0,1,0,1,1,1,1,0,0,X];
     "Wait for !DRDY
     [C,1,0,1,1,1,0,1,1,1,1,1,X,0,1,0,0] -> [ S6,0,1,0,1,0,0,0,1,1,0,0,X];
     "Done
     [C,1,0,1,1,1,0,1,1,0,1,1,X,1,1,0,0] -> [ S0,1,1,0,1,1,1,1,1,1,1,1,X];
     "Back to S0
     [C,1,1,1,1,1,1,0,1,1,0,1,1,X,1,1,0,0] -> [ S0,1,1,0,1,1,1,1,1,1,1,1,X];
     "MADS deasserted

"     C R M N C C M M L L C N N D E R M  ->   M  L R M M M M M N N M K S S
"     L E A B M W A C C L D B B R A D K      S  C N D A S E B B K W W N
"     K S D A I R 2 A A E T R R D R Y E      T  D B O L E O R R E E E P
"       E S E O   7 C C N S E D Y L S N      A  T R E E L C D D N N N I
"       T   N       H H     Q     Y R D      T  S E          Y Y   D D N
"                   E E             C S      E    Q                    V

"23. Write Back
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,1,1,0,0] -> [ S1,1,1,0,1,0,1,1,1,0,0,0,X];
     "Burst of 4, Wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,1,1,0,0] -> [ S1,1,1,0,1,0,1,1,1,0,0,0,X];
     "Wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,0,1,X,0,1,0,0] -> [ S2,0,1,0,1,0,1,0,1,0,0,0,X];
     "First !DRDY, wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,1,1,0,0] -> [ S2,0,1,0,1,0,1,1,1,0,0,0,X];
     "Wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,0,1,0,0] -> [ S3,0,1,0,1,0,1,0,1,0,0,0,X];
     "Second !DRDY, Wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,1,1,0,0] -> [ S3,0,1,0,1,0,1,1,1,0,0,0,X];
     "Wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,0,1,0,0] -> [ S5,0,1,0,1,0,1,0,1,0,0,0,X];
     "Third !DRDY, wait for !DRDY
     [C,1,0,1,1,1,1,1,0,1,X,1,1,X,1,1,0,0] -> [ S5,0,1,0,1,0,1,1,1,0,0,0,X];
     "Wait for !DRDY
```

```
    [C,1,0,1,1,1,1,0,1,X,1,1,X,0,1,1,0] -> [ S6,0,1,0,1,0,0,0,1,0,0,0,X];
    "Done
    [C,1,0,1,1,1,1,0,1,X,1,1,X,1,1,1,0] -> [ S0,1,1,0,1,1,1,1,1,0,1,1,X];
    "Back to S0
    [C,1,1,1,1,1,1,0,1,X,1,1,X,1,1,1,0] -> [ S0,1,1,0,1,1,1,1,1,0,1,1,X];
    "MADS deasserted

test_vectors ([CLK,RESET,CDTS,NBAEN,CMIO,MA27,MADS] -> [LCDTS,DRAMSEL])

"34. DRAMSEL generation
    [C,1,1,0,0,0,1] -> [1,1];
    [C,1,1,0,0,1,1] -> [1,0];
    [C,1,1,0,1,0,1] -> [1,1];
    [C,1,1,0,1,1,1] -> [1,0];

    [C,1,1,1,0,0,1] -> [1,1];
    [C,1,1,1,0,1,1] -> [1,1];

    [C,0,1,1,1,0,1] -> [1,1];
    [C,1,1,1,1,0,1] -> [1,1];
    [C,1,0,1,1,0,1] -> [0,0];
    [C,1,1,1,1,0,1] -> [0,0];
    [C,0,1,1,1,0,1] -> [1,1];

    [C,0,1,1,1,1,1] -> [1,1];
    [C,1,1,1,1,1,1] -> [1,1];
    [C,1,0,1,1,1,1] -> [0,0];
    [C,1,1,1,1,1,1] -> [0,0];
    [C,0,1,1,1,1,1] -> [1,1];

test_vectors ([NBAEN,MA27,CMIO,MKENDIS,MCACHE,LCACHE,CWR,LLEN] ->
    [BURST1,BURST2,BURST4])

"50. BURST signal generation
    [0,X,X,X,X,X,X,X] -> [1,0,0];          "MKEN=1, non-cacheable
    [1,0,X,X,X,X,X,X] -> [1,0,0];          "MKEN=1, non-cacheable
    [1,1,0,X,X,X,X,X] -> [1,0,0];          "MKEN=1, non-cacheable
    [1,1,1,1,X,X,X,X] -> [1,0,0];          "MKEN=1, non-cacheable
    [1,1,1,0,0,0,0,0] -> [0,0,1];
    [1,1,1,0,0,0,0,1] -> [0,0,1];
    [1,1,1,0,0,0,1,0] -> [0,0,1];
    [1,1,1,0,0,0,1,1] -> [0,0,1];
    [1,1,1,0,0,1,0,0] -> [0,0,1];
    [1,1,1,0,0,1,0,1] -> [0,0,1];
    [1,1,1,0,0,1,1,0] -> [0,0,1];
    [1,1,1,0,0,1,1,1] -> [0,0,1];
    [1,1,1,0,1,0,0,0] -> [1,0,0];
    [1,1,1,0,1,0,0,1] -> [0,1,0];
    [1,1,1,0,1,0,1,0] -> [1,0,0];
    [1,1,1,0,1,0,1,1] -> [0,1,0];
    [1,1,1,0,1,1,0,0] -> [1,0,0];
    [1,1,1,0,1,1,0,1] -> [0,1,0];
    [1,1,1,0,1,1,1,0] -> [1,0,0];
    [1,1,1,0,1,1,1,1] -> [0,1,0];

end
```


5 3 1 43 41
6 4 2 44 42 40
TOPSIDE
6 5 4 3 2 1 44 43 42 41 40
7
8
9
10
11
12
13
14
15
16
17
18 19 20 21 22 23 24 25 26 27 28
39
38
37
36
35
34
33
32
31
30
29
44-pin
PLCC
8 7
10 9
12 11
14 13
16 15
17
39
37 38
35 36
33 34
31 32
29 30
18 20 22 24 26 28
19 21 23 25 27
BACKSIDE

From netcom.com!ekopf Mon Apr 18 15:32:18 1994
Date: Mon, 18 Apr 1994 15:33:31 -0700
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
To: karl@eo.com
From: ekopf@netcom.com (Eric Kopf)
Subject: Loral's Hurricane Performance Issue
Content-Length: 1108

Hi Karl,

I ran Chow-chi's test program on Barry's board, and a different Rev. A board. Barry's board had a 36MHz crystal on it for some reason, so I fixed that. In the test program you can do both lossless and lossy compression. Here's what I found out:

Barry's board -lossy
Time on Hurricane board: 140/60 sec.

Rev A board w/ Loral PALs -lossy
Time on Hurricane board: 139/60 sec.

Barry's board w/ Loral PALs -lossy
Time on Hurricane board: 139/60 sec.

Barry's board @ 36MHz -lossless
Time on Hurricane board: 62/60 sec.

Barry's board -lossless
Time on Hurricane board: 45/60 sec.

Rev A board w/ Loral PALs -lossless
Time on Hurricane board: 45/60 sec.

Barry's board w/ Loral PALs -lossless
Time on Hurricane board: 45/60 sec.

I don't know what the xx/60 sec. refers to, but the Rev. B boards that we tested at Loral all came in around 59-60/60 sec. for lossless compression. The Rev. A boards all come in at about 45/60 sec. for lossless compression. This was not affected by the different PALs. I don't have a working Rev. B to compare with currently.

--Eric K

PS- Doug is back from Hawaii.

From netcom.com!ekopf Wed Apr 13 10:42:57 1994
Date: Wed, 13 Apr 1994 10:43:59 -0700
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
To: karl@eo.com
From: ekopf@netcom.com (Eric Kopf)
Subject: Loral's Rev. A hurricane
Content-Length: 1078

Hey Karl,

This might be useful information for you if you are going to talk to Chaw Chi. I've looked over theirRev. A board pretty thouroughly and here's what I've found:

Board SN#80410-01

PAL Checksums:
u1- 4CA8 (wierd, pfld PAL is either CA3A, BF39, or B86E-senko)
u4- A562 (normal)
u5- 4F31 (normal)
u13- 951F (Smart memory PAL)
u21- 1178 (wierd, most Rev. A boards have 048C)
u24- 281B (normal)

Passes all Hurricane Tester tests except:
Long A5 memory test
At D81B57F4, expected 55555555, got D5555555

I swapped the DRAM and this problem went away.

Passes all pTest tests.

Compiles code ok.

Runs Photoshop Plug-in ok.

I swapped all the PALs that were out of the ordinary (u1, u21) including a 16Mb memory PAL for the Smart memroy PAL. It didn't affect the results at all.

I don't have Chow Chi's testing program, so I can't run his benchmark. Also, I have to ship this board out today to a customer, so I am keeping u1, u13, and u21 from Loral's board so that we can try them in Barry's board at some later time when we have the benchmark software.

--Eric K

From netcom.com!ekopf Tue Apr 12 15:52:55 1994
Date: Tue, 12 Apr 1994 15:53:59 -0700
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
To: CCY@WDL.LORAL.COM
From: ekopf@netcom.com (Eric Kopf)
Subject: Hurricane Test program
Content-Length: 592

Hello Chaw Chi,

Could you send me an image or two that you use with your test program, along with some basic instructions on how to run it to my account here on Netcom. It's easier for me to download this way.

The PAL at location u1 is programmed differently on your Rev. A board. I have left a message with Karl Towsend to see if this might be the cause of the degraded performance/improved reliability. Barry thinks it might be attributed to the RAM on your Rev. A card.

I checked the i860 on the Rev. A board and it is the same version as on all the other Hurricanes.

--Eric K
68000

From netcom.com!ekopf Fri Apr 22 16:26:39 1994
Date: Fri, 22 Apr 1994 16:27:55 -0700
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
To: karl@eo.com
From: ekopf@netcom.com (Eric Kopf)
Subject: Aaaaaaaaa!!!
Content-Length: 964

Hey Karl,

I tried the i860 and cache controller from the Rev. B bords on a Rev. A board, and got the same results with Chaw-chi's test program. I started to pull chips from the newest Rev B card that I have, but the chip puller is mangling the Mach 210's and 110's. Using anything else mangles the sockets.

If you think you want to come in to look at the boards, give Doug a call. I hope this info is helpful:

The two Rev. A boards are installed in the lab mac. They have 16Mb of RAM each. The board from Loral has a Smart Mach220 PAL. The other board has a dumb Mach220 because it's RAM won't work with the Smart PAL. There is a set of 32Mb RAM and a 32Mb mach220 if you want to test with those. The Rev B boards are in a box on the floor in the lab. The lab books are there too. If you boot the machine up you'll find Chaw-chi's test program in a folder along with some instructions on how to run it. There is also a copy of SpectrePrint Pro 3.1.

--Eric K

From netcom.com!ekopf Tue Apr 19 13:23:18 1994
Date: Tue, 19 Apr 1994 13:24:26 -0700
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
To: karl@eo.com
From: ekopf@netcom.com (Eric Kopf)
Subject: Re: Loral's Hurricane Performance Issue
Content-Length: 2319

>Well, let me know when you've tried it with a Rev. B board.
>
>-Karl
From: Eric Kopf (4/19/94)

To: Stan Creasey, Doug Erickson, Franz MacMaster, Carol Mogannam, Bruce Popko, Barry Twycross, James Upton

Situation Normal - All operations operating as expected.

Hurricane status update...
I have gone through all the Hurricanes we have on hand and here is what I've got:

2 Rev. A boards--working; one from Loral, one from Barry- Currently used for testing.
4 Rev. B boards from the first run--all fail.
5 Rev. B boards from the second run--all fail

1 Rev. B board from the second run is at Hashtech. We were short one component when they were first assembled, and this one has never been tested. If we have any luck, we might get one more good Rev. B board out of this run.

> The good news (?) is now all the dead Rev. B boards seem to fail in the same
> way; you can write to the RAM from the Nubus (Mac) side, but the 860 can't
> successfully access the RAM. I don't know what this means. This is a job for
> Karl.

There is an issue now with Loral. They are using the Hurricane for compressing and decompressing images. According to Chow-chi's test program, the Rev. A boards are 30% faster at doing this than the Rev. B boards (of either prodution run). I sent my test results to Karl. He would like to see the results when from a Rev. B board, but I don't have one, so I will try to have him work with Chow-chi on that.

As far as sales go, I wouldn't hold my breath until one of the boards we have is working. Karl can probably give a better opinion on the possibility of fixing these. I also don't have high hopes for the one at Hashtech. If we have more orders, I looks like we'll have to build more Hurricanes.

cc: Karl Townsend
==========

This is a copy of an email I just sent out. I don't have a Rev. B board here that works. Is it possble that you can get details from Chow-chi? he is testing the Hurricane in a 40MHz IIfx, and so am I. I think his test mac has a few more Nubus cards installed, and I don't know what other variables might affect performance, but I don't know where I am going to get a working Rev. B board unless we build some more.

--Eric K

SIMM Presence Detects

											Capacitance (pF)				
Size	Configuration	Addr.	/RAS	/CAS	PD1	PD2	PD3	PD4	PD5	Addr.	Data	/RAS	/CAS	/WE	Notes
256Kx32	8 * 256Kx4	9	2	4	0	NC			NC	68	17	43	29	76	Hitachi*
512Kx32	16 * 256Kx4	9	4	4	NC	0			NC	121	17	48	48	137	Hitachi*
1Mx32	8 * 1Mx4	10	2	4	0	0			NC	68	17	43	29	76	Hitachi*
2Mx32	16 * 1Mx4	10	4	4	NC	NC			NC	121	29	48	48	137	Hitachi*
4Mx32	32 * 4Mx1	11	2	4	0	0			0	(252)	(19)	(128)	(64)	(284)	Samsung
	8 * 4Mx4	11	2	4	0	0			0	68	12	38	24	76	Samsung
8Mx32	16 * 4Mx4	11	4	4	NC	NC			0	136	19	48	48	152	Samsung
								Max. Load:		136	29	48	48	152	
								Typ. Load:		95	17	46	39	107	

	Speed	PD3	PD4
	-60	NC	NC
	-70	0	NC
	-80	NC	0
	-100	0	0
	-120	NC	NC

*Note: When data from both manufacturer's was available, Hitachi has higher values.