```
MODULE DRAM

TITLE 'DRAM Interface - State Machine Version'

" DRAM Interface  EE52 Project


" Description:  Controls DRAM access


" Revision History:
" Will Werst   3/12/2017   Wrote code



" Pins


"      pin   1                  input   unused
"      pin   2                  input   unused
"      pin   3                  input   unused
"      pin   4                  input   unused
"      pin   5                  input   unused
"      pin   6                  input   unused
"      pin   7                  input   unused
"      pin   8                  input   unused
"      pin   9                  input   unused
"      pin   10                 input   unused
"      pin   11                 input   unused
"      pin   12                 input   unused
"      pin   13                 input   unused
"      pin   14                 input   unused
!JTAG_NTRST pin   15;           "JTAG NTRST input
!NReset pin  16;                "Reset input
!NTRST pin   17  ISTYPE 'com';  "TRST output
RAS    pin   18  ISTYPE 'com';  "RAS line
CAS    pin   19  ISTYPE 'com';  "CAS line
!DRAM_WE pin  20  ISTYPE 'com';  "Write Enable Line to DRAM
!R_A_EN pin   21 ISTYPE 'com';  "Buffer row address enable
!C_A_EN pin   22 ISTYPE 'com';  "Buffer column address enable line
"      pin   23                 input   unused
"      pin   24                 input   unused
!NCS   pin   25;                "input chip select
!NWE   pin   26;                "input for whether write access
!NWAIT pin   27  ISTYPE 'reg';  "output for making cpu wait for refresh finish
!MCLK   pin   28;                "main clock input
!RCLK   pin   29;                "refresh clock input
"      pin   13                 output  unused
"      pin   14                 output  unused
"      pin   15                 output  unused
"      pin   16                 output  unused
St0    pin 37      ISTYPE 'reg';  "output  state bit 0
St1    pin 38      ISTYPE 'reg';  "output  state bit 1
St2    pin 39      ISTYPE 'reg';  "output  state bit 2
St3    pin 40      ISTYPE 'reg';  "output  state bit 3
St4    pin 41      ISTYPE 'reg';  "output  state bit 4
RefRqst pin      ISTYPE 'reg';  "refresh request bit
RefInProgress pin   ISTYPE  'reg_SR';   "bit indicating refresh just finished, used to
implement logic for holding nwait one more cycle
RefInProgressDelay pin  ISTYPE  'reg_D';
"      pin   17                 output  unused
"      pin   18                 output  unused



"the states
```

```
StateBits   =  [ St4, St3, St2, St1, St0 ];     " state bits
                                              " state assignments
Idle        =  [   0,   0,   0,   0,   0 ];     " idle state (waiting for a cycle to start)
Access0     =  [   0,   0,   0,   0,   1 ];     " enable row address for Access
Access1     =  [   0,   0,   0,   1,   0 ];     " RAS line low
Access2     =  [   0,   0,   0,   1,   1 ];     " disable row address for Access
Access3     =  [   0,   0,   1,   0,   0 ];     " enable column address for Access
Access4     =  [   0,   0,   1,   0,   1 ];     " CAS line low
Access5     =  [   0,   0,   1,   1,   0 ];     " Wait for data access
Access6     =  [   0,   0,   1,   1,   1 ];     " Wait for data access
Access7     =  [   0,   1,   0,   0,   0 ];     " Wait for data access
Access8     =  [   0,   1,   0,   0,   1 ];     " RAS, CAS line high
Access9     =  [   0,   1,   0,   1,   0 ];     " Pre-charge
Access10    =  [   0,   1,   0,   1,   1 ];     " Pre-charge
Ref0        =  [   0,   1,   1,   0,   0 ];     " CAS low
Ref1        =  [   0,   1,   1,   0,   1 ];     " RAS low
Ref2        =  [   0,   1,   1,   1,   0 ];     " Wait
Ref3        =  [   0,   1,   1,   1,   1 ];     " Wait
Ref4        =  [   1,   0,   0,   0,   0 ];     " Wait
Ref5        =  [   1,   0,   0,   0,   1 ];     " RAS, CAS high
Ref6        =  [   1,   0,   0,   1,   0 ];     " Pre-charge
Ref7        =  [   1,   0,   0,   1,   1 ];     " Pre-charge
Inval0      =  [   1,   0,   1,   0,   0 ];
Inval1      =  [   1,   0,   1,   0,   1 ];
Inval2      =  [   1,   0,   1,   1,   0 ];
Inval3      =  [   1,   0,   1,   1,   1 ];
Inval4      =  [   1,   1,   0,   0,   0 ];
Inval5      =  [   1,   1,   0,   0,   1 ];
Inval6      =  [   1,   1,   0,   1,   0 ];
Inval7      =  [   1,   1,   0,   1,   1 ];     " Invalid state
Inval8      =  [   1,   1,   1,   0,   0 ];     " Invalid state
Inval9      =  [   1,   1,   1,   0,   1 ];     " Invalid state
Inval10     =  [   1,   1,   1,   1,   0 ];     " Invalid state
Inval11     =  [   1,   1,   1,   1,   1 ];     " Invalid state


EQUATIONS


" NWAIT equation
NWAIT := (NCS & (RefInProgress));


NTRST = NReset & JTAG_NTRST;
DRAM_WE = NWE & NCS;


" clocks for the registered outputs (state bits)
StateBits.CLK  =  MCLK;        " use the global clock pin
NWAIT.CLK = !MCLK;

RefInProgress.S = (StateBits == Ref0);
RefInProgress.R = (StateBits == Idle);
RefInProgress.CLK = MCLK;
RefInProgressDelay.D = RefInProgress;
RefInProgressDelay.CLK = MCLK;
RefRqst.CLK = RCLK;
RefRqst.AR = (StateBits == Ref7);
RefRqst := 1;

STATE_DIAGRAM  StateBits        " a Mealy state machine


STATE  Idle:                    " in the idle state waiting for an access
```

```
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN Idle;
   ELSE IF (RefRqst) THEN  Ref0;
   "ELSE IF (NCS & NWE) THEN Write0;
   ELSE IF (NCS) THEN      Access0;
   ELSE                       Idle;     " otherwise just stay here

STATE Access0:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 1;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access1;

STATE Access1:

   RAS = 0;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 1;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access2;

STATE Access2:

   RAS = 0;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access3;

STATE Access3:

   RAS = 0;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access4;

STATE Access4:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access5;
```

```
STATE Access5:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access6;

STATE Access6:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access7;

STATE Access7:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access8;


STATE Access8:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access9;

STATE Access9:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access10;


STATE Access10:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;
```

```
   GOTO Idle;

STATE Ref0:

   RAS = 1;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                    Ref1;

STATE Ref1:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                    Ref2;


STATE Ref2:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                    Ref3;

STATE Ref3:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                    Ref4;


STATE Ref4:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                    Ref5;


STATE Ref5:


   RAS = 1;
```

```
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                   Ref6;

STATE Ref6:


   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                   Ref7;

STATE Ref7:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                   Idle;

STATE Inval0:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                   Idle;

STATE Inval1:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                   Idle;

STATE Inval2:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                   Idle;

STATE Inval3:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                   Idle;
```

```
STATE Inval4:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval5:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval6:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval7:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval8:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval9:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval10:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval11:
   RAS = 1;
```

```
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;



TEST_VECTORS

( [  MCLK,  RCLK, NReset,  NCS,  NWE ] -> [   RAS, CAS, DRAM_WE, R_A_EN, C_A_EN, NWAIT,
RefRqst, St0, St1, St2, St3, St4 ] )

  [     0,     0,      0,    0,   0  ] -> [   .X., .X.,     .X.,    .X.,    .X.,   .X.,
  .X., .X., .X.,  .X., .X., .X. ];


" reset the system
  [   .C.,     0,      1,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  .X., .X., .X.,  .X., .X., .X. ];
  [   .C.,     0,      1,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  .X., .X., .X.,  .X., .X., .X. ];
  [   .C.,     0,      1,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  .X., .X., .X.,  .X., .X., .X. ];

" Single Access
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access0
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access1
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access2
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access3
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access4
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access5
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle

" Single write
  [   .C.,     0,      0,    1,   1  ] -> [     1,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write0
  [   .C.,     0,      0,    1,   1  ] -> [     0,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write1
  [   .C.,     0,      0,    1,   1  ] -> [     0,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write2
  [   .C.,     0,      0,    1,   1  ] -> [     0,   1,       0,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write3
  [   .C.,     0,      0,    1,   1  ] -> [     0,   0,       0,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write4
  [   .C.,     0,      0,    1,   1  ] -> [     0,   0,       0,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write5
  [   .C.,     0,      0,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
```

```
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle

" Consecutive Accesss
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,       1,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access0
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,       1,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access1
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access2
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,       0,       1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access3
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,       0,       1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access4
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,       0,       1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access5
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,       1,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access0
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,       1,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access1
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access2
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,       0,       1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access3
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,       0,       1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access4
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,       0,       1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access5
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle

" Refresh
  [     0,   .C.,      0,    0,   0  ] -> [   .X., .X.,     .X.,     .X.,     .X.,   .X.,
  1, .X., .X.,  .X., .X., .X. ]; "Trigger the refresh request
  [   .C.,     0,      0,    0,   0  ] -> [     1,   0,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref0
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref1
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref2
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref3
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref4
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref5
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,       0,       0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Ref7
```

```
  [    .C.,      0,       0,     0,    0  ] -> [     1,    1,       1,       0,       0,      0,
  0,    0,    0,     0,    0,    0 ]; "Idle


END DRAM
```