**SDRAM CONTROLLER DESIGN**

Submitted in partial fulfillment of

EE 5313 Microprocessor Systems

**by**

**Prudhvi Adusumilli - 1000860735**

**Moyeedul Chowdhury - 1000857464**

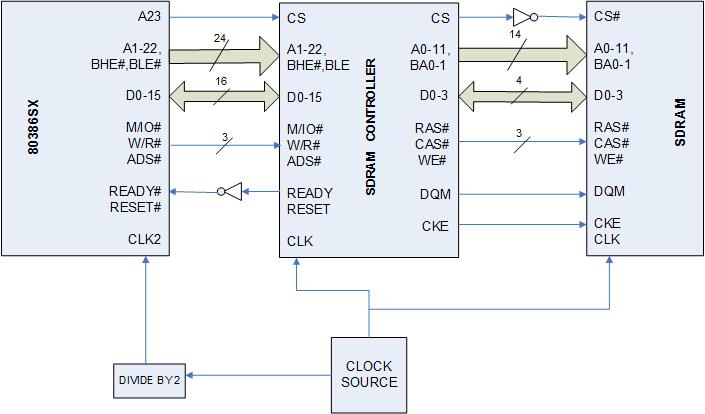
**Patanjalikumar Joshi - 1000858838**

**1. OVERVIEW**

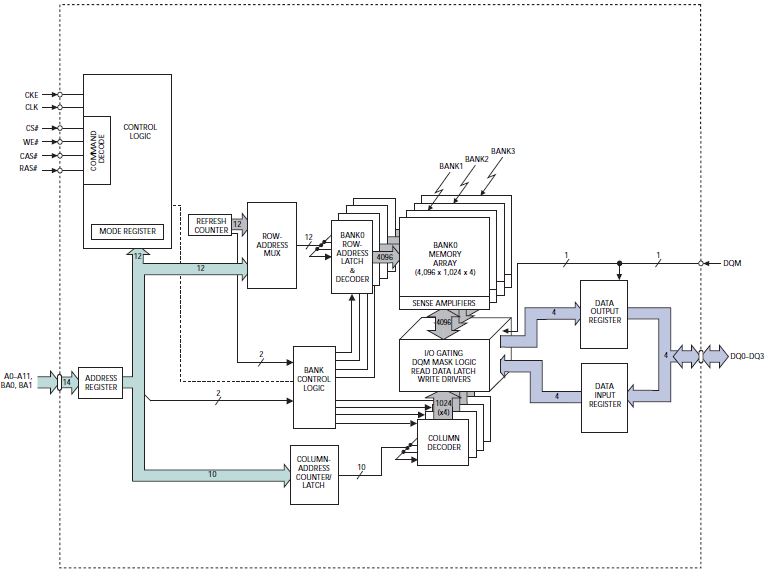
This report presents an SDRAM controlloer design that interfaces the MT48LC16M4A2 SDRAM memory with the 80386SX processor. This report contains the complete design, schematics and theory of operation.

Figure 1 shows the overall block diagram of the scope of our project where the basic system and interface blocks between the processor, the SDRAM controller, and the SDRAM are shown. The clock source shown is a 133 MHz clock which is directly used to clock the SDRAM controller and the SDRAM. The clock source is divided by 2 to clock the processor at 66.5 MHz.

Figure 2 shows the functional block diagram of 16Meg x 4 SDRAM.

****

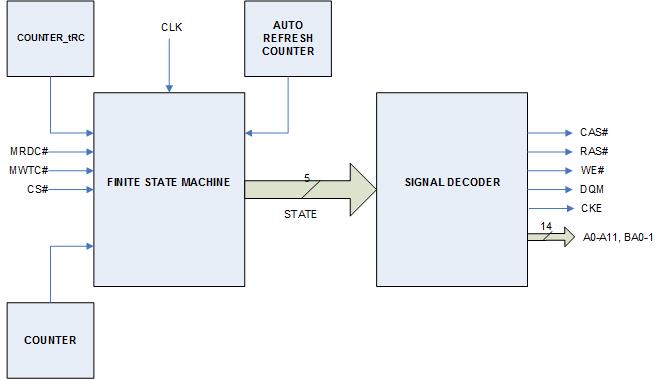
**Figure 1:** SDRAM Controller interfaced with the 80386SX processor and MT48LC16M4A2 SDRAM memory

 **Figure 2**: Functional Block Diagram 16Meg x 4 SDRAM

**2. SDRAM CONTROLLER FUNCTIONAL BLOCK DIAGRAM**

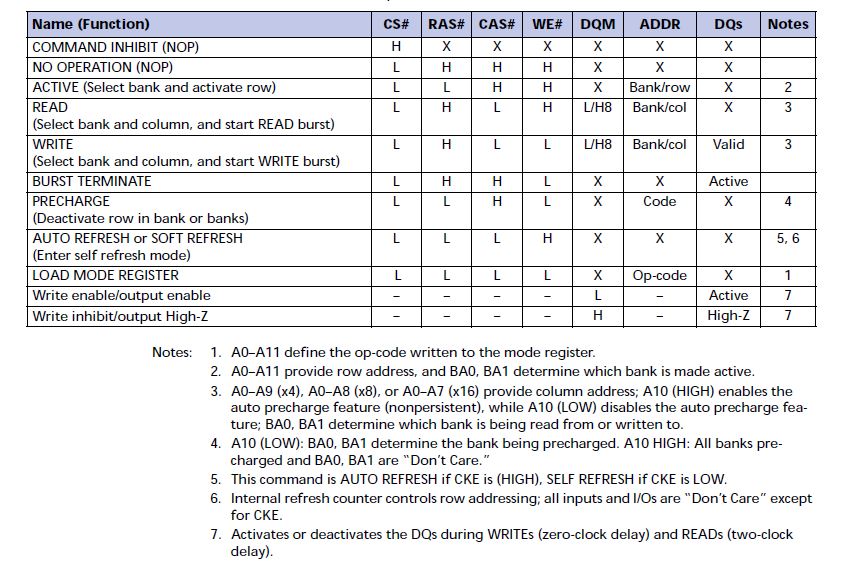
The finite state machine (FSM), explained in detail in later sections, provide the current state of the controller, which is used to decode the output signals used to control the SDRAM memory as shown overleaf in figure3.

The inputs to the FSM are the clock, control signals from the processor and the counters used such as the Auto Refresh Counter, Counter\_tRC and the programmable counter.

****

**Figure 3:** Finite State Machine and Signal Decoding Circuit

The command table shown below is used by the SDRAM Controller Signal Decoding Unit to generate the RAS#, CAS#, WE# signals to issue the particular commands needed in the various states of the FSM.



**Figure 4**: Commands and DQM operation -Truth Table

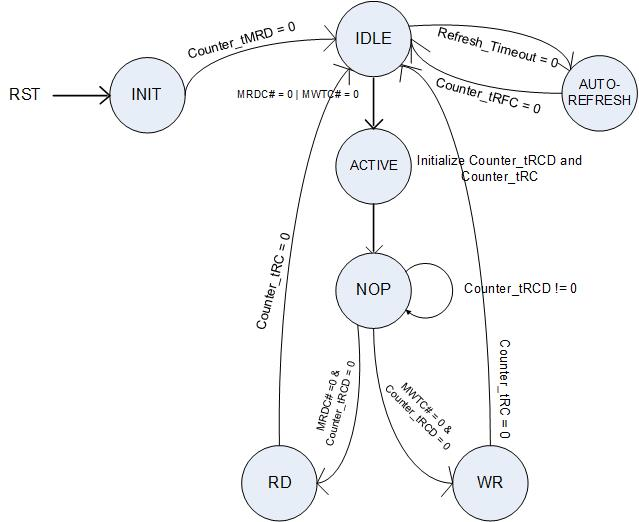
**3. FINITE STATE MACHINE**

The SDRAM Controller FSM consists of the Main state machine and 4 sub-state machines INIT, AUTO-REFRESH, RD, and WR. The main state machine and each of the 4 sub-state machines are explained in detail in the following pages.

**3.1 MAIN STATE DIAGRAMS**

On power-up, the SDRAM Controller goes to the INIT state, which initializes the SDRAM memory and then it stays in the IDLE state. Once Refresh\_Timeout = 0: the Auto-Refresh sub-state machine is executed. Back in the IDLE state, once MRDC# or MWTC# control signal is received from the processor, it goes to the ACTIVE state followed by a NOP and the RD sub-state machine or the WR sub-state machine is executed.

The state diagram, the state transition table, the output signals table, and the comments are shown in figure 5, table 1, table 2, and table 3.



**Figure 5:** Main State Diagram

|  |  |  |
| --- | --- | --- |
| **CURRENT STATE** | **CONDITIONS** | **NEXT STATE** |
| X | RESET | INIT |
| INIT | Couner\_tMRD = 0 | IDLE |
| IDLE | Refresh\_Timeout = 0 | AUTO-REFRESH |
| AUTO-REFRESH | Couner\_tRFC = 0 | IDLE |
| IDLE | MRDC# = 0 | MWTC #= 0 | ACTIVE |
| ACTIVE | (next clock) | NOP |
| NOP | Counter\_tRCD!=0 | NOP |
| NOP | MRDC# = 0 & Counter\_tRCD = 0 | RD |
| NOP | MWTC# = 0 & Counter\_tRCD = 0 | WR |
| RD | Counter\_tRC = 0 | IDLE |
| WR | Counter\_tRC = 0 | IDLE |

**Table 1:** Main State transition table

|  |  |
| --- | --- |
| **OUTPUT** | **EQUATION** |
|  |  |
| CKE | STATE == ALL STATES (except RESET) |
| RAS# | STATE == NOP |
| CAS# | STATE == ACTIVE | NOP |
| WE# | STATE == ACTIVE | NOP |
| BA1 | STATE == ACTIVE |
| BA0 | STATE == ACTIVE |
| A[11:0] | STATE == ACTIVE |
| READY | STATE == IDLE |

**Table 2:** Main State Output Signals

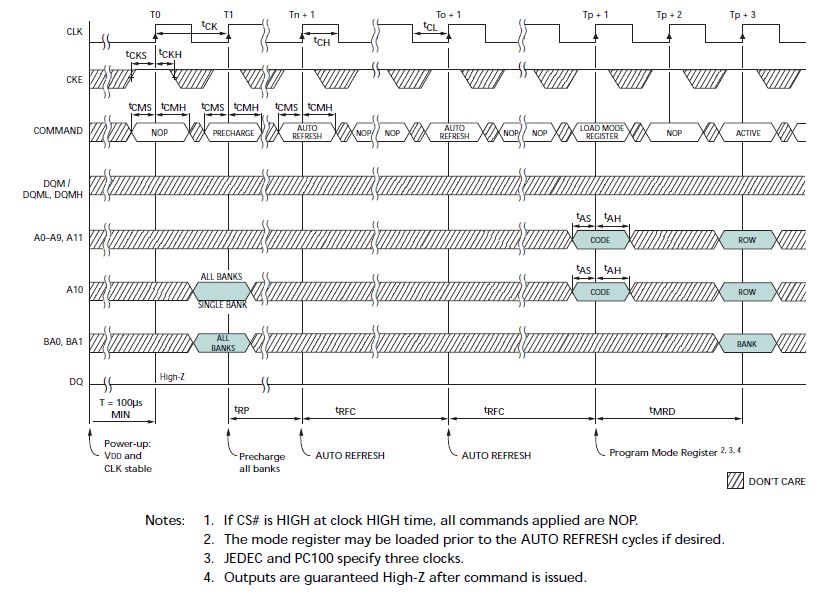
|  |  |  |
| --- | --- | --- |
| # | **STATE** | **COMMENTS** |
| 1 | INIT | i) INIT state machine |
| 2 | IDLE | i) Issue READY signal |
| 3 | AUTO-REFRESH | i) AUTO-REFRESH state machine |
| 4 | ACTIVE | i) Issue ACTIVE command, ADDR = Bank/row |
|  |  | ii) Initialize Counter\_tRCD, Counter\_Trc |
| 5 | NOP | i) Issue NOPs and wait until Counter\_tRCD = 0 |
| 6 | RD | i) READ state machine |
| 7 | WR | i) WRITE state machine |
|  | | |
| \* Before sending READY to the processor: READY signal is inverted since it is an ACTIVE LOW signal for the processor. | | |

**Table 3:** Main State Comments

**3.2 INIT STATE DIAGRAM**

Initialization timing diagram shown in figure 6 and the initialization sequence for the SDRAM memory shown in figure 7 were used to implement the INIT state.

The state diagram, the state transition table, the output signals table, and the comments are shown in figure 9, table 4, table 5, and table 6 respectively.

****

**Figure 6:** Initialization -Timing Diagram



**Figure 7:** Initialization Steps for SDRAM Memory

The Mode Register Definition shown in the below figure has been loaded with the following values :

Burst Length = 4 : 0 1 0

BT = Sequential : 0

CAS Latency = 2 : 0 1 0

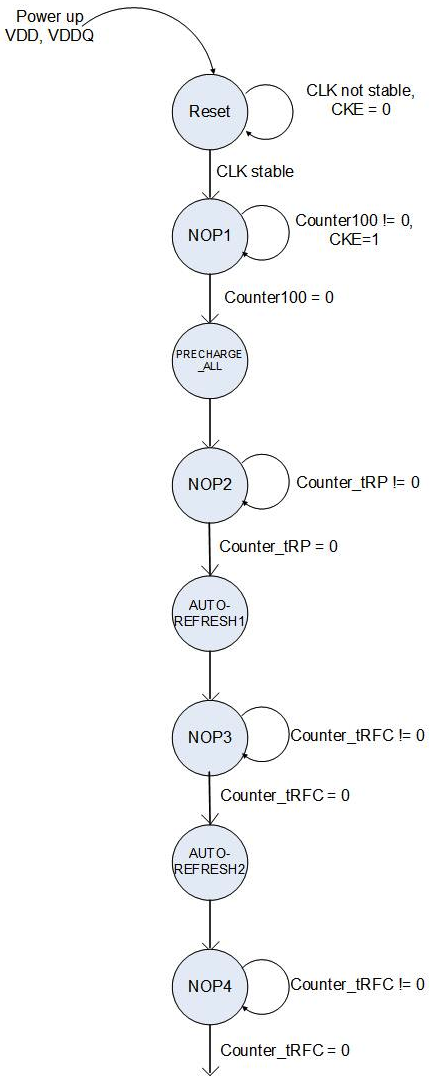
Op Mode = Standard Operation : 0 0

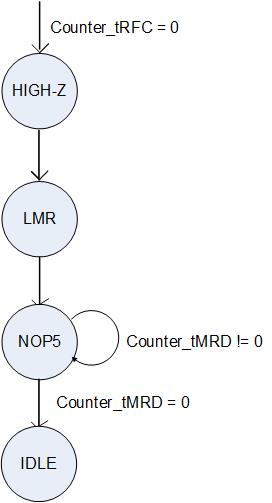
WB = Programmed Burst Length : 0

Reserved = 0 0

****

**Figure 8:** Mode Register Definition





**Figure 9:** INIT State Diagram

|  |  |  |
| --- | --- | --- |
| **CURRENT STATE** | **CONDITIONS** | **NEXT STATE** |
| X | RESET# = 0 | RESET |
| RESET | CLK STABLE | NOP1 |
| NOP1 | Counter100 !=0 | NOP1 |
| NOP1 | Counter100 =0 | PRECHARGE\_ALL |
| PRECHARGE\_ALL | (next clock) | NOP2 |
| NOP2 | Counter\_tRP!=0 | NOP2 |
| NOP2 | Counter\_tRP=0 | AUTO-REFRESH1 |
| AUTO-REFRESH1 | (next clock) | NOP3 |
| NOP3 | Counter\_tRFC!=0 | NOP3 |
| NOP3 | Counter\_tRFC=0 | AUTO-REFRESH2 |
| AUTO-REFRESH2 | (next clock) | NOP4 |
| NOP4 | Counter\_tRFC!=0 | NOP4 |
| NOP4 | Counter\_tRFC=0 | HIGH-Z |
| HIGH-Z | (next clock) | LMR |
| LMR | (next clock) | NOP5 |
| NOP5 | Counter\_tMRD!=0 | NOP5 |
| NOP5 | Counter\_tMRD =0 | IDLE |

**Table 4:** INIT State Transition table

|  |  |
| --- | --- |
| **OUTPUT** | **EQUATION** |
| CKE | STATE == NOPn | PRECHARGE\_ALL | AUTO-REFRESH1 | |
|  | AUTO-REFRESH2 | LMR | IDLE |
| RAS# | NOPn |
| CAS# | STATE == PRECHARGE\_ALL | NOPn |
| WE# | STATE == AUTO-REFRESH1 | AUTO-REFRESH2 | NOPn |
| A[11:0] | STATE == LMR |
| A10 | STATE == PRECHARGE\_ALL | LMR |
| DQM | STATE == HIGH-Z |
| BA1 | !(STATE == LMR) |
| BA0 | !(STATE == LMR) |
|  | |
| n = 1,2,3,4,5 | |

**Table 5:** INIT State Output Signals

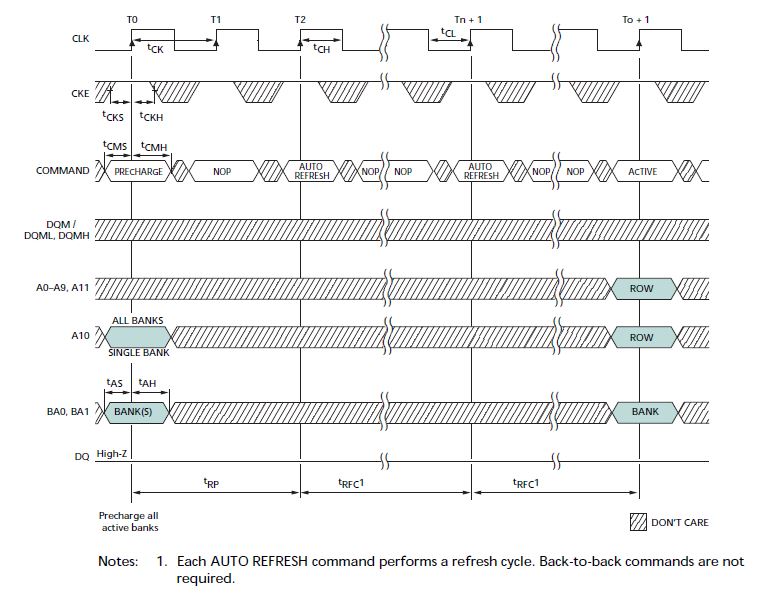
|  |  |  |
| --- | --- | --- |
| # | **STATE** | **COMMENTS** |
| 1 | RESET | i) This state is reached on power up when VDD and VDDQ are applied |
|  |  | ii) Hold CKE low |
| 2 | NOP1 | i) Issue NOPs and wait until Counter100 = 0 |
|  |  | ii) At some point in NOP1, bring CKE high |
| 3 | PRECHARGE\_ALL | i) issue PRECHARGE ALL command |
| 4 | NOP2 | i) Issue NOPs and wait until Counter\_tRP = 0 |
| 5 | AUTO-REFRESH1 | i) Issue AUTO REFRESH command |
| 6 | NOP3 | i) Issue NOPs and wait until Counter\_tRFC = 0 |
| 7 | AUTO-REFRESH2 | i) Issue AUTO REFRESH command |
| 8 | NOP4 | i) Issue NOPs and wait until Counter\_tRFC = 0 |
| 9 | HIGH-Z | i) Issue HIGH-Z command |
| 10 | LMR | i) Program Mode Register with BA1,B01 = 0, ADDR = Op-code |
| 11 | NOP5 | i) Issue NOPs and wait until Counter\_tRFC = 0 |
| 12 | IDLE |  |

**Table 6:** INIT State Comments

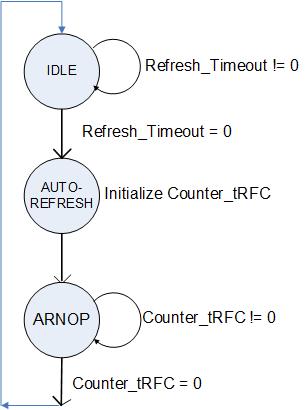
**3.3 AUTO-REFRESH STATE DIAGRAM**

The Auto-Refresh timing diagram of figure 10 was used to implement the sub-state machine AUTO-REFRESH.

The state diagram, the state transition table, the output signals table, and the comments are shown in figure 11, table 7, table 8, and table 9.

****

**Figure 10:** Auto-Refresh mode- Timing diagram

****

**Figure 11:** Auto-Refresh State Diagram

|  |  |  |
| --- | --- | --- |
| **CURRENT STATE** | **CONDITIONS** | **NEXT STATE** |
| IDLE | Refresh\_Timeout = 0 | AUTO-REFRESH |
| AUTO-REFRESH | (next clock) | ARNOP |
| ARNOP | Counter\_tRFC != 0 | ARNOP |
| ARNOP | Counter\_tRFC = 0 | IDLE |

**Table 7:** Auto-Refresh State Transition table

|  |  |
| --- | --- |
| **OUTPUT** | **EQUATION** |
| CKE | STATE == AUTO-REFRESH | ARNOP |
| RAS# | STATE == ARNOP |
| CAS# | STATE == ARNOP |
| WE# | STATE == AUTO-REFRESH | ARNOP |

**Table 8:** Auto-Refresh State Output Signals

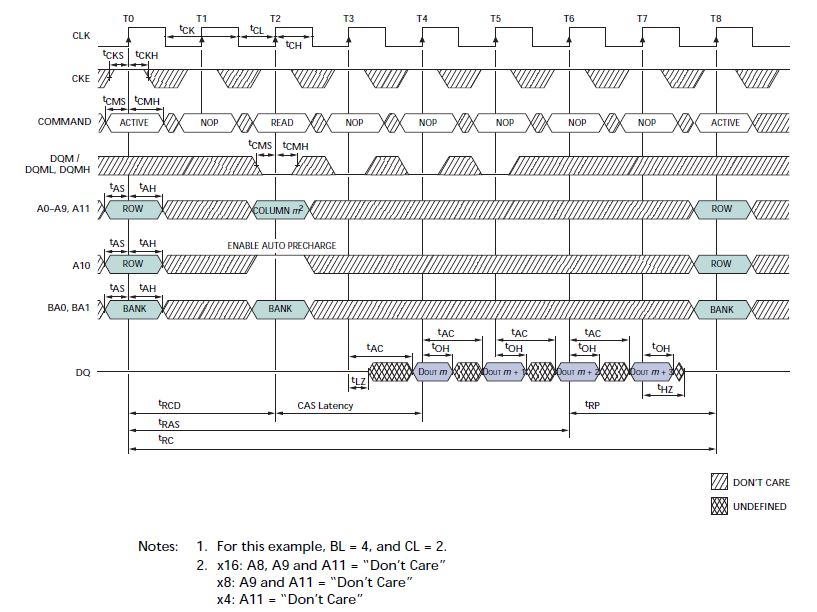
|  |  |  |
| --- | --- | --- |
| # | **STATE** | **COMMENTS** |
| 1 | IDLE | i) Wait until Refresh\_Timeout != 0 |
| 2 | AUTO-REFRESH | i) Issue AUTO REFRESH command |
| 3 | ARNOP | i) Issue NOPs and wait until Counter\_tRFC = 0 |

**Table 9:** Auto-Refresh State Comments

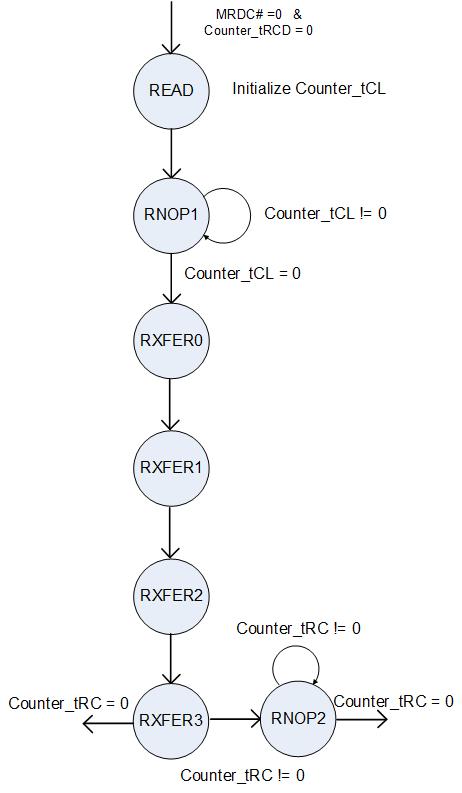
**3.4 RD STATE DIAGRAM**

The Read cycle timing diagram of figure 12 was used to implement the sub-state machine RD.

The state diagram, the state transition table, the output signals table, and the comments are shown in figure 13, table 10, table 11, and table 12.

****

**Figure 12:** READ CYCLE - Timing diagram with Auto-Pre-charge



**Figure 13:** Read State Diagram

|  |  |  |
| --- | --- | --- |
| **CURRENT STATE** | **CONDITIONS** | **NEXT STATE** |
| NOP | MRDC# = 0 & Counter\_tRCD = 0 | READ |
| READ | (next clock) | RNOP1 |
| RNOP1 | Counter\_tCL != 0 | RNOP1 |
| RNOP1 | Counter\_tCL = 0 | RXFER0 |
| RXFER0 | (next clock) | RXFER1 |
| RXFER1 | (next clock) | RXFER2 |
| RXFER2 | (next clock) | RXFER3 |
| RXFER3 | Counter\_tRC != 0 | RNOP2 |
| RXFER3 | Counter\_tRC=0 | IDLE |
| RNOP2 | Counter\_tRC!=0 | RNOP2 |
| RNO2 | Counter\_tRC=0 | IDLE |

**Table 10:** Read State Transition table

|  |  |
| --- | --- |
| **OUTPUT** | **EQUATION** |
| CKE | STATE == RNOP1 | RNOP2 | READ | RXFERn |
| RAS# | STATE == RNOP1 | RNOP2 | READ |
| CAS# | STATE == RNOP1 | RNOP2 |
| WE# | STATE == RNOP1 | RNOP2 | READ |
| A10 | STATE == READ |
| DQM | STATE == READ | RNOP1 | RXFER0 | RXFER1 |
| BA1 | STATE == READ |
| BA0 | STATE == READ |
| A[0:9] | STATE == READ |
| DQ | STATE == RXFERn |
|  | |
| n = 0,1,2,3 | |

**Table 11:** Read State Output Signals

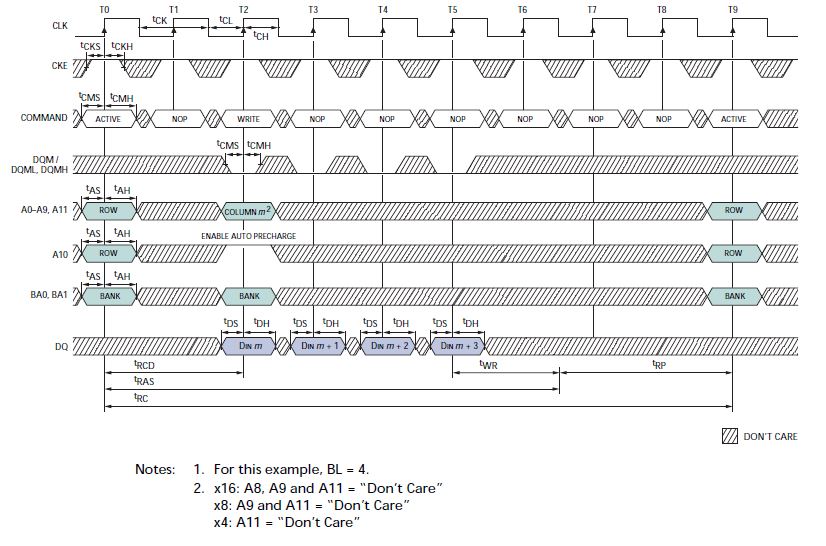
|  |  |  |
| --- | --- | --- |
| # | **STATE** | **COMMENTS** |
| 1 | READ | i) issue READ command, ADDR = Bank/col |
|  |  | ii) issue first DQM = 0 for 4 clocks keeping with BL = 4 |
|  |  | iii) Initialize Counter\_tCL |
| 2 | RNOP1 | i) Issue NOPs and wait until Counter\_tCL = 0 |
| 3 | RXFER0 | i) Bits 0-3 valid |
| 4 | RXFER1 | i) Bits 4-7 valid |
| 5 | RXFER2 | i) Bits 8-11 valid |
| 6 | RXFER3 | i) Bits 12-15 valid |
| 7 | RNOP2 | ii) Issue NOPs and wait until Counter\_tRC = 0 |
|  | | |
| \* Since we do Auto Precharge: By providing sufficient time for tRC, we ensure that the required times tRAS and tRP are satisfied. | | |
|  | | |
| \* Before sending READY to the processor: READY signal is inverted since it is an ACTIVE LOW signal for the processor. | | |

**Table 12:** Read State Comments

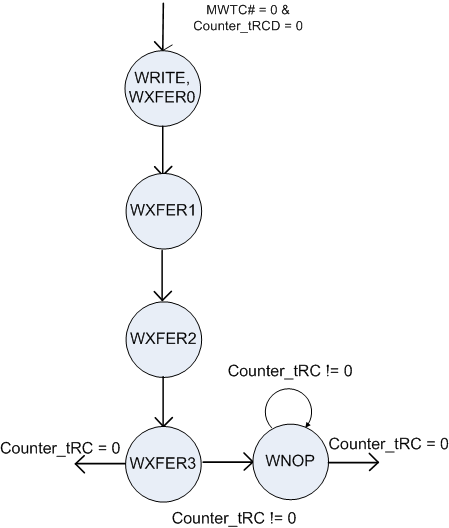
**3.5 WR STATE DIAGRAM**

The Write cycle timing diagram of figure 14 was used to implement the sub-state machine WR.

The state diagram, the state transition table, the output signals table, and the comments are shown in figure 15, table 13, table 14, and table 15.



**Figure 14:** WRITE CYCLE - Timing diagram With Auto-Pre-charge

****

**Figure 15:** Write State Diagram

|  |  |  |
| --- | --- | --- |
| **CURRENT STATE** | **CONDITIONS** | **NEXT STATE** |
| NOP | MWTC# = 0 & Counter\_tRCD = 0 | WRITE/WXFER0 |
| WRITE/WXFER0 | (next clock) | WXFER1 |
| WXFER1 | (next clock) | WXFER2 |
| WXFER2 | (next clock) | WXFER3 |
| WXFER3 | Counter\_tRC != 0 | WNOP |
| WXFER3 | Counter\_tRC=0 | IDLE |
| WNOP | Counter\_tRC!=0 | WNOP |
| WNOP | Counter\_tRC=0 | IDLE |

**Table 13:** Write State Transition table

|  |  |
| --- | --- |
| **OUTPUT** | **EQUATION** |
| CKE | STATE == WNOP | WXFERn | WRITE |
| RAS# | STATE == WNOP | WRITE |
| CAS# | STATE == WNOP |
| WE# | STATE == WNOP |
| A10 | STATE == WRITE |
| DQM | STATE == WRITE | WXFERn |
| BA1 | STATE == WRITE |
| BA0 | STATE == WRITE |
| A[0:9] | STATE == WRITE |
| DQ | STATE == WRITE | WXFERn |
|  | |
| n =0,1,2,3 | |

**Table 14:** Write State Output Signals

|  |  |  |
| --- | --- | --- |
| # | **STATE** | **COMMENTS** |
| 1 | WRITE, WXFER0 | i) issue WRITE command, ADDR = Bank/col, bits 0-3 valid |
| 2 | WXFER1 | i) Bits 4-7 valid |
| 3 | WXFER2 | i) Bits 8-11 valid |
| 4 | WXFER3 | i) Bits 12-15 valid |
| 5 | WNOP | ii) Issue NOPs and wait until Counter\_tRC = 0 |
|  | | |
| \* Since we do Auto Precharge: By providing sufficient time for tRC, we ensure that the required times tRAS, tWR, and tRP are satisfied. | | |
|  | | |
| \* Before sending READY to the processor: READY signal is inverted since it is an ACTIVE LOW signal for the processor. | | |

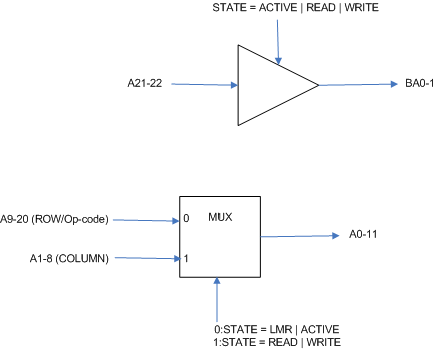
**Table 15:** Write State Comments

**5. SIGNAL DECODER**

**5.1 ROW, BANK AND COLUMN SIGNAL GENERATION**

A tri-state buffer is used for bank signal generation. The bank signals are available in states ACTIVE, READ, and WRITE.

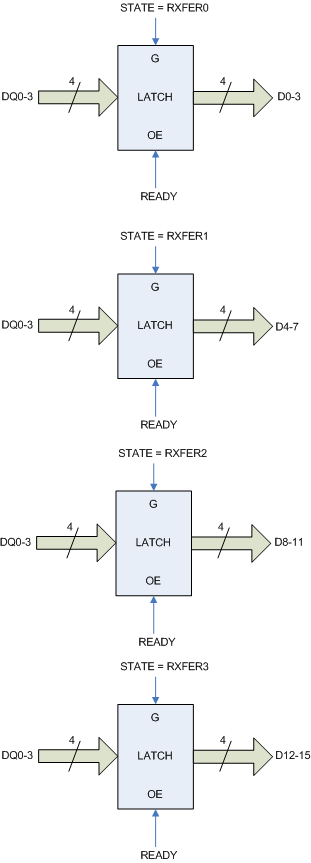
A 2:1 MUX is used to choose between the ROW, in states LMR or ACTIVE, and COLUMN, in states READ or WRITE. The address lines A1-8 from the processor are considered as MSBs and the two remaining address lines to fulfill the SDRAM A0-11 are hardwired to zeros.

****

**Figure 16:** Row, Bank and Column signal generation

**5.2 READING DATA FROM SDRAM**

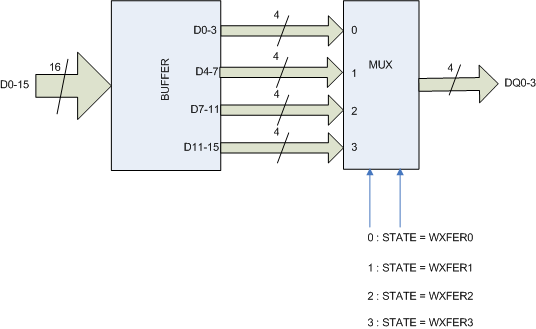
Figure 17 explains the data read sequence: The nibbles DQ0-3, DQ4-7, DQ8-11, DQ12-15 from the SDRAM memory are latched in during states RXFER0, RXFER1, RXFER2, RXFER3. Once the 4 nibbles have been written, READY is asserted which then presents the 16-bit data to the processor.

****

**Figure 17:**  Read Data circuitry

**5.3 WRITING DATA TO SDRAM**

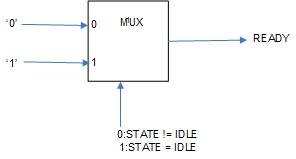
Figure 18 explains the data write sequence: The 16-bit data from the processor is available and is stored in the buffer shown. The MUX then outputs each of the nibbles D0-3, D4-7, D8-11, D12-15 individually to be written to the SDRAM during states WXFER0, WXFER1, WXFER2, WXFER3 respectively in the Write state machine.”

****

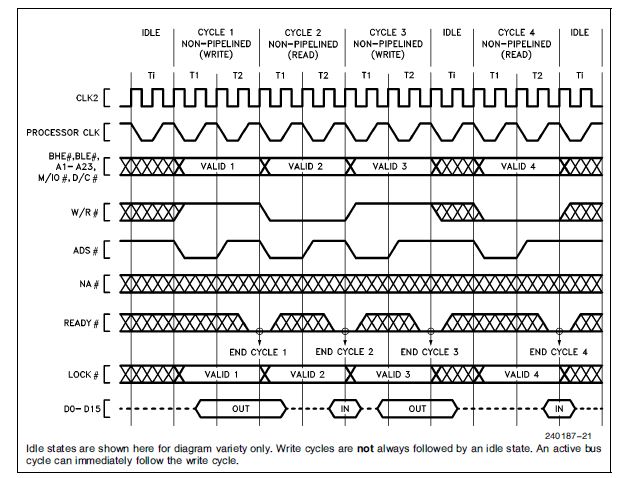
**Figure 18:** Write Data Circuitry

**5.4 READY SIGNAL GENERATION**

A 2:1 MUX used for the generation of the READY signal. READY is asserted when in IDLE state. Figure 20 shows the READY signal generation required by the processor. For our design, the SDRAM controller asserts the READY signal when data becomes valid at the end of the READ or WRITE cycles.

****

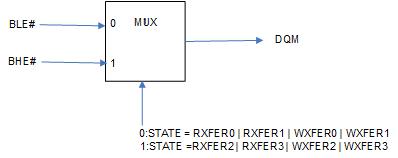
**Figure 19:** READY Signal generation

****

**Figure 20:** READY Signal timing for 80386SX processor

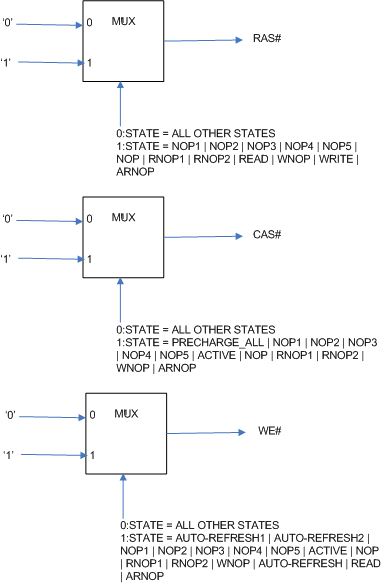
**5.5 DQM SIGNAL GENERATION**

A 2:1 MUX used for the generation of the DQM signal for which BLE# and BHE# are used as inputs to the MUX. BLE# is passed as the DQM signal in states RXFER0 | RXFER1 | WXFER0 | WXFER1. BHE# is passed as the DQM signal in states RXFER2 | RXFER3 | WXFER2 | WXFER3.

****

**Figure 21:** DQM Signal Generation

**5.6 RAS ,CAS AND WE GENERATION**

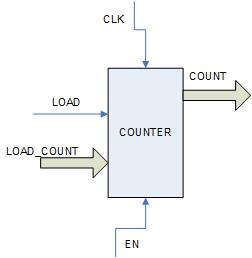
****

**Figure 22:** RAS#, CAS#, WE# Signal Generation

**5.7 COUNTER**

We have 2 dedicated counters for COUNTER\_tRC and the AUTO REFRESH Counter for which the times of tRC and the refresh times are loaded.

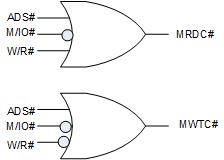
The third counter is available to load the values for Counter\_tRCD, Counter\_tRFC, and Counter\_tCL when needed. We have not used dedicated counters for these times since they are not used simultaneously in any of the state machines.

****

**Figure 23:** Programmable Counter

**5.8 MRDC# AND MWTC# Signal generation**

We used the processor ADS#, M/IO#, and W/R# signals to generate the MRDC# and MWTC# input signals to the SDRAM Memory Controller.



**Figure 24:** Read and Write Control Signals