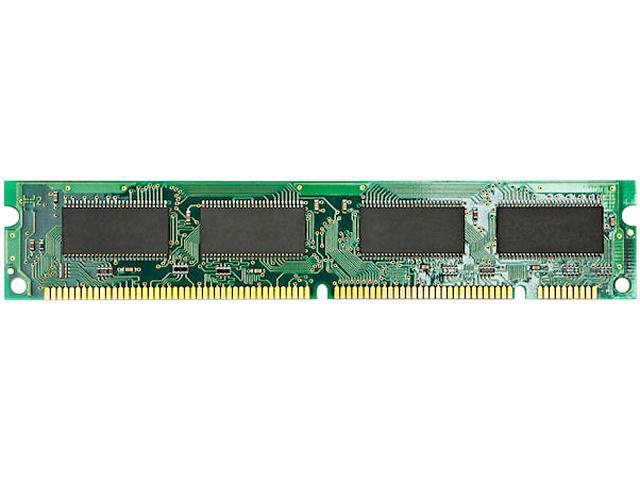
EE 5313

Project

**Synchronous**



**Design**

**CONTROLLER**

By

* Anirudh
* Arun
* Krishna

**Table of Contents.**

1. Introduction.
2. SDRAM Controller Interface.
3. State machine Diagrams & State Transition Tables.
4. Address Generation and Data Interface.
5. Control Signals, Delays and Counters Generation.
6. Extra- Credits.

**Introduction.**

The DRAM’s asynchronous operation caused many design challenges when interfacing it to a synchronous processor. SDRAM (Synchronous DRAM) was designed to synchronize the DRAM operation to the rest of the computer system and to eliminate defining all the different modes of memory operations based on the sequence of CE# (Chip Enable active low), RAS#, CAS# and WE# edge transitions.

SDRAM added a clock signal and the concept of memory commands. The type of memory command is determined by the state of CE#, RAS#, CAS# and WE# signals at the rising edge of the SDRAM clock.

The Data is divided into several banks allowing to access chip to access several memories at a time, interleaved among the separate banks. This allows higher data rate than a synchronous DRAM.

80386DX is an asynchronous microprocessor with a 32bit address lines and 32 bit data lines (4 Banks). The 4GiB memory space is organized as 1G\*32-bits.

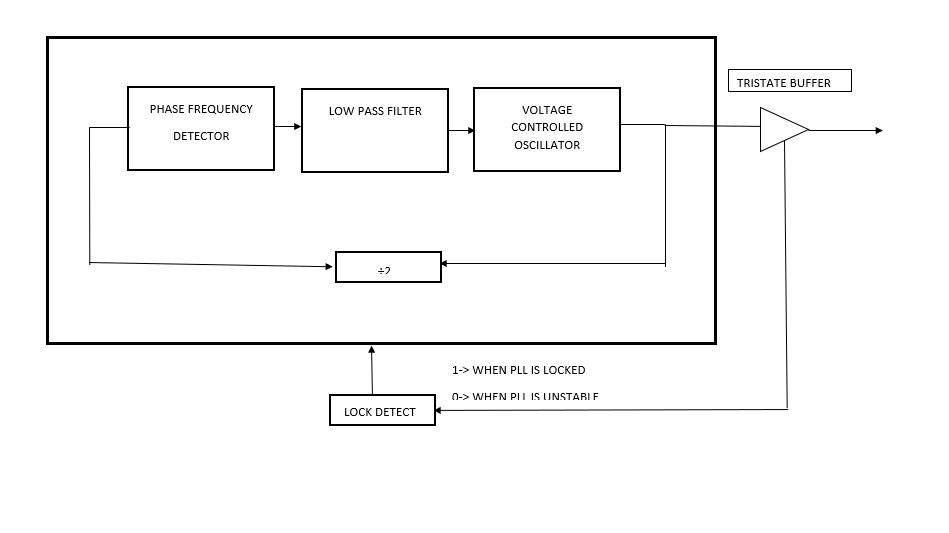
The Designed SDRAM CONTROLLER supports the connection between a 80386DX microprocessor and a synchronous SDRAM memory (MT48LC4M16A2). The controller enables the conversion of asynchronous 80386DX commands into synchronous SDRAM memory command and control words. The design supports a burst length of 4 for the SDRAM operation.

**Advantages:**

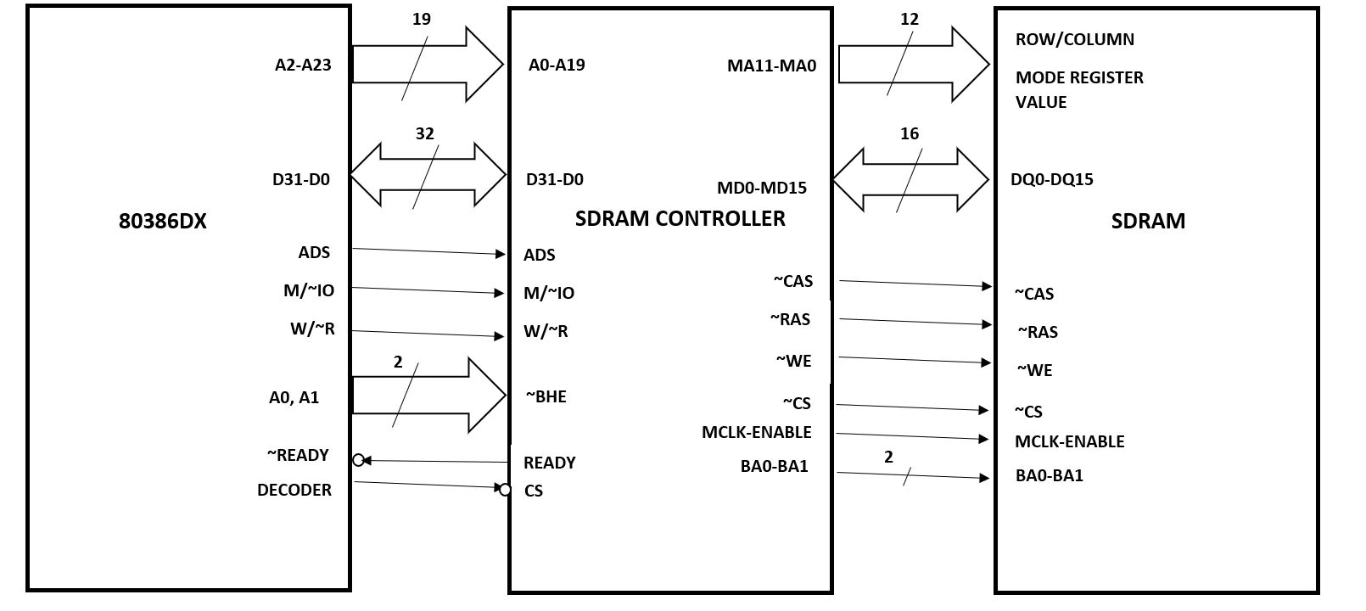
* The Interface is simplified from a SDRAM command interface to a standard system read/write interface.
* The State machine is unique for SDRAM initialization, Command and Clock stabilization.
* The configured mode registers and the SDRAM timing specification optimizes automatically the R/W access times.
* Easy to design.
* The phase locked loop helps in taking a slower system clock. The interfaced clock is not the same as the SDRAM clock.

**Phase Locked Loop:**

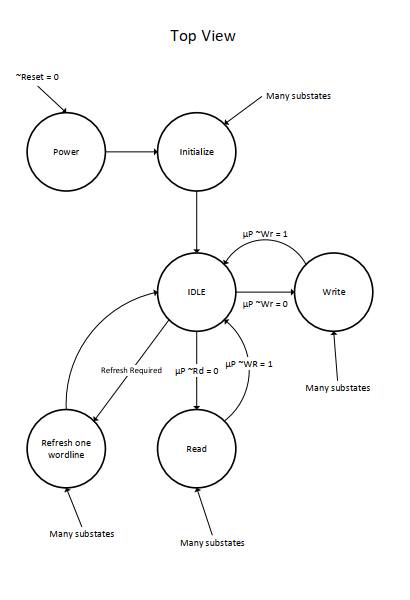
The PLL provides a clock output which is the base clock for all state machines of the SDRAM controller. At the start, the PLL doesn’t provide a stable output stable clock. It takes a certain amount of time to lock the clock with a certain frequency which is thereafter stable. The phase detector also called as EX-OR gate is used to lock and make the output stable. Until this happens no output is observed, and all the states of the state machines are off. There is an LPF, voltage-controlled oscillator and a divider (by 2) that helps in attaining the stable clock.



**SDRAM Controller Interface:**

****

**State Diagram Top View:**

****

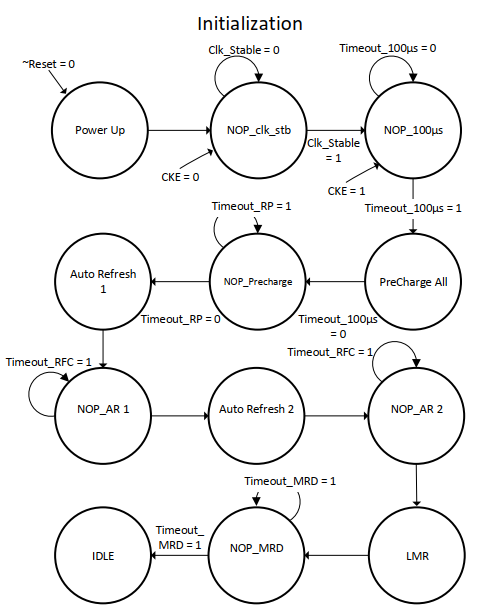
Up ~ RD = 1

**INITIALIZATION:**

SDRAMs must be powered up and initialized in a predefined manner. Operational procedures other than those specified may result in undefined operation. Once power is applied to VDD and VDDQ (simultaneously) and the clock is stable (stable clock is defined as a signal cycling within timing constraints specified for the clock pin), the SDRAM requires a 100µs delay prior to issuing any command other than a COMMAND INHIBIT or a NOP. Starting at some point during this 100µs period and continuing at least through the end of this period, COMMAND INHIBIT or NOP commands should be applied. Once the 100µs delay has been satisfied with at least one COMMAND INHIBIT or NOP command having been applied, a PRECHARGE command should be applied. All banks must be Precharged, thereby placing the device in the all banks idle state. Once in the idle state, two AUTO REFRESH cycles must be performed. After the AUTO REFRESH cycles are complete, the SDRAM is ready for mode register programming. Because the mode register will power up in an unknown state, it should be loaded prior to applying any operational command.

**State Machine Diagram:**

**Initialization:**



**State Transition Table:**

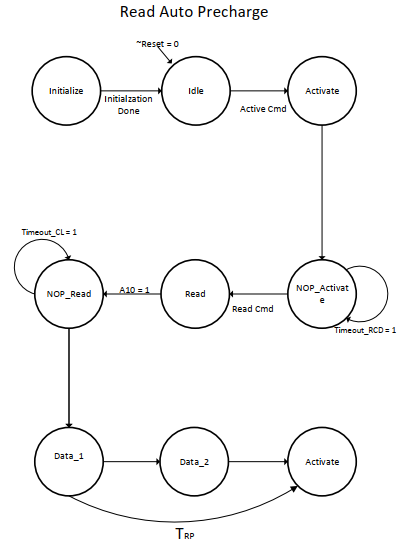
|  |  |  |
| --- | --- | --- |
| **STATE** | **CONDITION** | **NEXT STATE** |
| X | ~RESET = 0  ~MEMCS = 1 | Power UP |
| Power Up | CKE = 0  ~MEMCS = 1  CLK Stable = 0 | NOP\_clk\_stb |
| NOP\_clk\_stb | CKE = 1  CLK Stable = 1 | NOP\_100µs |
| NOP\_100µs | Timeout = 0 | Precharge All |
| Precharge All | Load Timeout\_RP | NOP\_Precharge |
| NOP\_Precharge | Timeout\_RP = 0 | Auto Refresh |
| Auto Refresh 1 | Load Timeout\_RFC | NOP\_AR 1 |
| NOP\_AR 1 | Timeout\_RFC = 0 | Auto Refresh 2 |
| Auto Refresh 2 | Load Timeout\_RFC | NOP\_AR 2 |
| NOP\_AR 2 | Timeout\_RFC = 0 | LMR |
| LMR | Load Timeout\_MRD | NOP\_MRD |
| NOP\_MRD | Timeout\_MRD = 0 | IDLE |

**READ:**

READ bursts are initiated with a READ command. The starting column and bank addresses are provided with the READ command, and auto Precharge is either enabled or disabled for that burst access. If auto Precharge is enabled, the row being accessed is Precharged at the completion of the burst. During READ bursts, the valid data-out element from the starting column address will be available following the CAS latency after the READ command. Each subsequent data-out element will be valid by the next positive clock edge.

Upon completion of a burst, assuming no other commands have been initiated, the DQs will go High-Z. A full-page burst will continue until terminated.

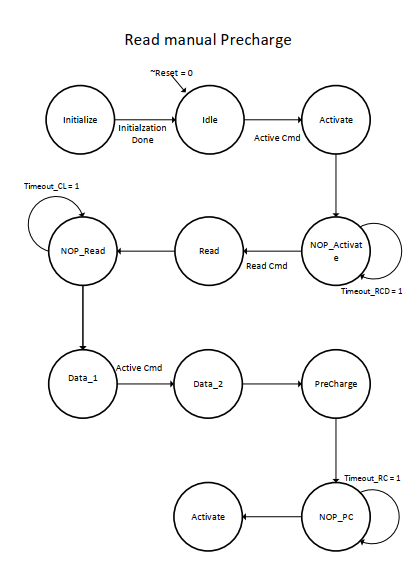
**Read with Auto Precharge:**

****

**State Transition Table Read:**

|  |  |  |
| --- | --- | --- |
| **STATE** | **CONDITION** | **NEXT STATE** |
| Initialize | X | Idle |
| Idle | ~Reset = 0 | Idle |
| Idle | Active Command | Activate |
| Activate | Load Timeout\_RCD  Load Timeout\_RC | NOP\_Activate |
| NOP\_Activate | Timeout\_RCD = 0 | Read |
| NOP\_Activate | A10 = 1 | Enable Auto Precharge |
| Read | Load Timeout\_TCL | NOP\_Read |
| NOP\_Read | Timeout\_TCl = 0 | Data 1 |
| Data 1 | X | Data 2 |
| Data 1 | Timeout\_RP | Activate |
| Data 2 | Timeout\_RC = 0 | Activate |

**Read with Manual Precharge:**



Timeout\_RP = 1

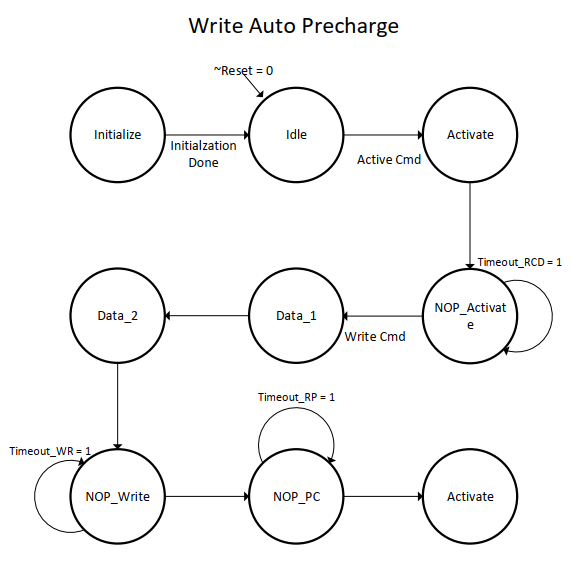
**State Transition Table Read:**

|  |  |  |
| --- | --- | --- |
| **STATE** | **CONDITION** | **NEXT STATE** |
| Initialize | X | Idle |
| Idle | ~Reset = 0 | Idle |
| Idle | Active Command | Activate |
| Active | Load Timeout\_RCD  Load Timeout\_RC | NOP\_Activate |
| NOP\_Activate | Timeout\_RCD = 0 | Read |
| NOP\_Activate | A10 = 0 | Disable Auto Precharge |
| Read | Load Timeout\_TCL | NOP\_Read |
| NOP\_Read | Timeout\_TCL = 0 | Data 1 |
| Data 1 | X | Data 2 |
| Data 2 | Timeout\_RAS = 0 | Precharge |
| Precharge | Load Timeout\_RP | NOP\_PC |
| NOP\_PC | Timeout\_RP = 0 | Activate |

**WRITE:**

WRITE bursts are initiated with a WRITE command. The starting column and bank addresses are provided with the WRITE command, and auto Precharge is either enabled or disabled for that access. If auto Precharge is enabled, the row being accessed is Precharged at the completion of the burst. During WRITE bursts, the first valid data-in element will be registered coincident with the WRITE command. Subsequent data elements will be registered on each successive positive clock edge. Upon completion of a fixed-length burst, assuming no other commands have been initiated, the DQs will remain High-Z and any additional input data will be ignored. A full-page burst will continue until terminated.

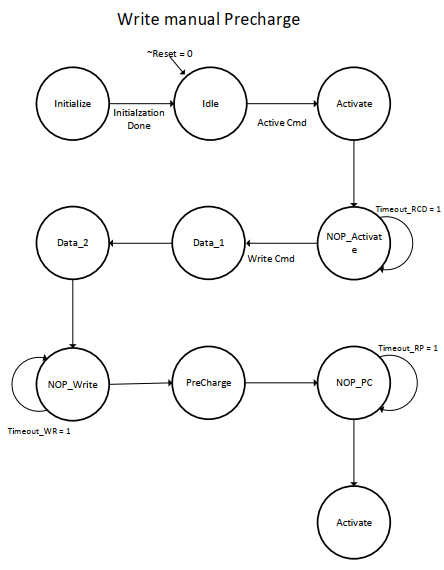
**Write with Auto Precharge:**

****

**State Transition Table Write:**

|  |  |  |
| --- | --- | --- |
| **STATE** | **CONDITION** | **NEXT STATE** |
| Initialize | X | Idle |
| Idle | ~Reset = 0 | Idle |
| Idle | Active Command | Activate |
| Active | Load Timeout\_RCD  Load Timeout\_RC | NOP\_Activate |
| NOP\_Activate | Timeout\_RCD = 0 | Write |
| NOP\_Activate | A10 = 1 | Enable Auto Precharge |
| NOP\_Activate | Column Address, Banks | Data 1 |
| Data 1 | X | Data 2 |
| Data 2 | Load Timeout\_WR | NOP\_Write |
| NOP\_Write | Timeout\_WR=0 load Timeout\_RP | NOP\_PC |
| NOP\_PC | Timeout\_RP = 0 | Activate |

**Write with Manual Precharge:**

****

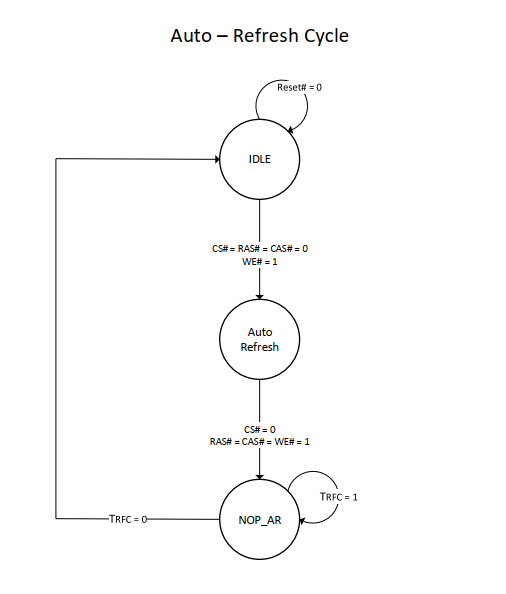
**State Transition Table Write:**

|  |  |  |
| --- | --- | --- |
| **STATE** | **CONDITION** | **NEXT STATE** |
| Initialize | X | Idle |
| Idle | ~Reset = 0 | Idle |
| Idle | Active Command | Activate |
| Activate | Load Timeout\_RCD  Load Timeout\_RC | NOP\_Activate |
| NOP\_Activate | Timeout\_RCD = 0 | Write |
| NOP\_Activate | A10 = 0 | Disable Auto Precharge |
| NOP\_Activate | Column Address, Banks | Data 1 |
| Data 1 | X | Data 2 |
| Data 2 | Timeout\_WR | NOP\_WR |
| NOP\_WR | Timeout\_WR=0 | Precharge |
| Precharge | load Timeout\_RP | NOP\_PC |
| NOP\_PC | Timeout\_RP = 0 | Activate |

**AUTO REFRESH:**

It is used during normal operation of the SDRAM and is analogous to CAS#-BEFORE-RAS# (CBR) REFRESH in conventional DRAMs. This command is nonpersistent, so it must be issued each time a refresh is required. All active banks must be PRECHARGED prior to issuing an AUTO REFRESH command. The AUTO REFRESH command should not be issued until the minimum tRP has been met after the PRECHARGE command as shown in the operation section. The addressing is generated by the internal refresh controller. This makes the address bits “Don’t Care” during an AUTO REFRESH command. The 64Mb SDRAM requires 4,096 AUTO REFRESH cycles every 64ms (t REF), regardless of width option. Providing a distributed AUTO REFRESH command every 15.625µs will meet the refresh requirement and ensure that each row is refreshed. Alternatively, 4,096 AUTO REFRESH commands can be issued in a burst at the minimum cycle rate (t RC), once every 64ms.

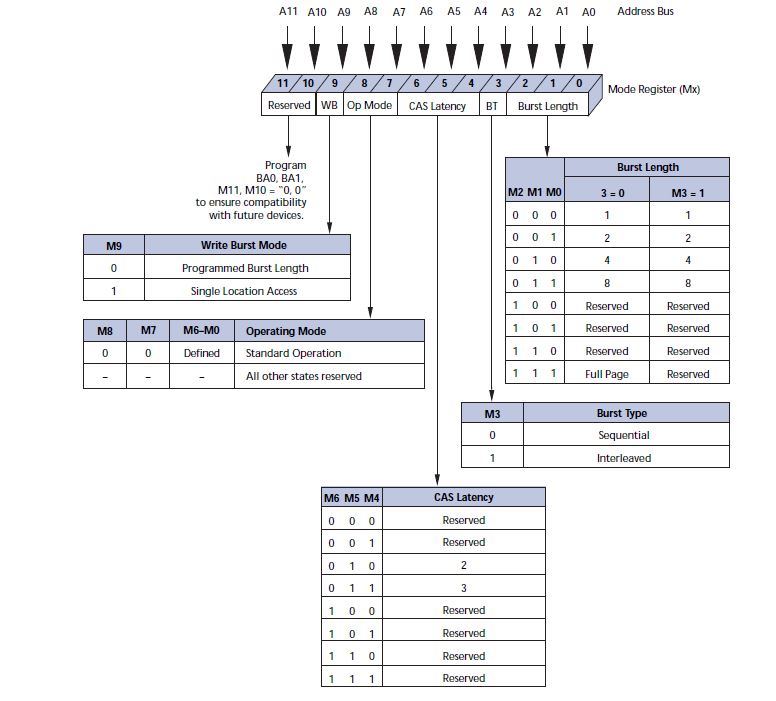
**Refresh:**

****

**State Transition Table:**

|  |  |  |
| --- | --- | --- |
| **STATE** | **CONDITION** | **NEXT STATE** |
| Initialize | X | Idle |
| Idle | ~Reset = 0 | Idle |
| Idle | Command Auto\_Refresh | Auto\_Refresh |
| Auto\_Refresh | Load Timeout\_RFC | NOP\_AR |
| NOP\_AR | Timeout\_RFC = 0 | Activate |

**Load Mode Register:**



**LMR Initialization:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RESERVED | WRITE BURST MODE | OPERATING MODE | CAS LATENCY | BURST TYPE | BURST LENGTH |
| X X | 0 | 0 0 | 0 1 0 | 0 | 0 0 1 |

**Address & Data Generation:**

A10

A0, A1

2

|  |
| --- |
| LATCH  OE |

Precharge

Cmd

8

DQM/DQML,DQMH

1

MUX

A2-9

COL

8

RAS#

CAS#

.

SDRAM

12

0



A10-21

12

ROW

|  |
| --- |
| LATCH  OE |

12

CAS#

BA0-3

CS#

A22, A23 [BA0-3]

8

2

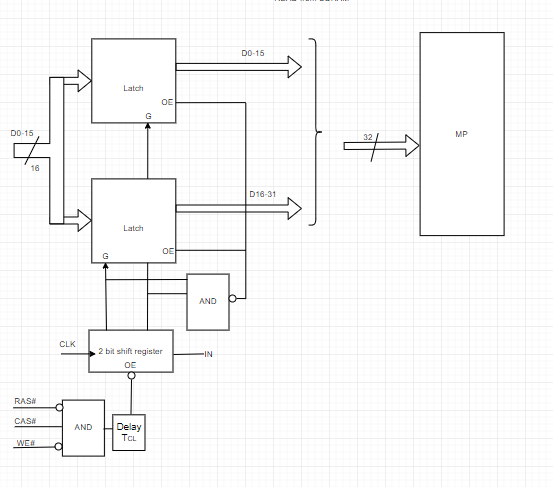
DECODER\_SDRAM

RAS#

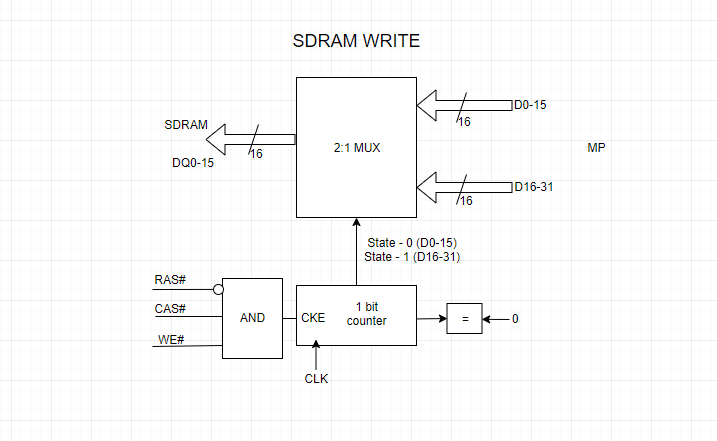
.

A24-31

**Read Signal, SDRAM:**

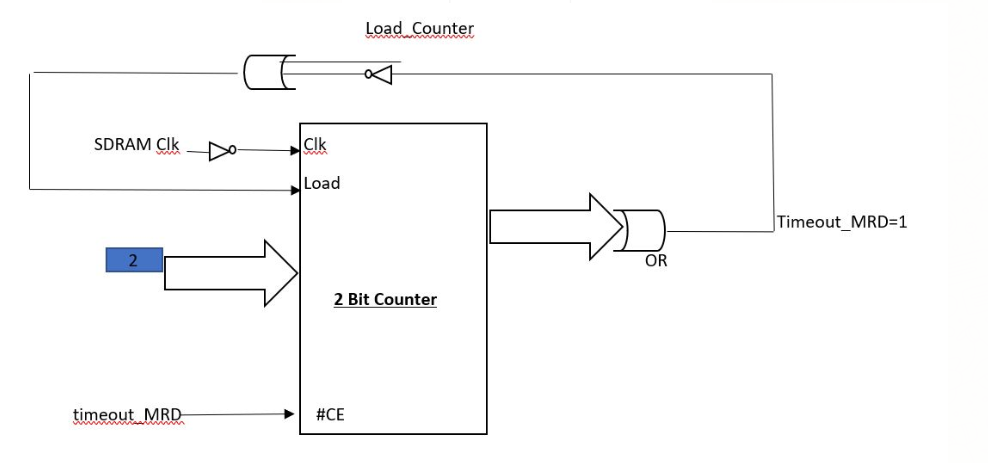


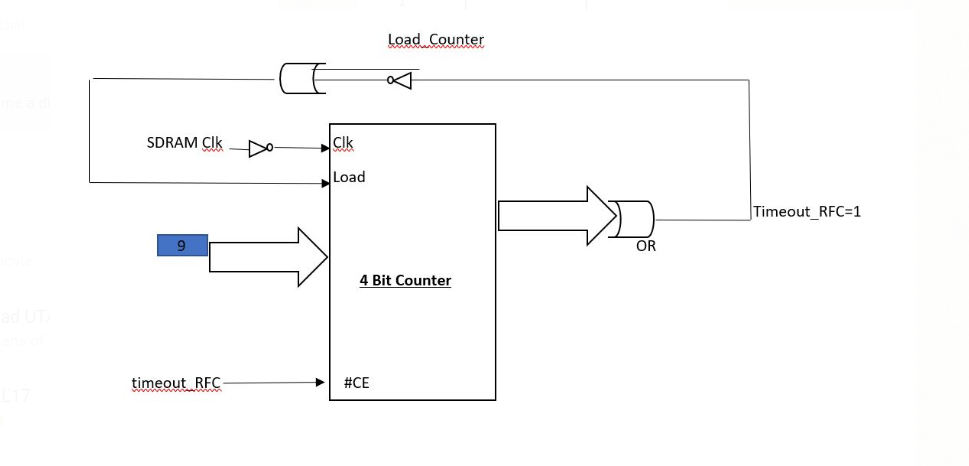
**Write Signal, SDRAM:**

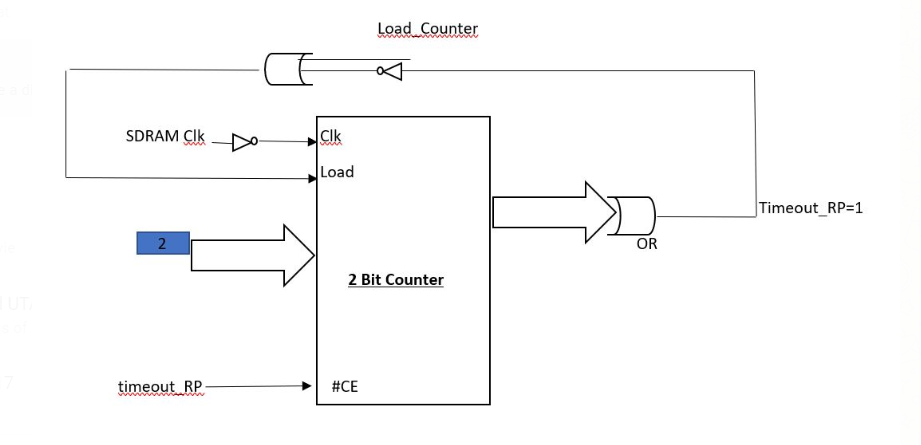
****

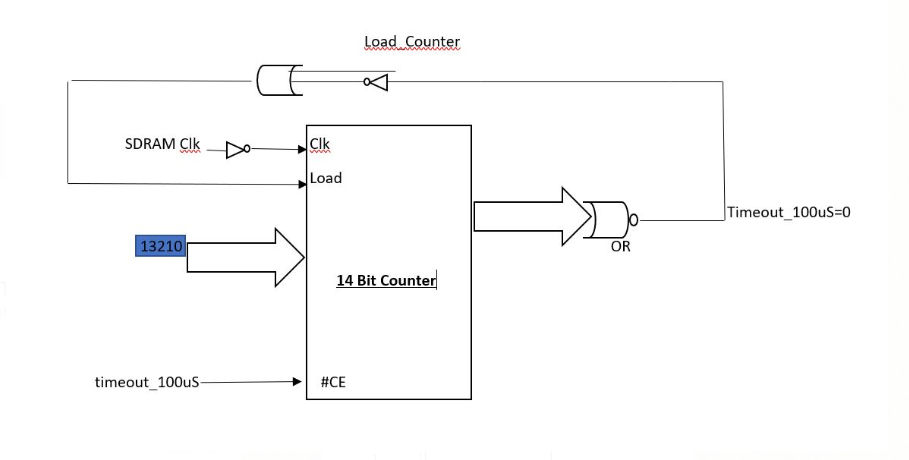
1

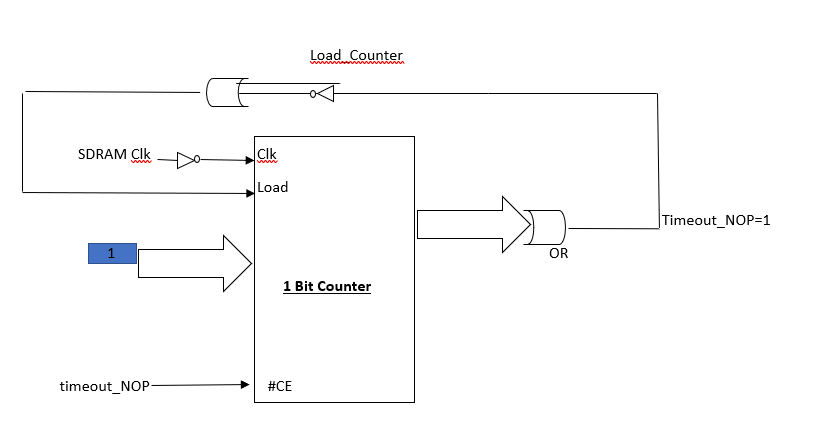
**Delay Counters:**

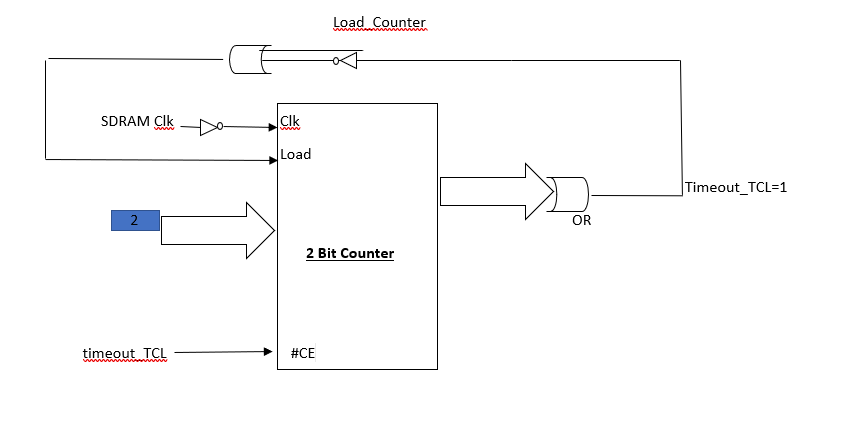
****

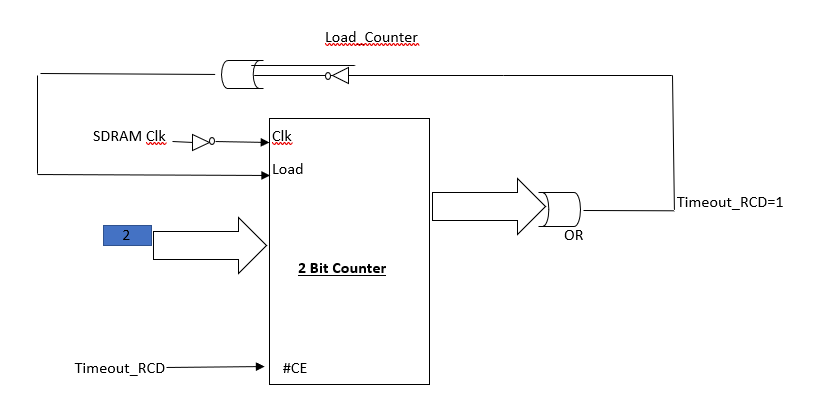
****

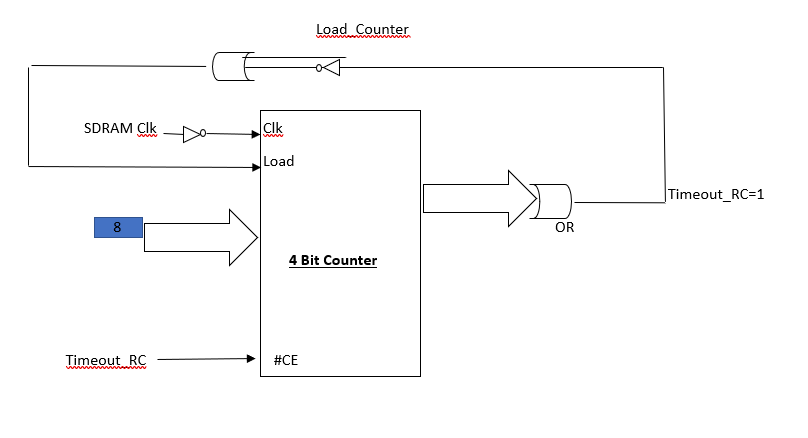
****

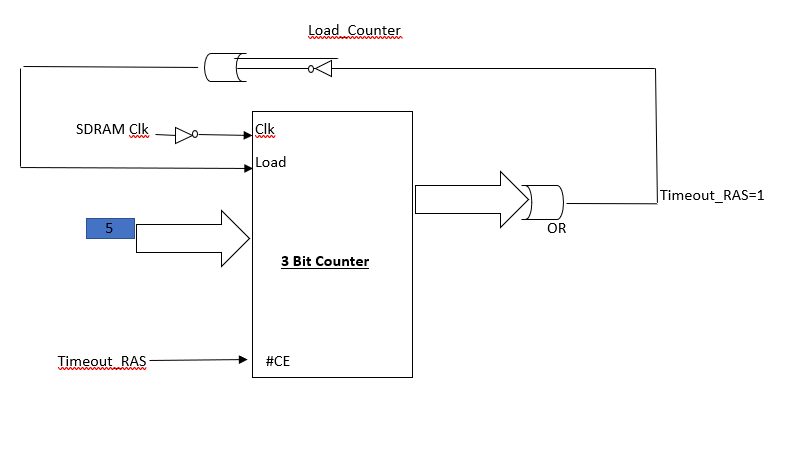
****

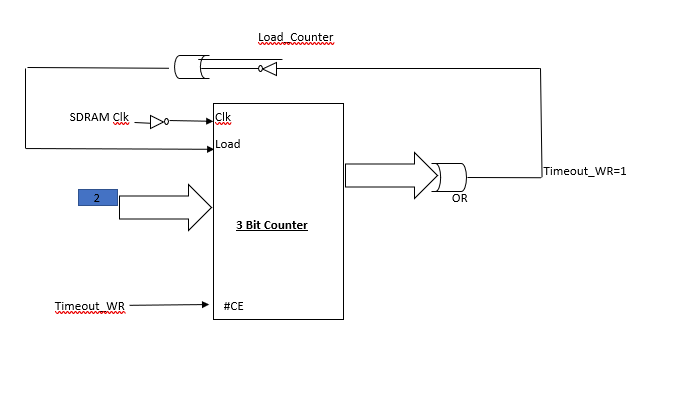
****

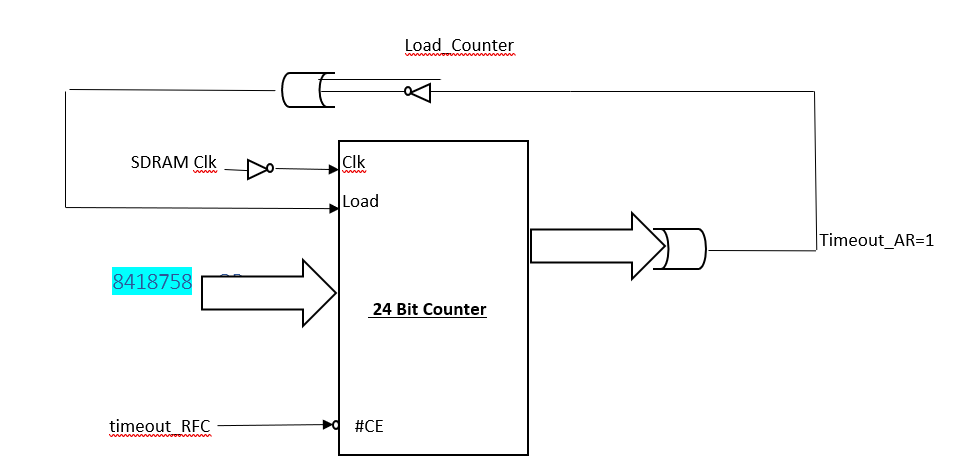
****

****

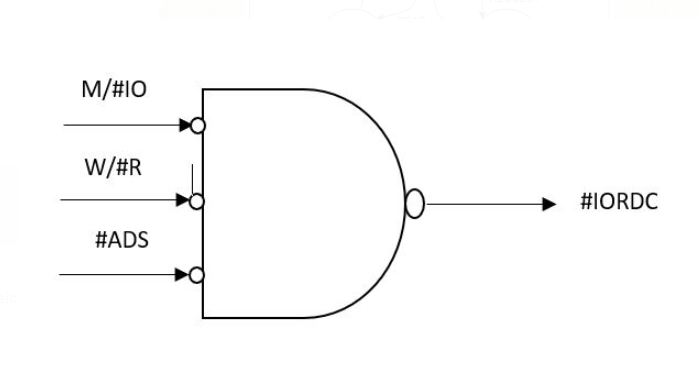
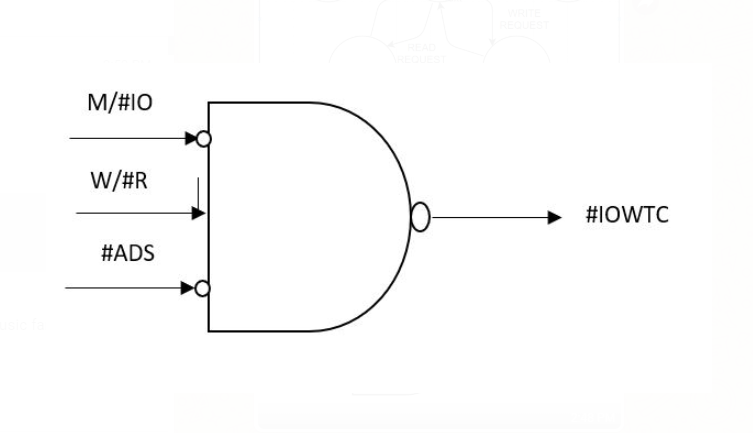
****

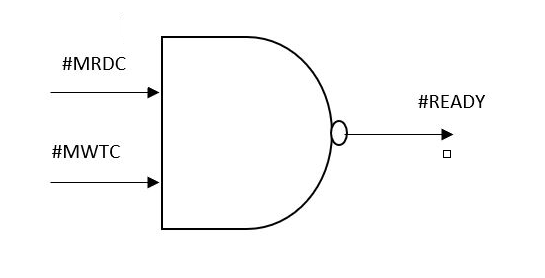
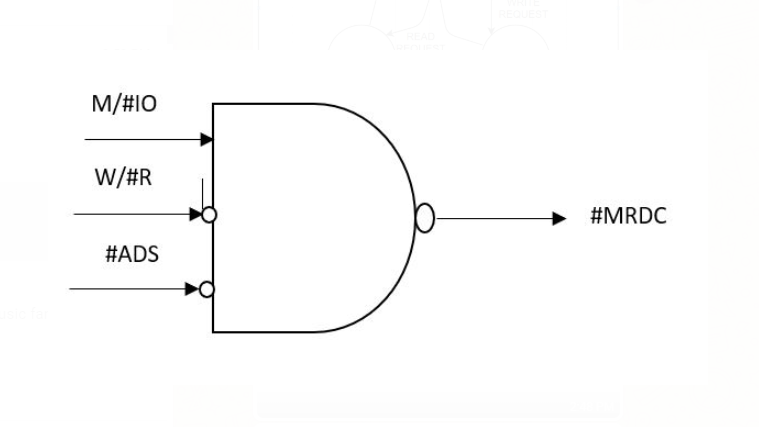
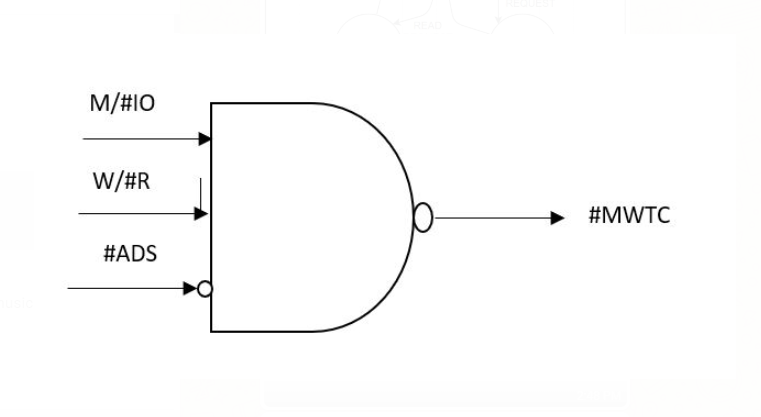
****

****

****

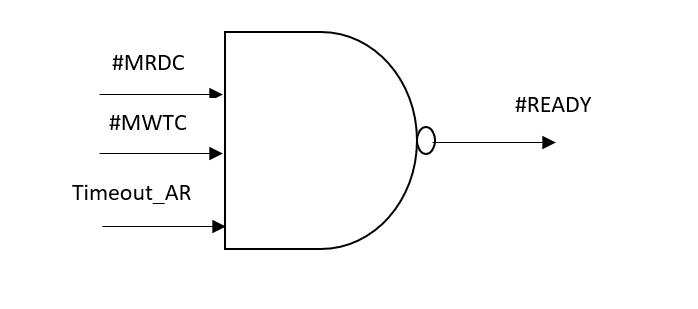
**Control Signal Generation:**

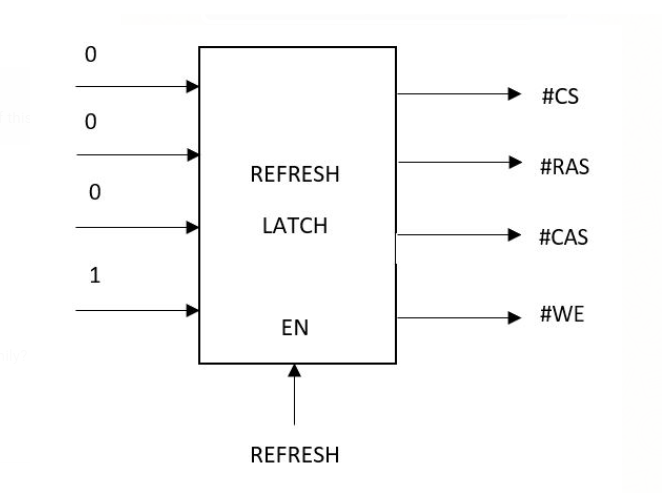
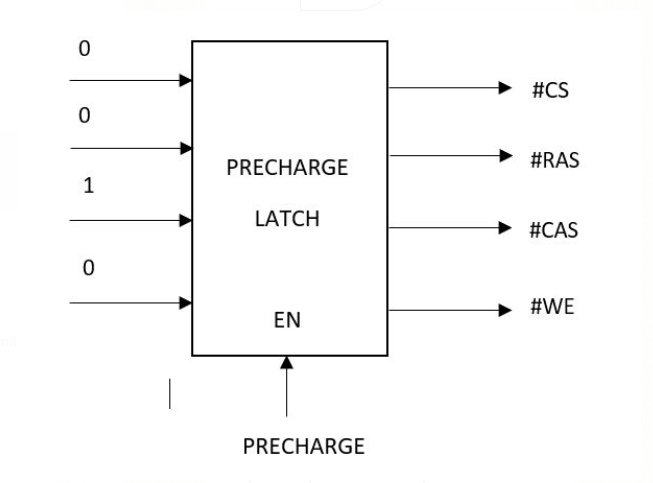
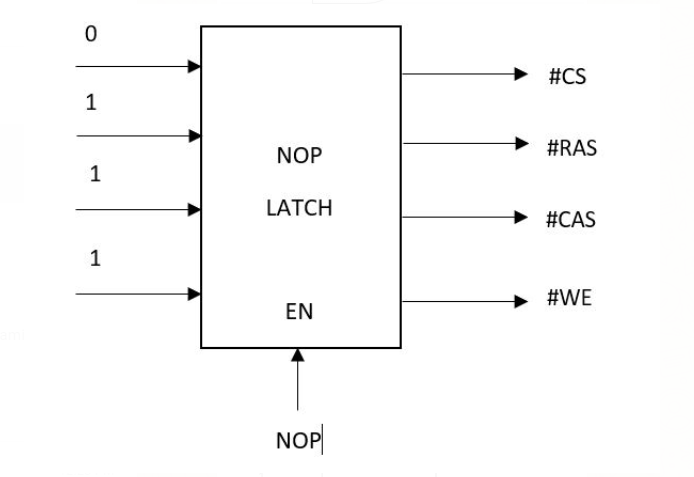
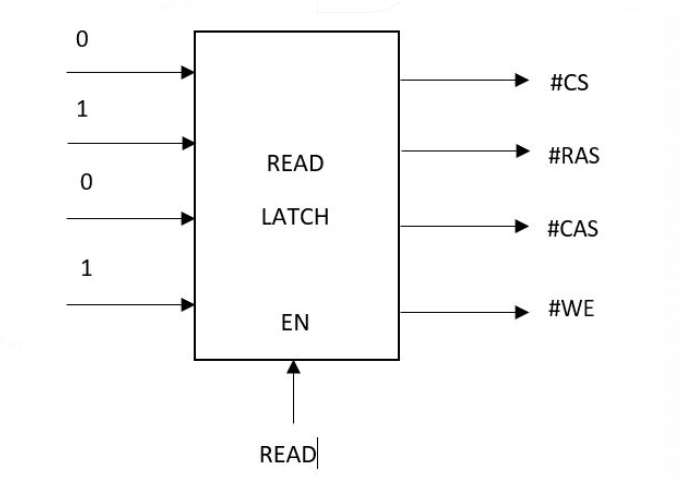
 

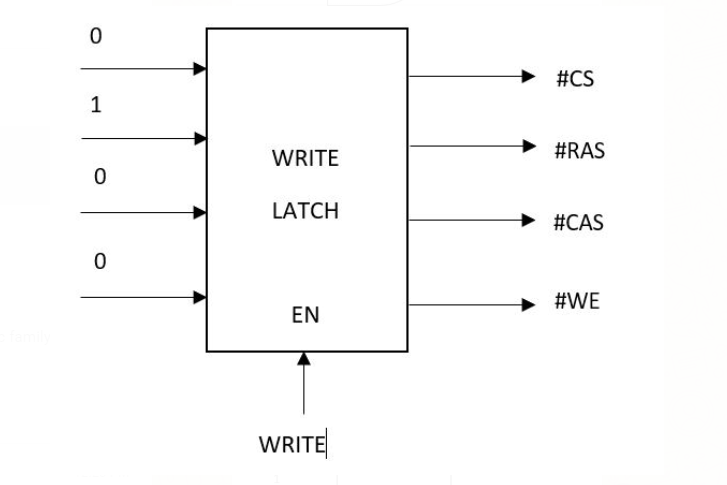


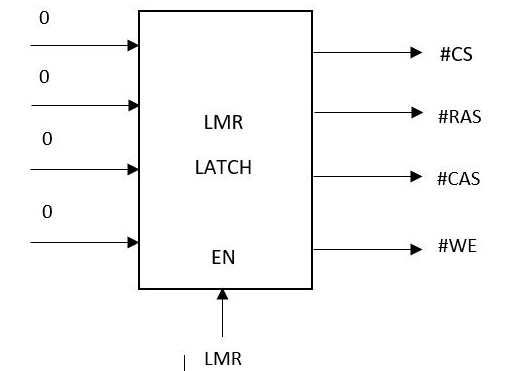
**READY SIGNAL:**

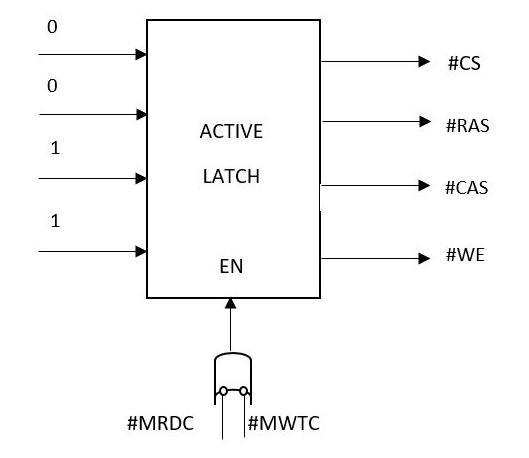
Although, MRDC# and MWTC# are signals that aren’t needed we use them to generate ready signal. For special cases like Ready signal when refresh command is issued varies accordingly as shown below. Where Timeout\_AR is the time remaining after each refresh signal is issued such that a read/ write operation can be performed.  
The Timeout\_AR is an output from the counter which is shown above, and it is termed as Timeout after refresh.



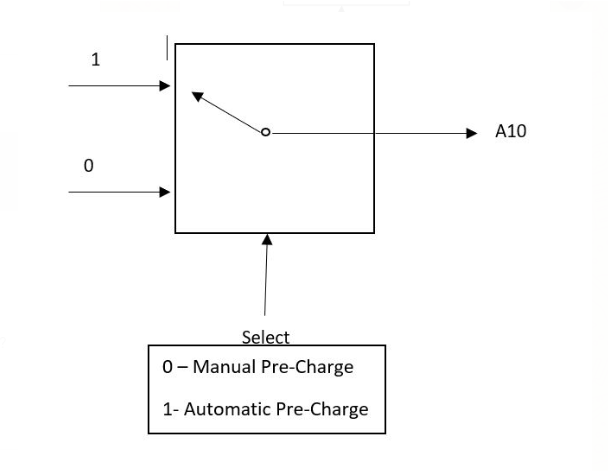
  







**Generation of A10 signal:**



**EXTRA CREDITS:**

**PROGRAMMABILITY:**

The considered programmability gives access to variable timing that can be accessed from memory (Not SDRAM). This helps in altering the timing, that is, the timing is stored in a memory address of variable bit width. The address location is predefined, and the value of the timing is stored there. This way variable timing is achieved.

|  |  |  |
| --- | --- | --- |
| **Signals** | **Bit width** | **Address(H)** |
| Tmrd | 16 | 00000 |
| Trfc | 16 | 00004 |
| Trp | 16 | 00008 |
| Trcd | 16 | 0000C |
| Tcl | 16 | 00010 |
| Tras | 16 | 00014 |
| Trc | 16 | 00018 |
| Twr | 16 | 0001C |
| Address | 32 | 0001F |
| Address in Data | 1 | 00002 |
| ~R/W | 1 | 00003 |
| ~ADS | 1 | 00006 |
| M/~IO | 1 | 00007 |
| ~CS | 1 | 0000A |
| Clock | 1 | 0000B |
| ~Reset | 1 | 0000E |
| Ready | 1 | 0000F |
| Clock Stable | 1 | 00012 |

The 32bit address sends the address from the system to the SDRAM controller when needed for an operation.

The 32 bit Data is indicated to the controller when there is a need for data transfer by the signal address in data.

ADS, ~R/W, M/~IO, CS, Clock, ~Reset, Ready, Clock Stable are a 1bit width data which tell us about the address strobe signal, read and write operation, memory input output, chip select, clock (66MHZ), reset, ready & clock stable (for PLL).

Multiplexed address is sent to the SDRAM by a 12bit width bus.

Multiplexed data is sent to the SDRAM with a 16bit bus width.

~CAS, ~RAS, ~WE indicate whether there is a transfer of row or column by a 1bit width bus.

M-Clock, M-Clock-Enable, ~M-CS are of 1b width for the SDRAM.

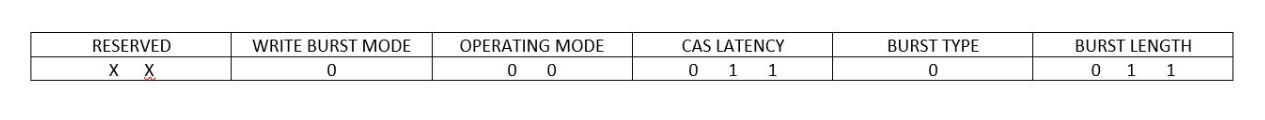
The working of the SDRAM is dependent on the control signals which can be generated using the same address of the microprocessor, we can consider A22,23 address lines and implement it.

Read Operation: 0 0

Write operation: 0 1

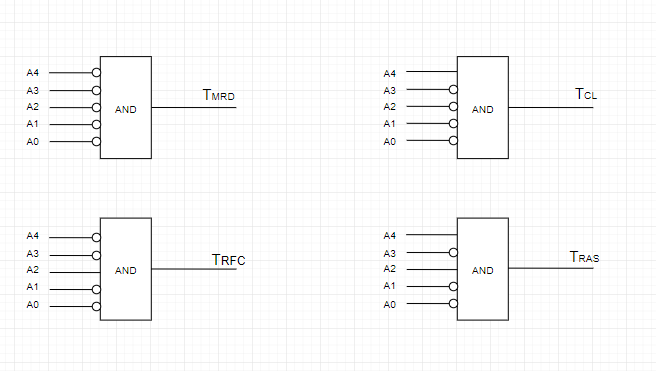
Auto Refresh Operation: 1 0

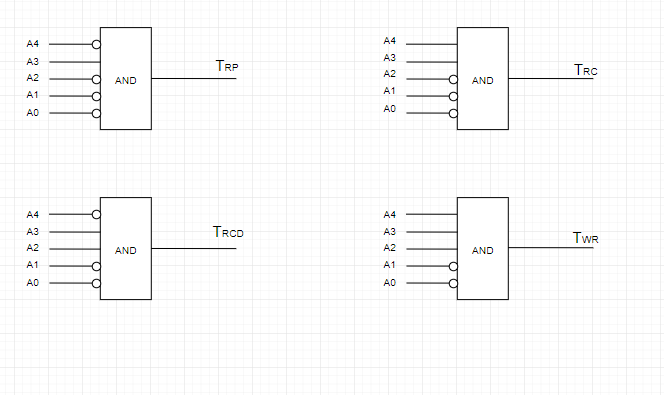
For varying the burst length and CAS latency we should consider the Load Mode Register value.  
Another example for load mode register is given below for a burst length 4 and CAS latency 2.

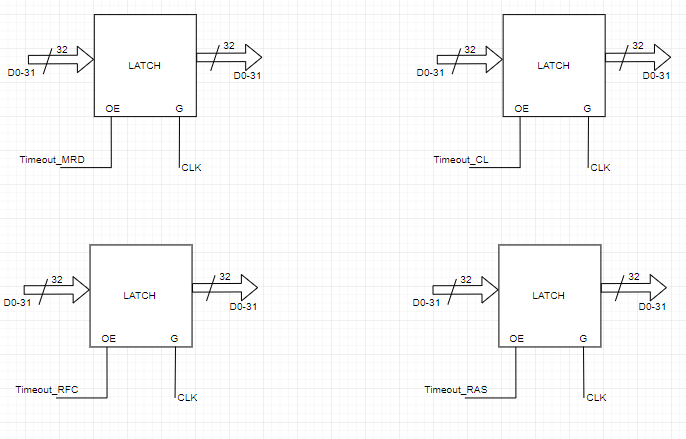
  
The timings are generated as shown above in the control signal generation and the control signals are generated using the relevant address lines as shown below.

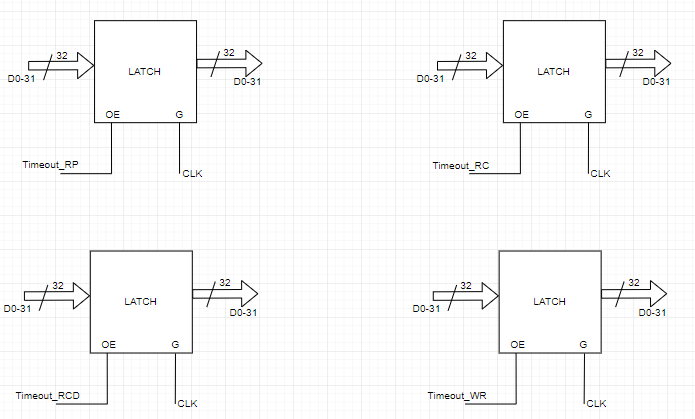
The controls signals are generated by the use of VHDL/ Verilog code.

The Latches that indicate the data transfer from the device and their logic is mentioned below.

****

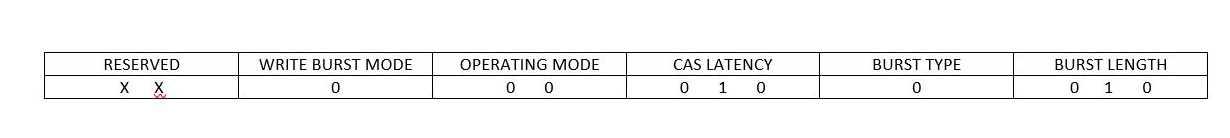
****

****

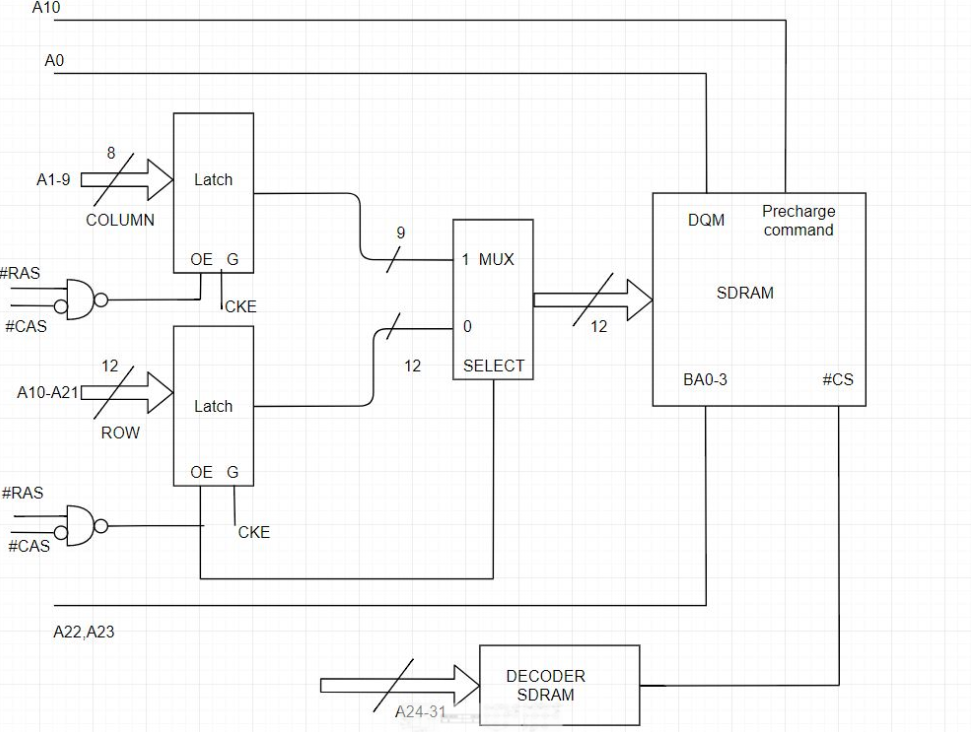
****

**Mode (Burst length = 4 & CL = 2)**

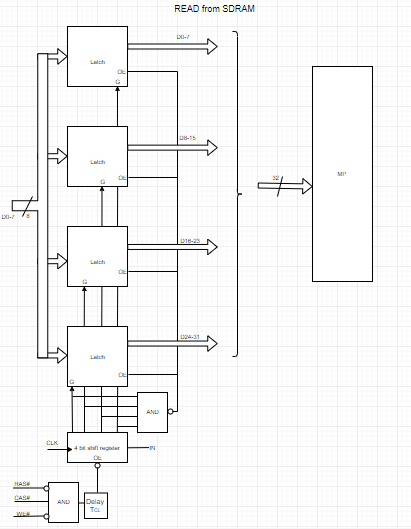
**Load Mode Register:**

****

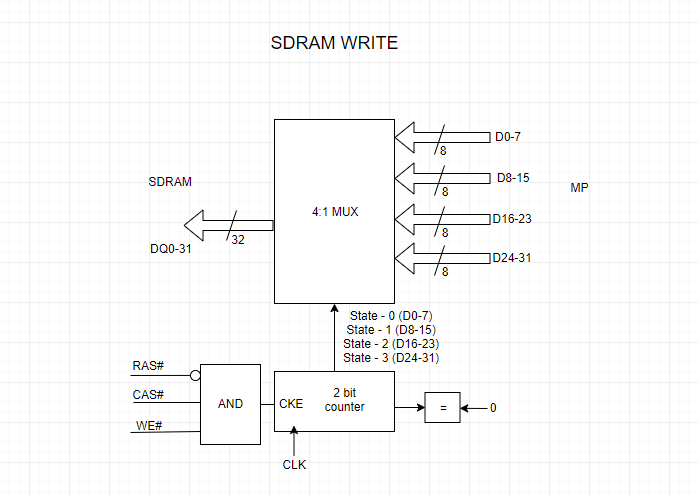
**Address Generation:**

****

**Read, SDRAM:**

****

**Write, SDRAM:**

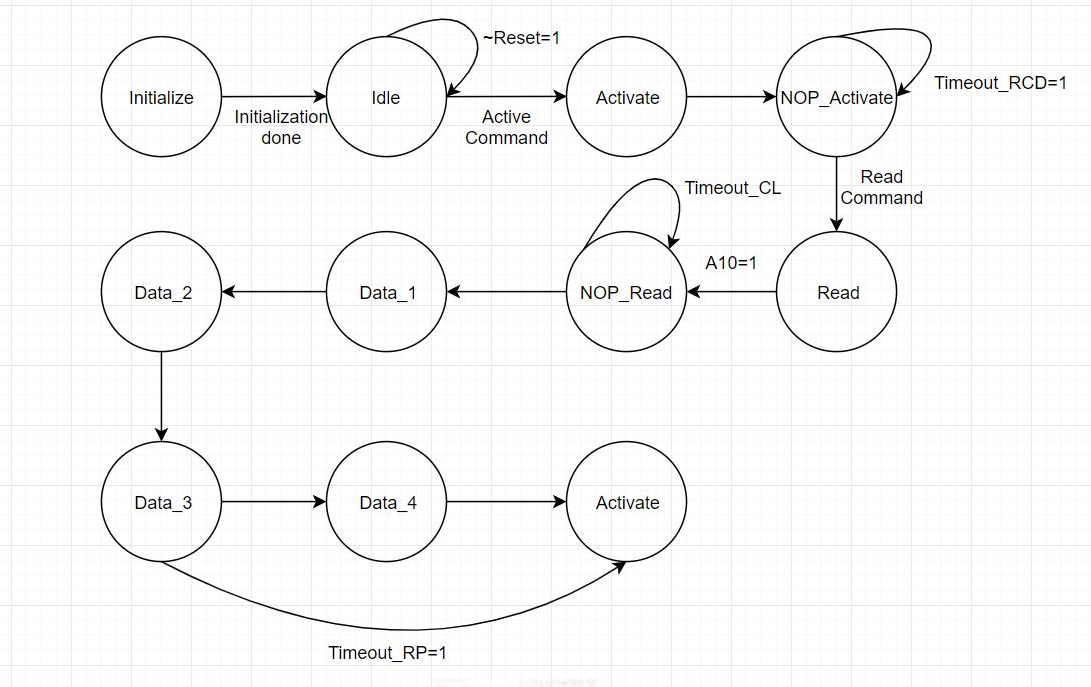
****

3

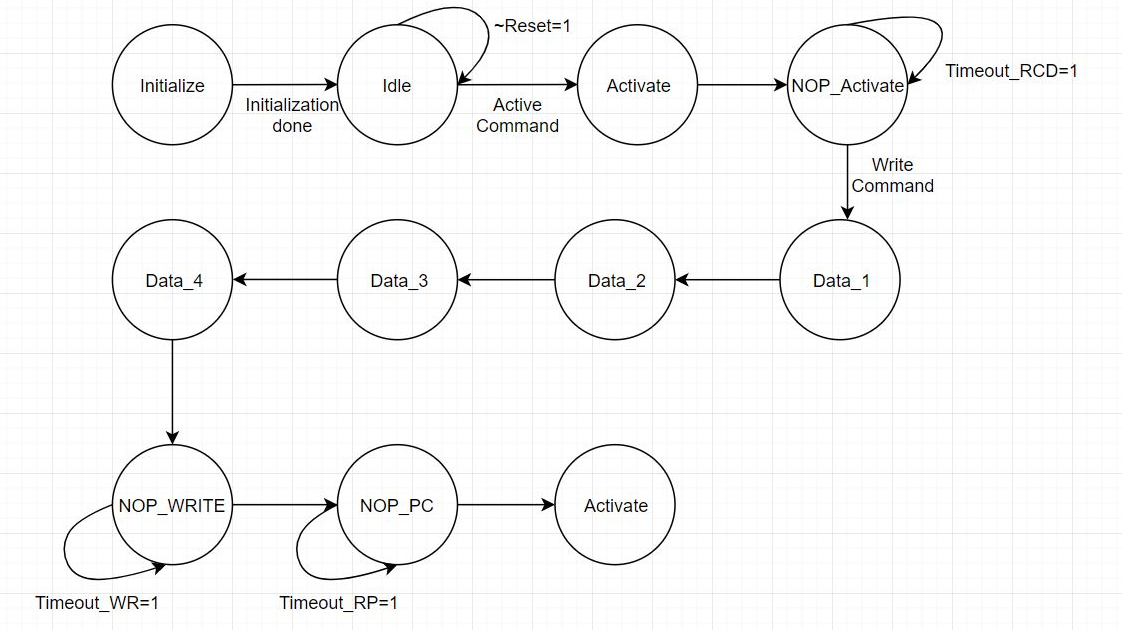
DQ0-7

8

**Read Signal:**

****

**Write Signal:**

****