Microprocessor Systems EE 5313
Project
SDRAM Controller Design
Spring 2018

<u>Project by:</u>

Banadahalli Mallesh, Suhas	**1001560048**
Mysore Veere Gowda, Manoj	**1001547261**
Veluswamy, Lokesh	**1001561782**

ABSTRACT

SDRAM, or Synchronous Dynamic Random Access Memory is a form of DRAM semiconductor memory can run at faster speeds than conventional DRAM. SDRAM memory is widely used in computers and other computing related technology. After SDRAM was introduced, further generations of double data rate RAM have entered the mass market – DDR which is also known as DDR1, DDR2, DDR3 and DDR4. The use of SDRAM was so effective that it only took about four years after its introduction in 1996/7 before its use had exceeded that of DRAM in PCs because of its greater speed of operation. Nowadays SDRAM based memory is the major type of dynamic RAM used across the computing spectrum.

SDRAM is more advantageous as it is faster & more efficient as it is up to four times the performance of standard DRAMs, Potential to eliminate the more expensive EDO/L2- cache combo, "In sync" operation it removes timing constraints and unlocks the power of advanced processors, Internal interleaving which is dual bank operation allows continuous data flow, Bursting capability is up to full page (on x16 chips), Pipelined addresses which allows a second data access to begin before the first is completed.

Objective:

The goal of this project is to design an SDRAM controller that allows SDRAM memory to be interfaced with a microprocessor having only asynchronous memory support. The requirement is that support for a single MT48LC8M16A2 be provided. Interfacing this SDRAM with any 32-bit data processor provided SDRAM memory support is not already provided for that processor. In this project we have taken 80386DX to interface with.

Features:

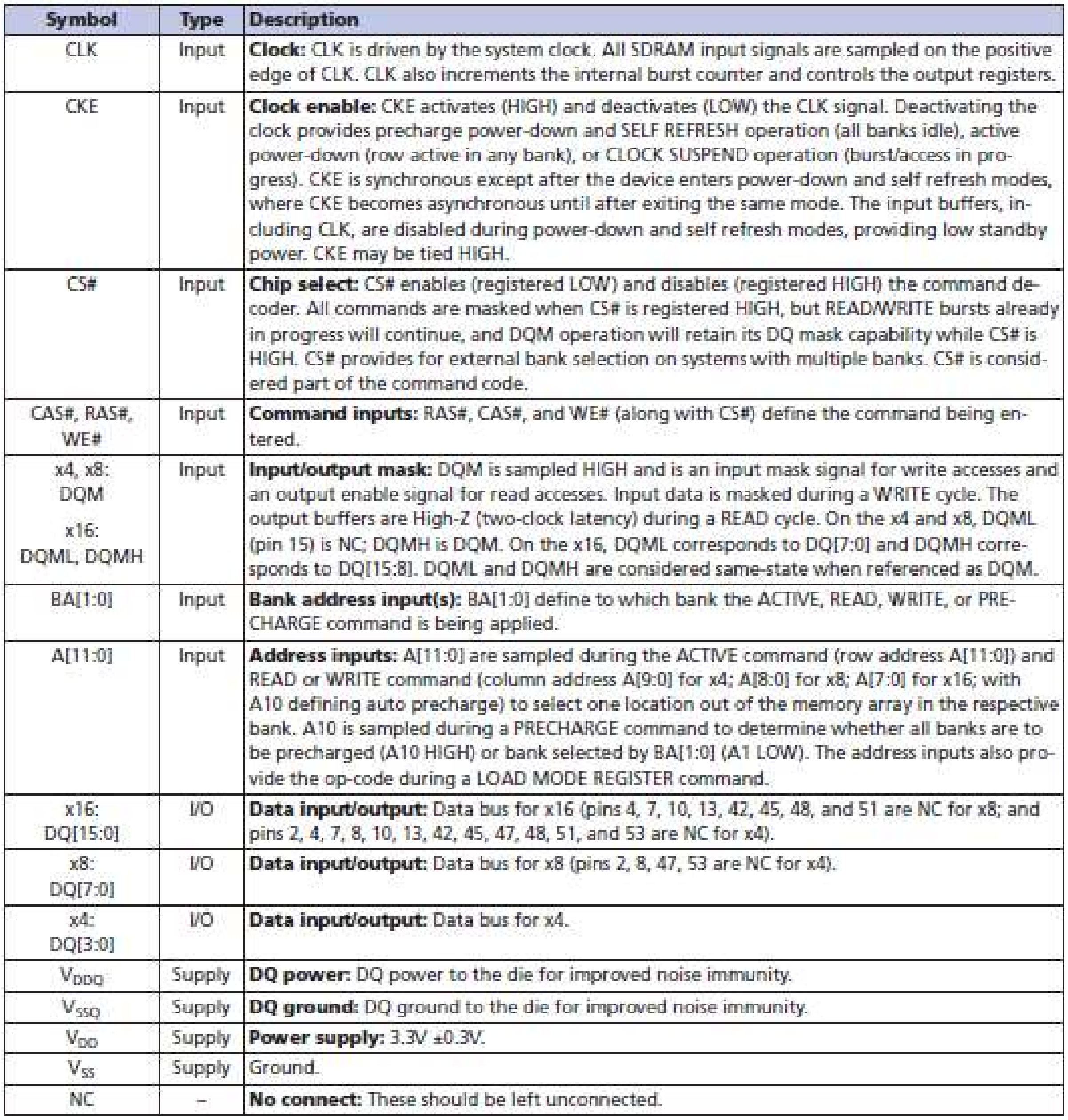
1. MT48LC4M16A2 is the memory device.
2. 80386DX with 32-bit data bus not having SDRAM support is used.
3. A complete controller solution including state machine, row, column, and bank signal generation, data masking, data flow, ready logic, and refresh support are provided.
5. Interfacing with ~ADS, W/~R, and M/~IO control signals (or equivalent) is done.
6. Supported sending of AUTO REFRESH commands at a rate sufficient to ensure memory integrity.
7. Supported READY logic of the selected microprocessor system.
8. Supported burst length 2 transfers, we have added higher burst length support for lower bit-width memories
9. Added programmability to the host processor for variable timing.

Contents

80386Dx Datasheet

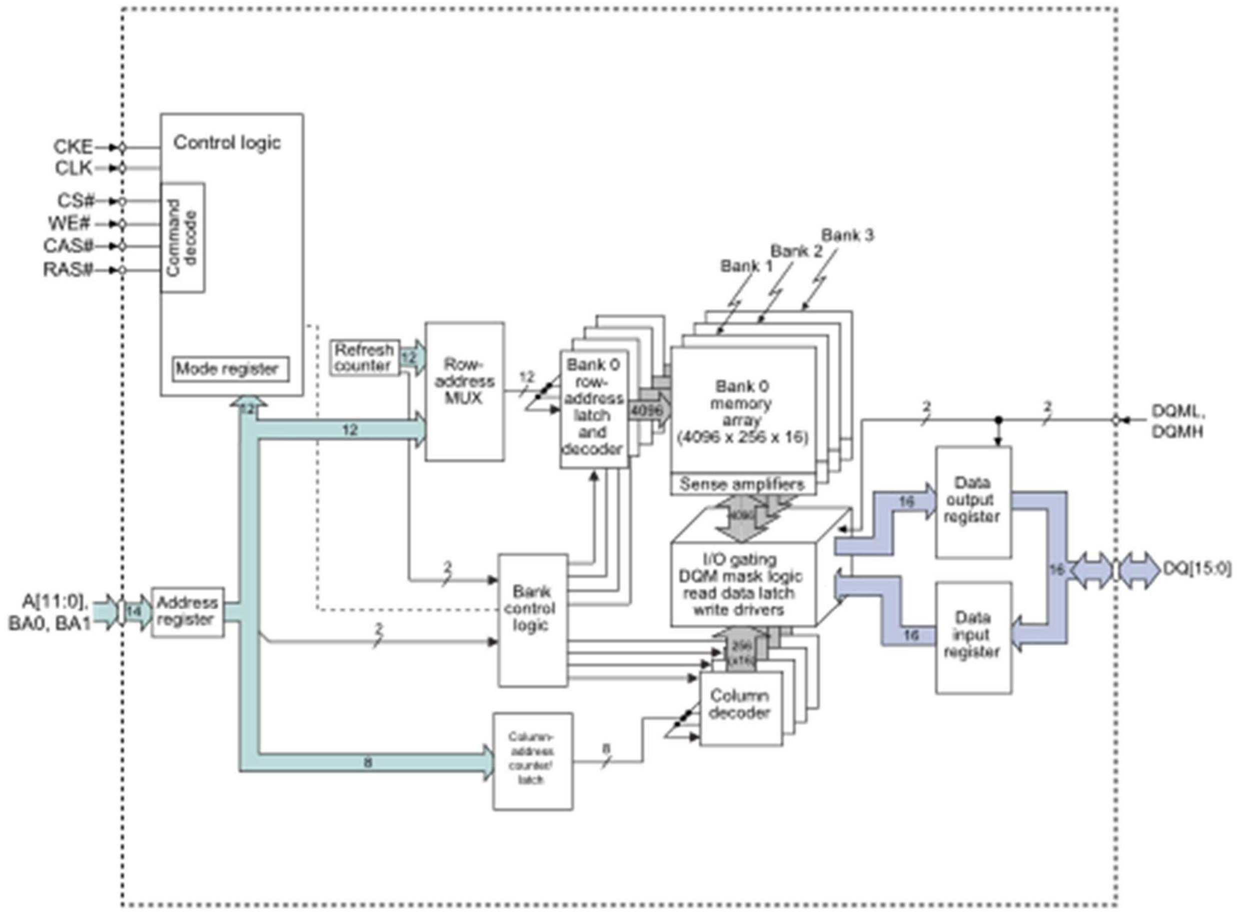
Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the Intel386 DX.
$D_{31}-D_0$	I/O	**DATA BUS** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.
$A_{31}-A_2$	O	**ADDRESS BUS** outputs physical memory or port I/O addresses.
BE0#–BE3#	O	**BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle.
W/R#	O	**WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles.
D/C#	O	**DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching.
M/IO#	O	**MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles.
LOCK#	O	**BUS LOCK** is a bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.
ADS#	O	**ADDRESS STATUS** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BE0#, BE1#, BE2#, BE3# and $A_{31}-A_2$) are being driven at the Intel386 DX pins.
NA#	I	**NEXT ADDRESS** is used to request address pipelining.
READY#	I	**BUS READY** terminates the bus cycle.
BS16#	I	**BUS SIZE 16** input allows direct connection of 32-bit and 16-bit data buses.
HOLD	I	**BUS HOLD REQUEST** input allows another bus master to request control of the local bus.
HLDA	O	**BUS HOLD ACKNOWLEDGE** output indicates that the Intel386 DX has surrendered control of its local bus to another bus master.
BUSY#	I	**BUSY** signals a busy condition from a processor extension.
ERROR#	I	**ERROR** signals an error condition from a processor extension.
PEREQ	I	**PROCESSOR EXTENSION REQUEST** indicates that the processor extension has data to be transferred by the Intel386 DX.
INTR	I	**INTERRUPT REQUEST** is a maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** is a non-maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
RESET	I	**RESET** suspends any operation in progress and places the Intel386 DX in a known reset state. See **Interrupt Signals** for additional information.
N/C	—	**NO CONNECT** should always remain unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the Intel386 DX.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides 0V connection from which all inputs and outputs are measured.

MT48LC4M16A2 (SDRAM) datasheet



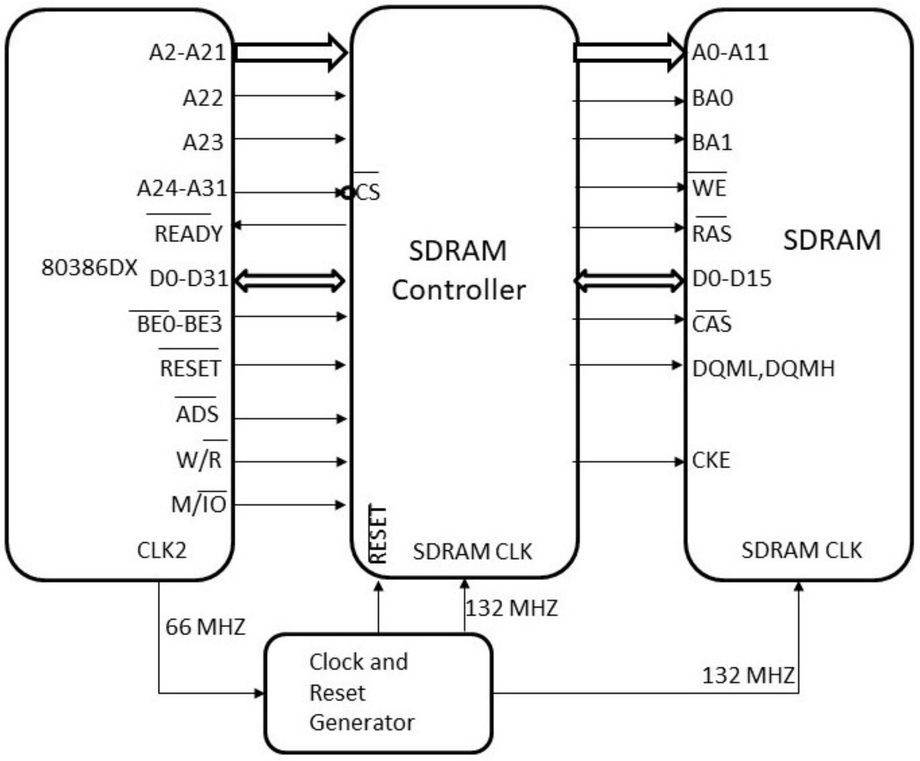
Symbol	Type	Description
CLK	Input	**Clock:** CLK is driven by the system clock. All SDRAM input signals are sampled on the positive edge of CLK. CLK also increments the internal burst counter and controls the output registers.
CKE	Input	**Clock enable:** CKE activates (HIGH) and deactivates (LOW) the CLK signal. Deactivating the clock provides precharge power-down and SELF REFRESH operation (all banks idle), active power-down (row active in any bank), or CLOCK SUSPEND operation (burst/access in progress). CKE is synchronous except after the device enters power-down and self refresh modes, where CKE becomes asynchronous until after exiting the same mode. The input buffers, including CLK, are disabled during power-down and self refresh modes, providing low standby power. CKE may be tied HIGH.
CS#	Input	**Chip select:** CS# enables (registered LOW) and disables (registered HIGH) the command decoder. All commands are masked when CS# is registered HIGH, but READ/WRITE bursts already in progress will continue, and DQM operation will retain its DQ mask capability while CS# is HIGH. CS# provides for external bank selection on systems with multiple banks. CS# is considered part of the command code.
CAS#, RAS#, WE#	Input	**Command inputs:** RAS#, CAS#, and WE# (along with CS#) define the command being entered.
x4, x8: DQM x16: DQML, DQMH	Input	**Input/output mask:** DQM is sampled HIGH and is an input mask signal for write accesses and an output enable signal for read accesses. Input data is masked during a WRITE cycle. The output buffers are High-Z (two-clock latency) during a READ cycle. On the x4 and x8, DQML (pin 15) is NC; DQMH is DQM. On the x16, DQML corresponds to DQ[7:0] and DQMH corresponds to DQ[15:8]. DQML and DQMH are considered same-state when referenced as DQM.
BA[1:0]	Input	**Bank address input(s):** BA[1:0] define to which bank the ACTIVE, READ, WRITE, or PRECHARGE command is being applied.
A[11:0]	Input	**Address inputs:** A[11:0] are sampled during the ACTIVE command (row address A[11:0]) and READ or WRITE command (column address A[9:0] for x4; A[8:0] for x8; A[7:0] for x16; with A10 defining auto precharge) to select one location out of the memory array in the respective bank. A10 is sampled during a PRECHARGE command to determine whether all banks are to be precharged (A10 HIGH) or bank selected by BA[1:0] (A1 LOW). The address inputs also provide the op-code during a LOAD MODE REGISTER command.
x16: DQ[15:0]	I/O	**Data input/output:** Data bus for x16 (pins 4, 7, 10, 13, 42, 45, 48, and 51 are NC for x8; and pins 2, 4, 7, 8, 10, 13, 42, 45, 47, 48, 51, and 53 are NC for x4).
x8: DQ[7:0]	I/O	**Data input/output:** Data bus for x8 (pins 2, 8, 47, 53 are NC for x4).
x4: DQ[3:0]	I/O	**Data input/output:** Data bus for x4.
V_{DDQ}	Supply	**DQ power:** DQ power to the die for improved noise immunity.
V_{SSQ}	Supply	**DQ ground:** DQ ground to the die for improved noise immunity.
V_{DD}	Supply	**Power supply:** 3.3V ±0.3V.
V_{SS}	Supply	Ground.
NC	–	**No connect:** These should be left unconnected.

FUNCTIONAL BLOCK DIAGRAM
(4 Meg x 16 SDRAM)



SDRAM CONTROLLER INTERFACING

The main aim of the project is to design a SDRAM controller which is interfaced to 80386DX microprocessor and a SDRAM memory device MT48LC4M16A2 (1Meg x 16 x 4banks). The 80386DX processor has 32-bit address line. These 32-bit address lines are used to interface the controller to a memory device. As we know, the SDRAM memory has 4 banks that is of 16Mib each. The controller is designed such that it supports the SDRAM to the various operations such as Initializations, Read, Write and Auto Refresh. The organization of SDRAM memory consists of 4096 rows, 256 columns of 16 bit width. Therefore there are 12 address lines to be interfaced with the processor.

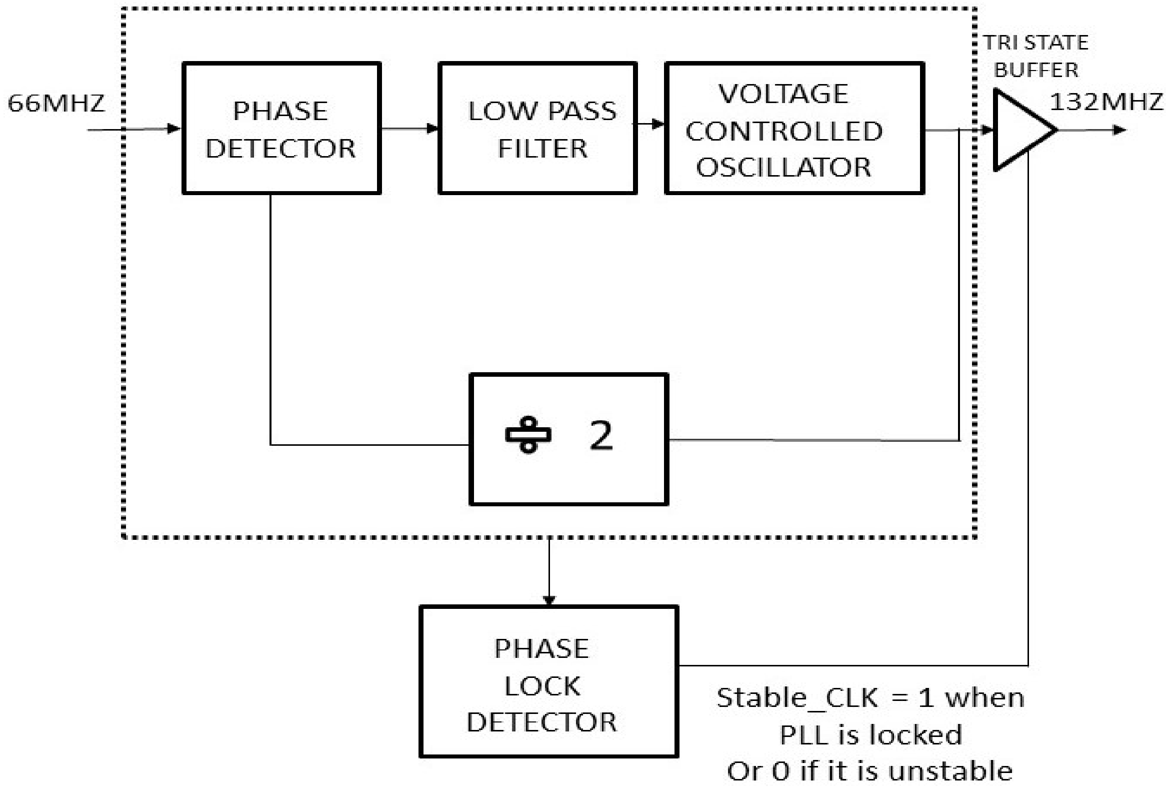


The above is the interfacing of 80386DX processor to SDRAM memory device with a SDRAM controller.

1. The DQML/ DQMH signal is generated from the BE0#, BE1#, BE2# and BE3# from the microprocessor.
2. The address lines A2-A13 of Microprocessor are mapped to A0-A11 address lines of SDRAM, where A2-A9 are mapped for column address A0-A7. The lines A8, A9 and A11 of the SDAM are don't care (x) for the READ/WRITE command.
3. The A12 address line is mapped to A10 of the SDRAM and used to (i). select Auto / Manual Pre-charging while READ/WRITE command (ii). Select All banks or particular bank while closing the row after Read/write if the Manual Pre-charging is issued.
4. The address lines A10-A21 of the Microprocessor is mapped to A0-A11 of the SDRAM while ACTIVATE command (2^12 columns).
5. A22, A23 is directly given to BA0 and BA1.

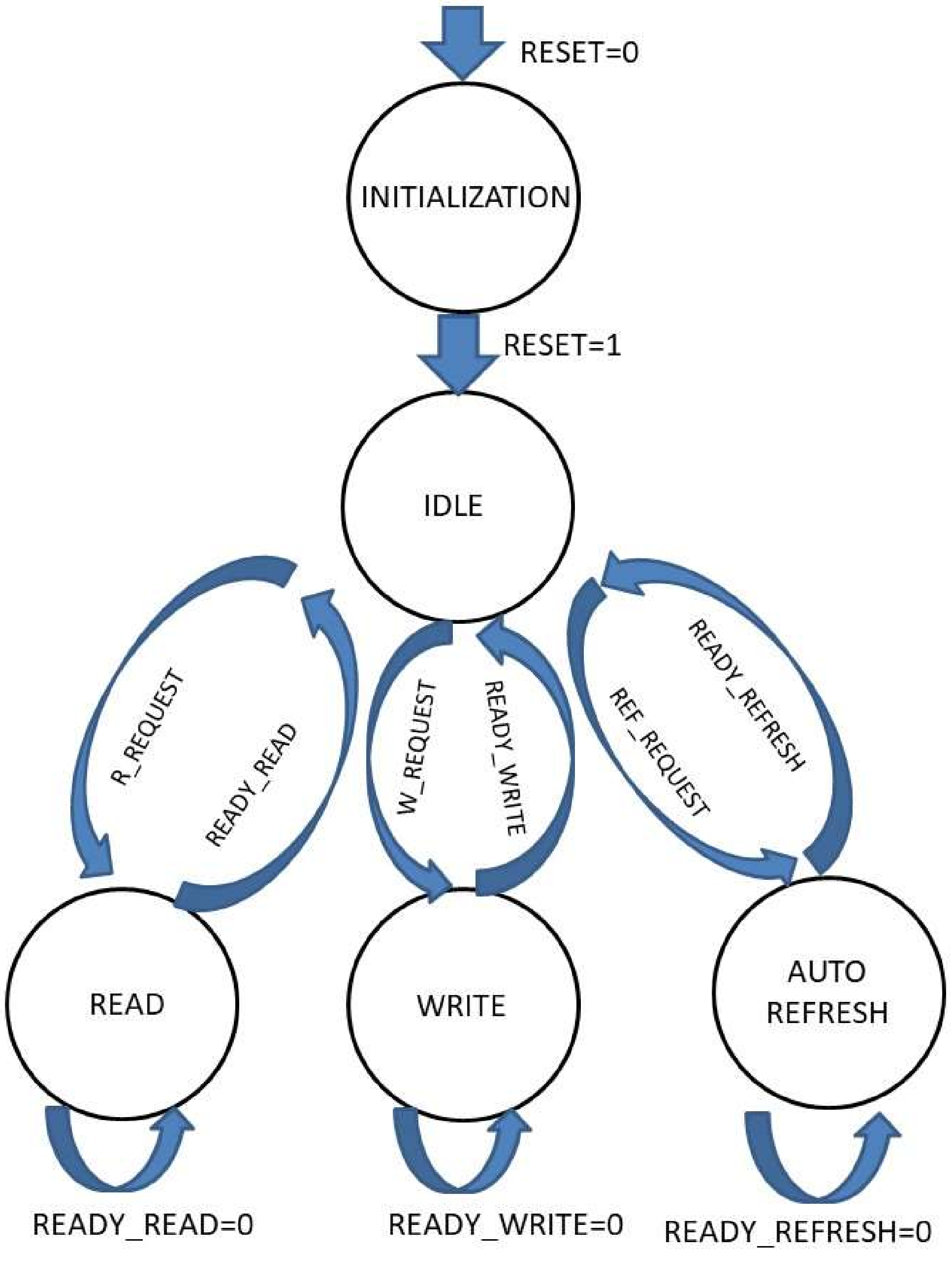
6. The data bus consists of D0-31.
7. CS#, the chip select signal is enabled using address lines A24-31.
8. The ADS#, W/R# and M/IO# signals are mapped to WE#, RAS# and CAS# using a combinational logic.
9. The clock signal is generated using a clock Generator. This clock generator takes the input from the CLK2 of the microprocessor which by default configured as 66Mhz.
10. The phase lock loop is used to obtain the clock signal that is sent to the microcontroller and SDRAM.

PHASE LOCK LOOP (PLL)

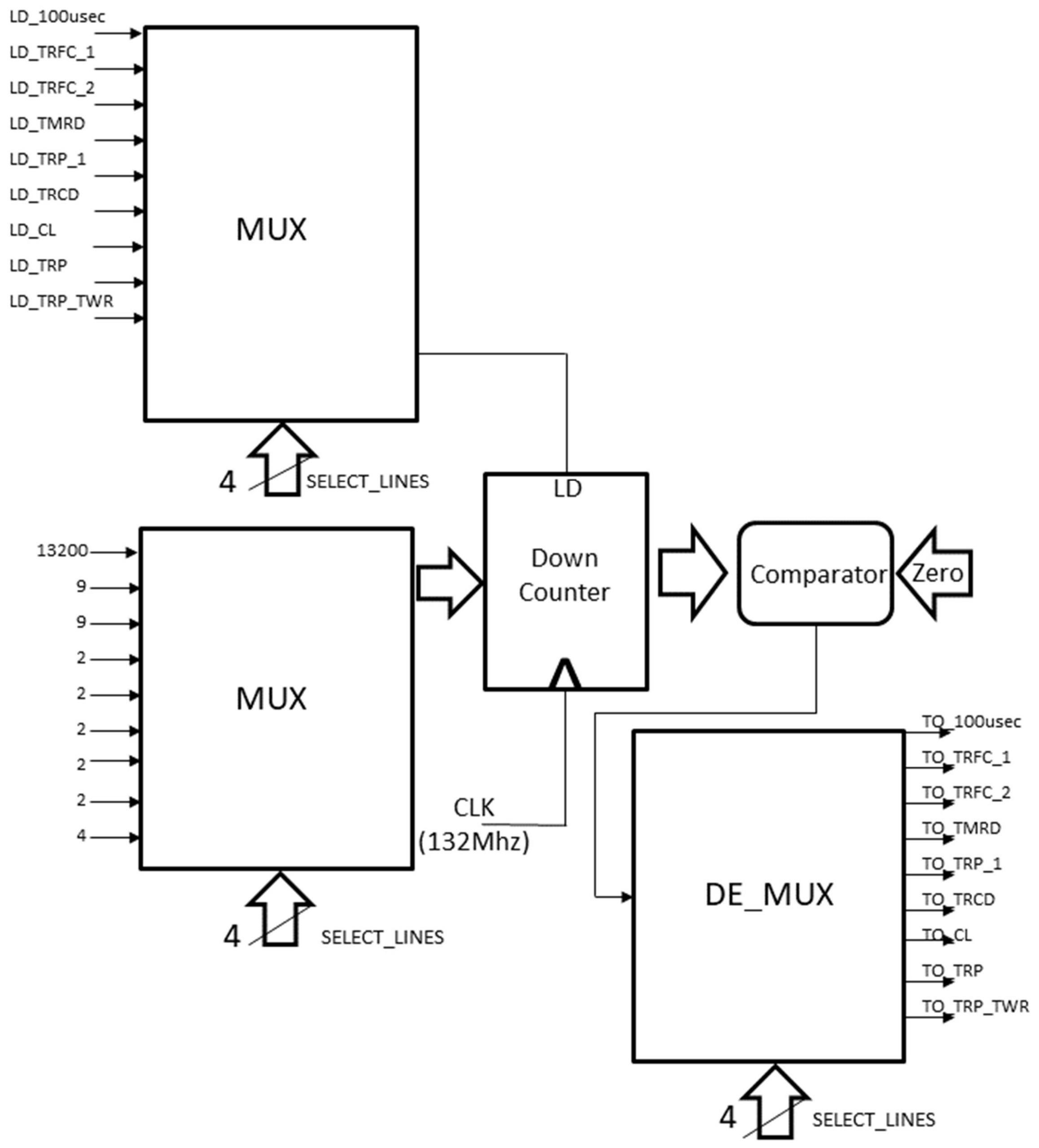

66MHZ
PHASE DETECTOR
LOW PASS FILTER
VOLTAGE CONTROLLED OSCILLATOR
TRI STATE BUFFER
132MHZ
÷ 2
PHASE LOCK DETECTOR
Stable_CLK = 1 when PLL is locked Or 0 if it is unstable

The pre-scaler is used to pre-scale the input from the microprocessor. The Phase detector is used to determine the Phase output that is given to Low pass filter. The voltage control oscillator is used to set the voltage accordingly and then to scale the output voltage. Thus, the output clock is given as clock signal for SDRAM controller and SDRAM memory device.

HIGH-LEVEL FINITE STATE MACHINE

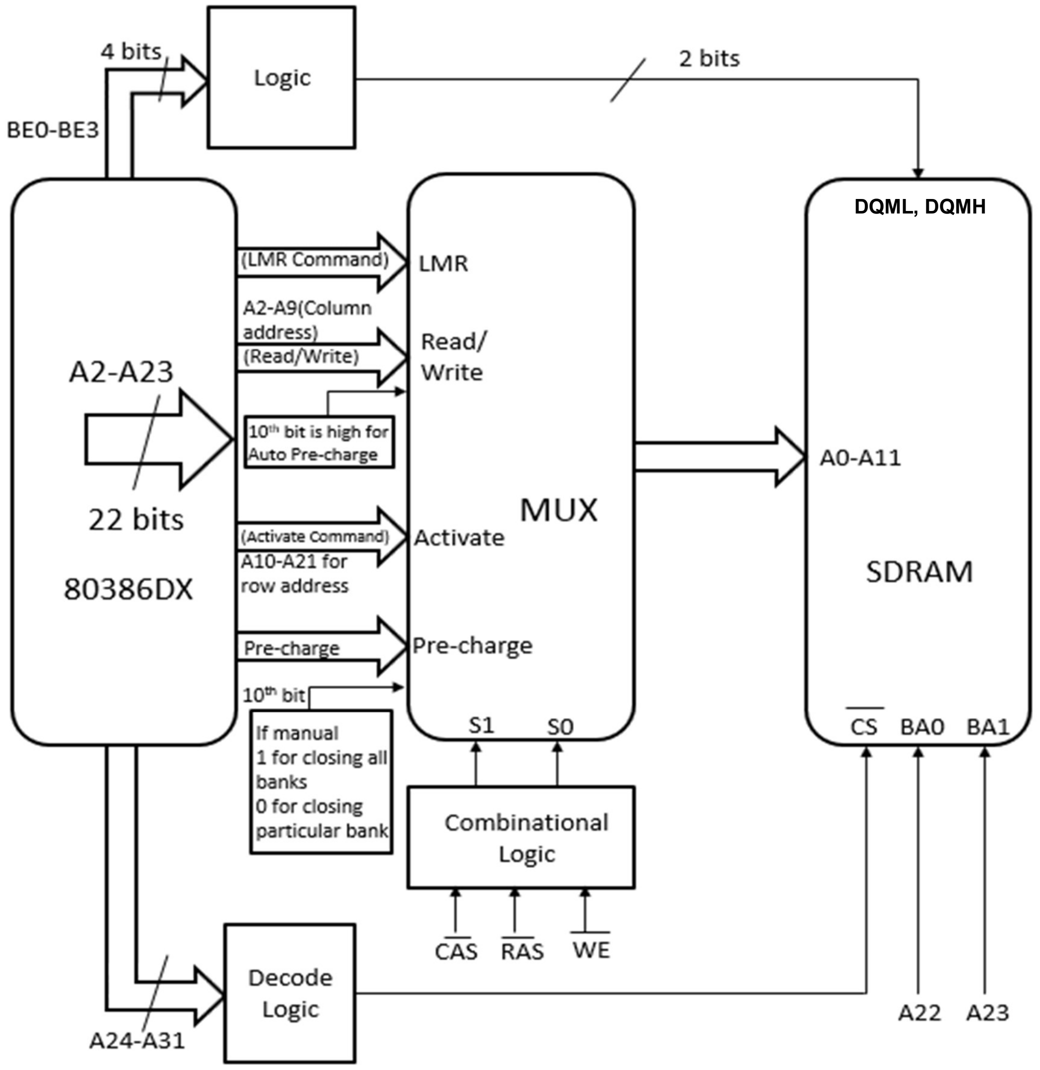


UNIVERSAL COUNTER FOR INTRODUCING DELAYS



ADDRESS GENERATION

The 4 x 1 multiplexer is used to load the address lines to the SDRAM based on the select lines that are used as control signals to the MUX that enables one of the four modes. Those 4 modes are LMR, READ/WRITE, ACTIVATE, PRECHARGE.

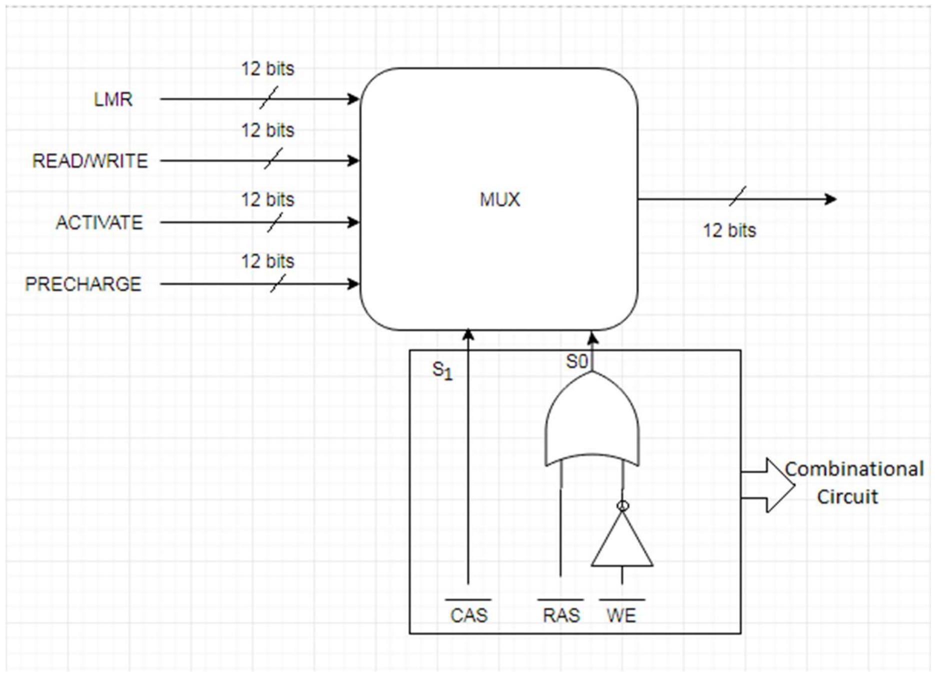


1. **GENERATION OF SELECT LINES (S0 & S1):**

Truth table:

S1	S0	COMMAND	CAS#	RAS#	WE#
0	0	LMR	L	L	L
0	1	R/W	L	H	X
1	0	ACTIVE	H	L	H
1	1	PRECHARGE	H	L	L

<u>Logic Diagram:</u>

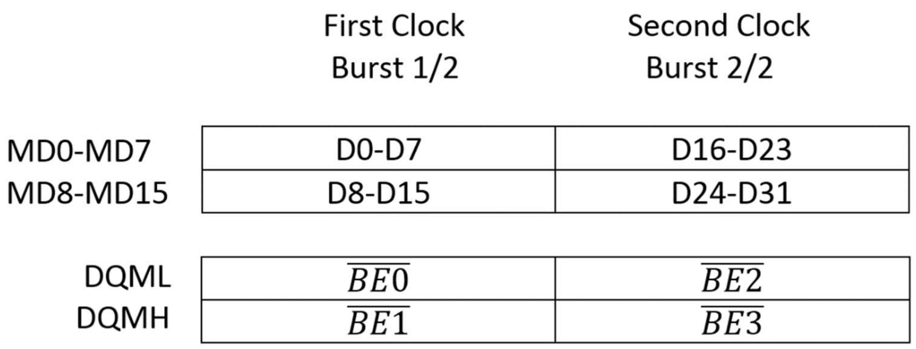


<u>Signal equation:</u>

S1 = $\overline{CAS}$
S0 = $\overline{RAS}$+WE

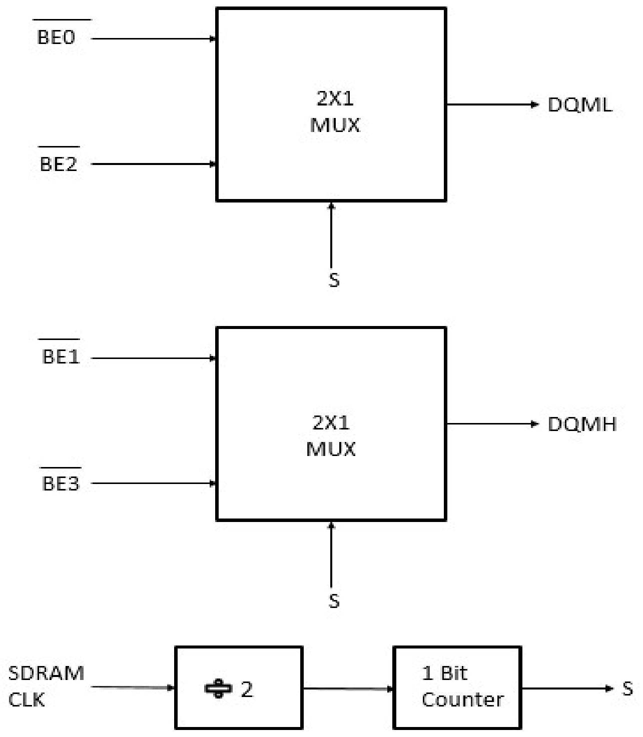
2. GENERATION OF DQMH, DQML SIGNALS:

<u>BEx – DQMx mapping:</u>



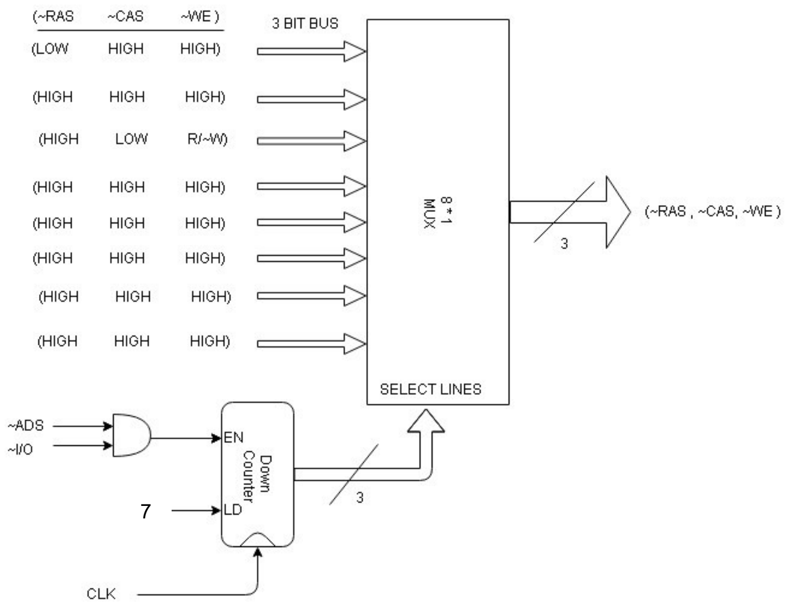
	First Clock Burst 1/2	Second Clock Burst 2/2
MD0-MD7	D0-D7	D16-D23
MD8-MD15	D8-D15	D24-D31
DQML	$\overline{BE0}$	$\overline{BE2}$
DQMH	$\overline{BE1}$	$\overline{BE3}$

Logic Diagram:



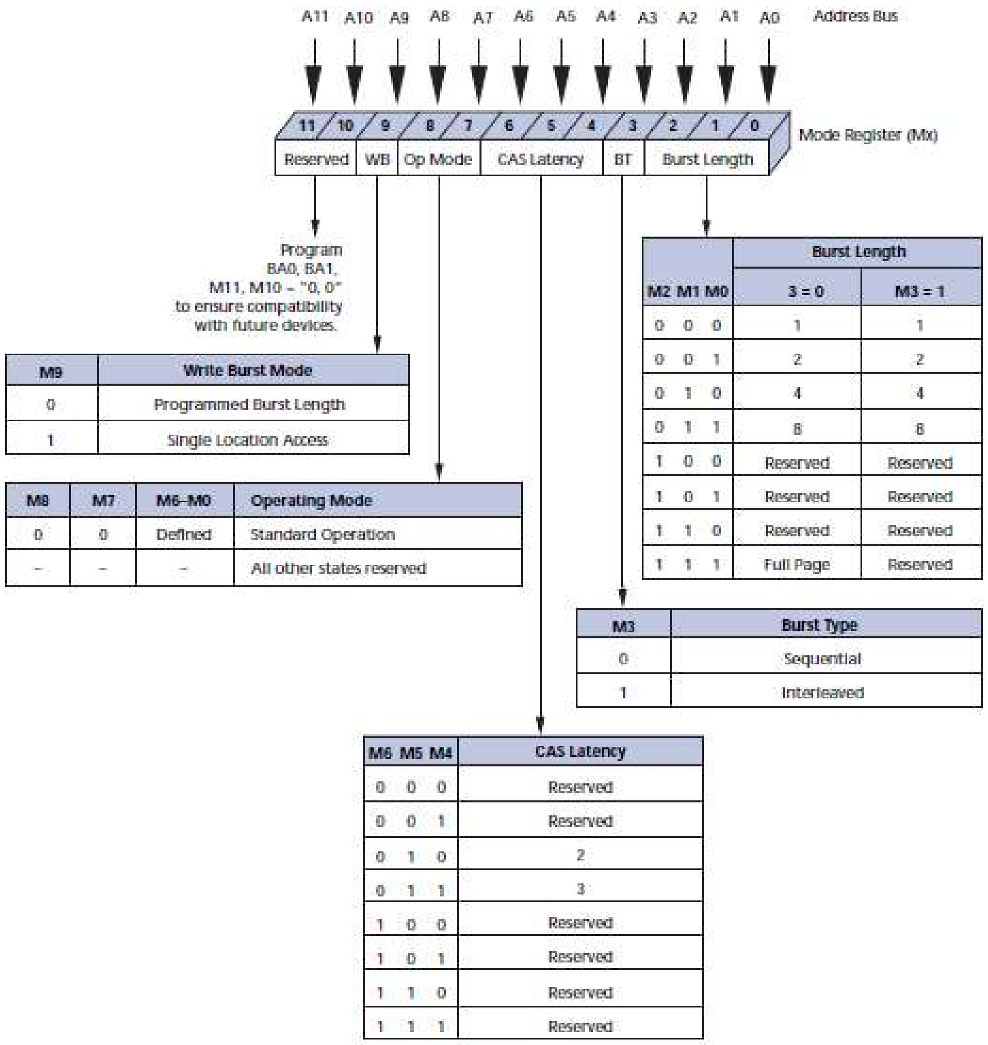
The selection of DQML and DQMH completely is arbitrary. We have chosen in such a way that the data bits D0-15 of the microprocessor will have the data of the first burst and D16-31 will have the data of the second burst.

3. GENERATION OF RAS#,CAS# AND WE# FOR READ/WRITE:



LOAD MODE REGISTER

The specific mode operations of the SDRAM are defined using this load mode register i.e., this defines the Burst length, burst type, CAS latency, operating mode and Write burst mode. This loads the A0-A11 of the SDRAM. The project is designed for a Burst length of 2 and the CAS latency is set to 2. This defines the variant of the SDRAM.
The LMR will change based on this BL and CAS latency. The below is the load mode register with BL = 2 and CL = 2.



A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
Address Bus
11 10 9 8 7 6 5 4 3 2 1 0
Mode Register (Mx)
Reserved
WB
Op Mode
CAS Latency
BT
Burst Length
Program BA0, BA1, M11, M10 = "0, 0" to ensure compatibility with future devices.

M2 M1 M0	Burst Length	
	3 = 0	M3 = 1
0 0 0	1	1
0 0 1	2	2
0 1 0	4	4
0 1 1	8	8
1 0 0	Reserved	Reserved
1 0 1	Reserved	Reserved
1 1 0	Reserved	Reserved
1 1 1	Full Page	Reserved

M9	Write Burst Mode
0	Programmed Burst Length
1	Single Location Access

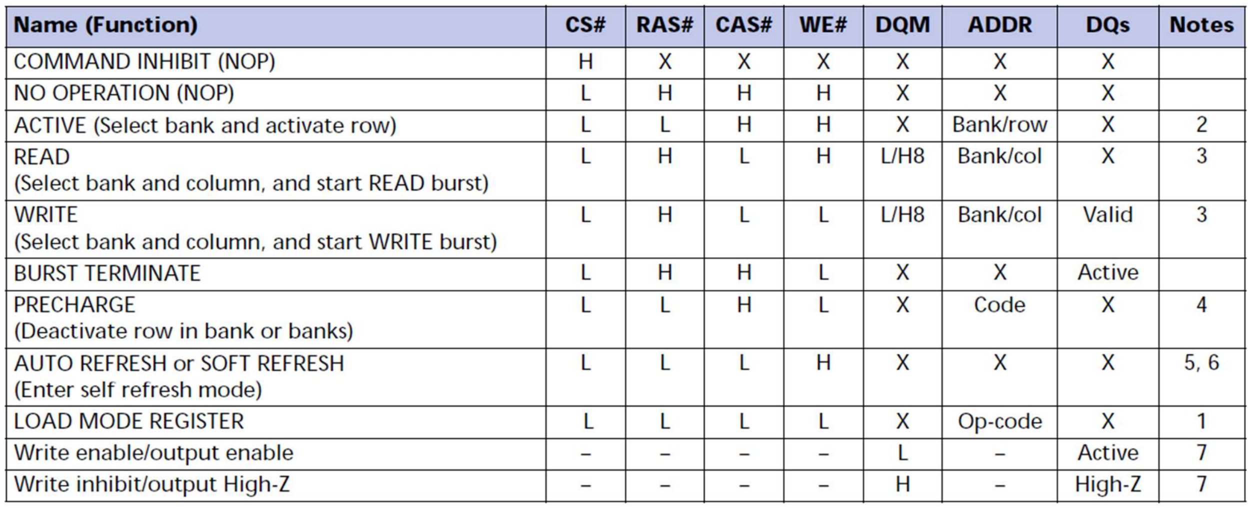
M8	M7	M6–M0	Operating Mode
0	0	Defined	Standard Operation
–	–	–	All other states reserved

M3	Burst Type
0	Sequential
1	Interleaved

M6 M5 M4	CAS Latency
0 0 0	Reserved
0 0 1	Reserved
0 1 0	2
0 1 1	3
1 0 0	Reserved
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

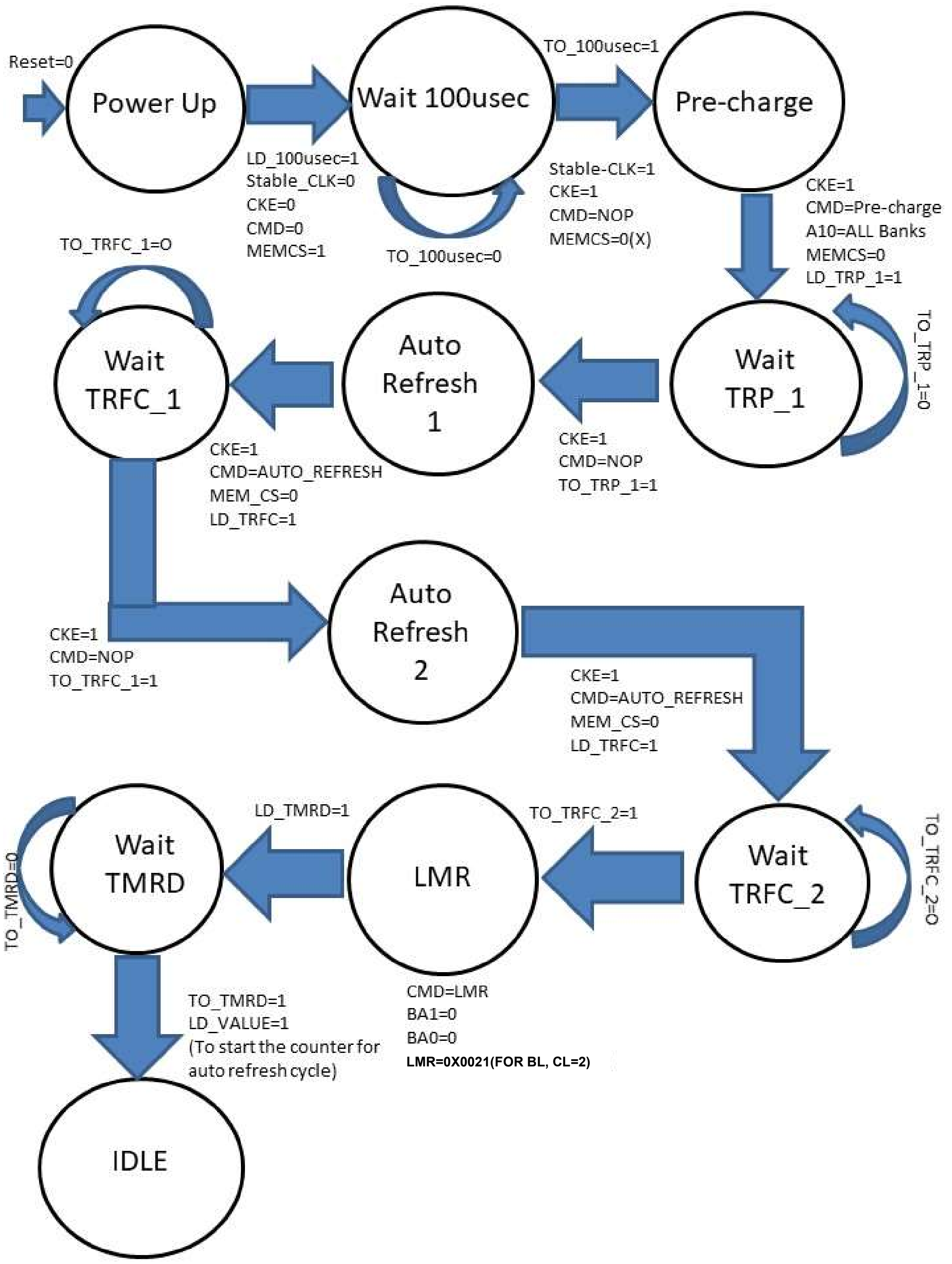
Burst Length = 2.
Burst type is sequential.
CAS Latency is 2.
Operating mode is Standard Operation.
Write Burst is a programmable Burst Length.
Hence the LMR value will be 0x0021

COMMANDS AND DQMs OPERATIONS



Name (Function)	CS#	RAS#	CAS#	WE#	DQM	ADDR	DQs	Notes
COMMAND INHIBIT (NOP)	H	X	X	X	X	X	X	
NO OPERATION (NOP)	L	H	H	H	X	X	X	
ACTIVE (Select bank and activate row)	L	L	H	H	X	Bank/row	X	2
READ (Select bank and column, and start READ burst)	L	H	L	H	L/H8	Bank/col	X	3
WRITE (Select bank and column, and start WRITE burst)	L	H	L	L	L/H8	Bank/col	Valid	3
BURST TERMINATE	L	H	H	L	X	X	Active	
PRECHARGE (Deactivate row in bank or banks)	L	L	H	L	X	Code	X	4
AUTO REFRESH or SOFT REFRESH (Enter self refresh mode)	L	L	L	H	X	X	X	5, 6
LOAD MODE REGISTER	L	L	L	L	X	Op-code	X	1
Write enable/output enable	–	–	–	–	L	–	Active	7
Write inhibit/output High-Z	–	–	–	–	H	–	High-Z	7

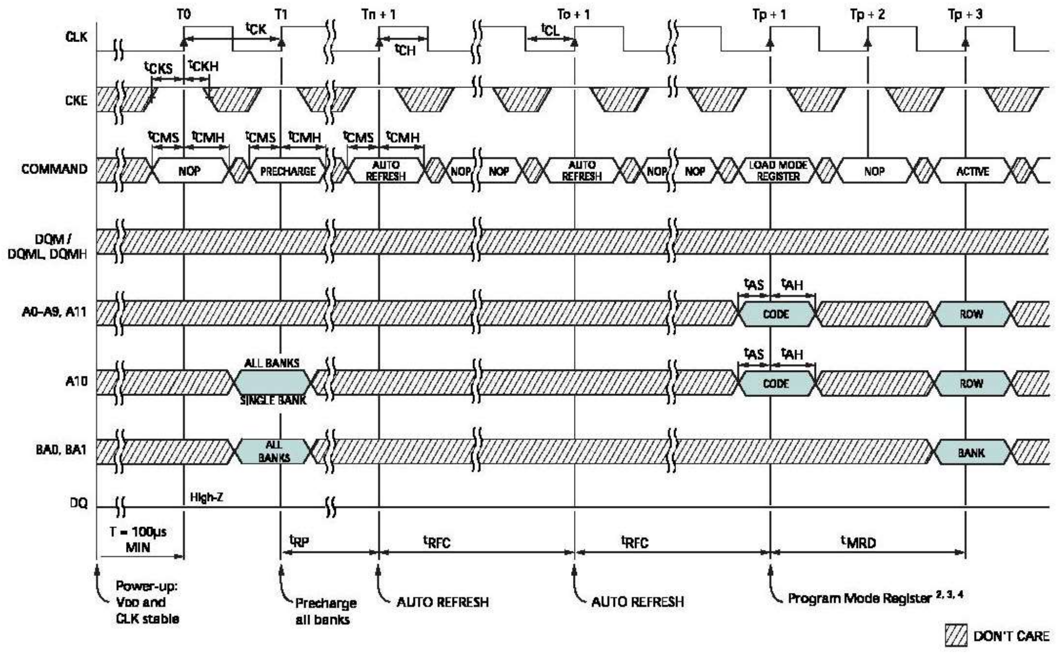
1. **INITIALIZATION**
   - **Finite state machine**



The 13 steps of Initialization is done immediately when the RESET signal is asserted. The above FSM is designed based on these 13 steps that is provided in the SDRAM Datasheet. The counters for the wait time used in the FSM is designed above.

For this project, we are considering CL = 2. As we are using 132 MHz clock for the SDRAM, we are using -7E variant of the MT48LC4M16A2.
The clock time (Tck) for this SDRAM chip is 7.5 ns(1/132MHz).

- **Timing Diagram for Initialization of SDRAM**



- **The recommended power-up sequence for SDRAMs:**

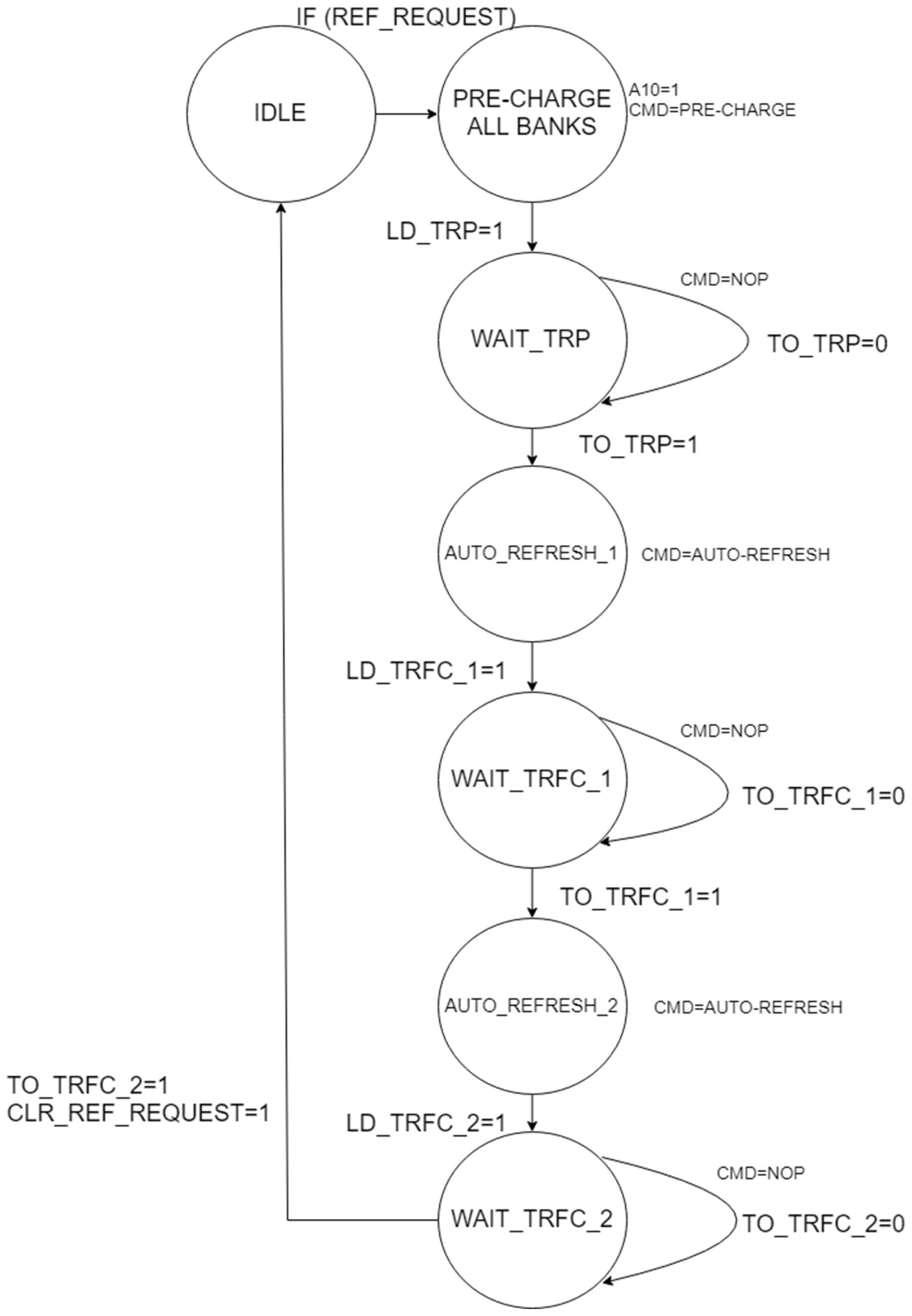
1. Simultaneously apply power to VDD and VDDQ.
2. Assert and hold CKE at a LVTTL logic LOW since all inputs and outputs are LVTTLcompatible.
3. Provide stable CLOCK signal. Stable clock is defined as a signal cycling within timing constraints specified for the clock pin.
4. Wait at least 100µs prior to issuing any command other than a COMMAND INHIBITor NOP.
5. Starting at some point during this 100µs period, bring CKE HIGH. Continuing at least through the end of this period, 1 or more COMMAND INHIBIT or NOP commands must be applied.
6. Perform a PRECHARGE ALL command.
7. Wait at least tRP time; during this time NOPs or DESELECT commands must be given. All banks will complete their precharge, thereby placing the device in the all banks idle state.
8. Issue an AUTO REFRESH command.
9. Wait at least tRFC time, during which only NOPs or COMMAND INHIBIT commands are allowed.
10. Issue an AUTO REFRESH command.
11. Wait at least tRFC time, during which only NOPs or COMMAND INHIBIT commands are allowed.
12. The SDRAM is now ready for mode register programming. Because the mode register will power up in an unknown state, it should be loaded with desired bit values prior to applying any operational command. Using the LOAD MODE REGISTER command, program the mode register. The mode register is programmed via the MODE REGISTER SET command with BA1 = 0, BA0 = 0 and retains the stored information until it is programmed again or the device loses power. Not programming the mode register upon initialization will result in default settings which may not be desired. Outputs are guaranteed High-Z after the LOAD MODE REGISTER command is issued. Outputs should be High-Z already before the LOAD MODE REGISTER command is issued.
13. Wait at least tMRD time, during which only NOP or DESELECT commands are allowed

- **State transition table for initialization:**

STATE	**CONDITION**	**NEXT STATE**
X	$\overline{\text{RESET}}$=0	POWER_UP
POWER_UP	STABLE_CLK=0	POWER_UP
POWER_UP	STABLE_CLK=1	WAIT_100usec
WAIT_100usec	TIMEOUT_100usec=0	WAIT_100usec
WAIT_100usec	TIMEOUT_100usec=1	PRECHARGE_ALL
PRECHARGE_ALL	X	WAIT_TRP_1
WAIT_TRP_1	TIMEOUT_TRP_1=0	WAIT_TRP_1
WAIT_TRP_1	TIMEOUT_TRP_1=1	AUTO_REFRESH_1
AUTO_REFRESH_1	X	WAIT_TRFC_1
WAIT_TRFC_1	TIMEOUT_TRFC_1=0	WAIT_TRFC_1
WAIT_TRFC_1	TIMEOUT_TRFC_1=1	AUTO_REFRESH_2
AUTO_REFRESH_2	X	WAIT_TRFC_2
WAIT_TRFC_2	TIMEOUT_TRFC_2=0	WAIT_TRFC_2
WAIT_TRFC_2	TIMEOUT_TRFC_2=1	LMR
LMR	X	WAIT_TMRD
WAIT_TMRD	TIMEOUT_TMRD=0	WAIT_TMRD
WAIT_TMRD	TIMEOUT_TMRD=1	IDLE

2. AUTO REFRESH

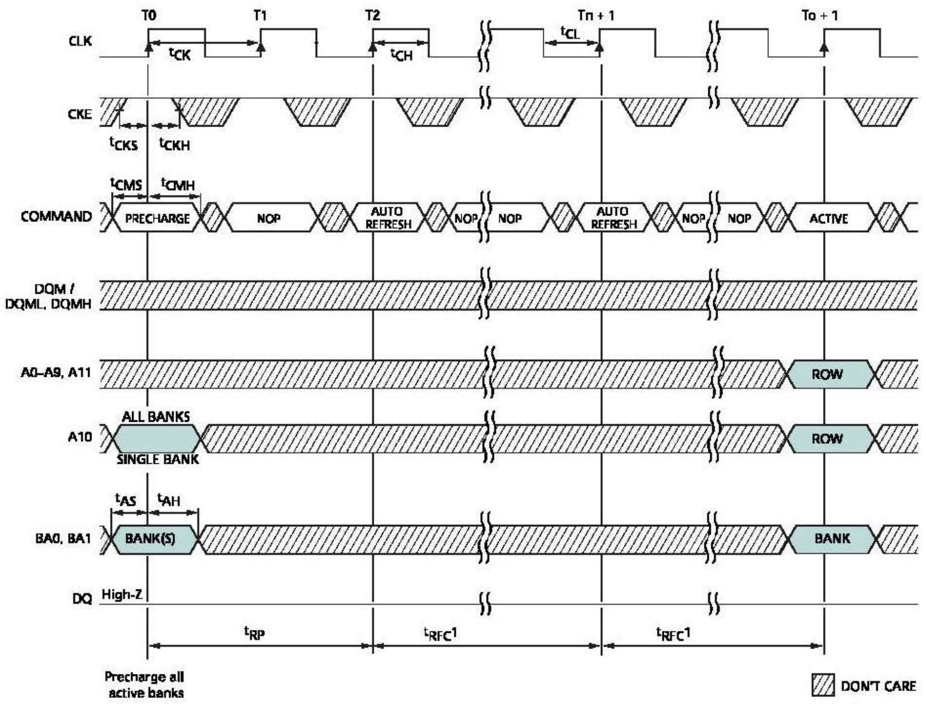
Finite state machine



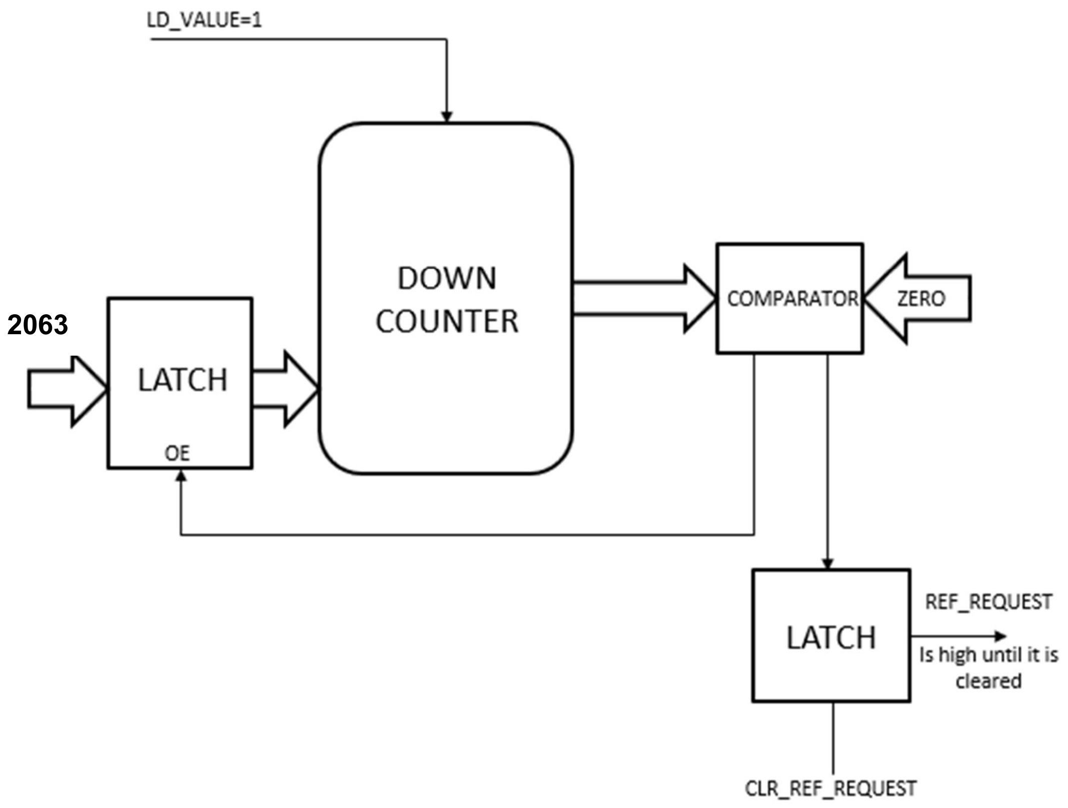
State transition table

STATE	CONDITION	NEXT STATE
IDLE	If(REF_REQUEST)	PRECHARGE
PRECHARGE	X	WAIT_TRP
WAIT_TRP	TO_TRP=0	WAIT_TRP
WAIT_TRP	TO_TRP=1	AUTO_REFRESH_1
AUTO_REFRESH_1	X	WAIT_TRFC_1
WAIT_TRFC_1	TO_TRFC_1=0	WAIT_TRFC_1
WAIT_TRFC_1	TO_TRFC_1=1	AUTO_REFRESH_2
AUTO_REFRESH_2	X	WAIT_TRFC_2
WAIT_TRFC_2	TO_TRFC_2=0	WAIT_TRFC_2
WAIT_TRFC_2	TO_TRFC_2=1	IDLE

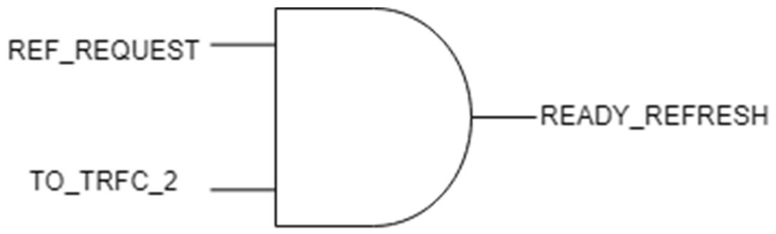
Timing Diagram:



Generation of REF_REQUEST signal:
The signal REF_REQUEST is generated every 15.625usec irrespective of read and write cycles. All the rows has to be refreshed every 64ms. Hence the AUTO_REFRESH cycles has to run 4096 times in 64ms to refresh all the 4096 rows. Below is the counter that generates REF_REQUEST every 15.625usec.

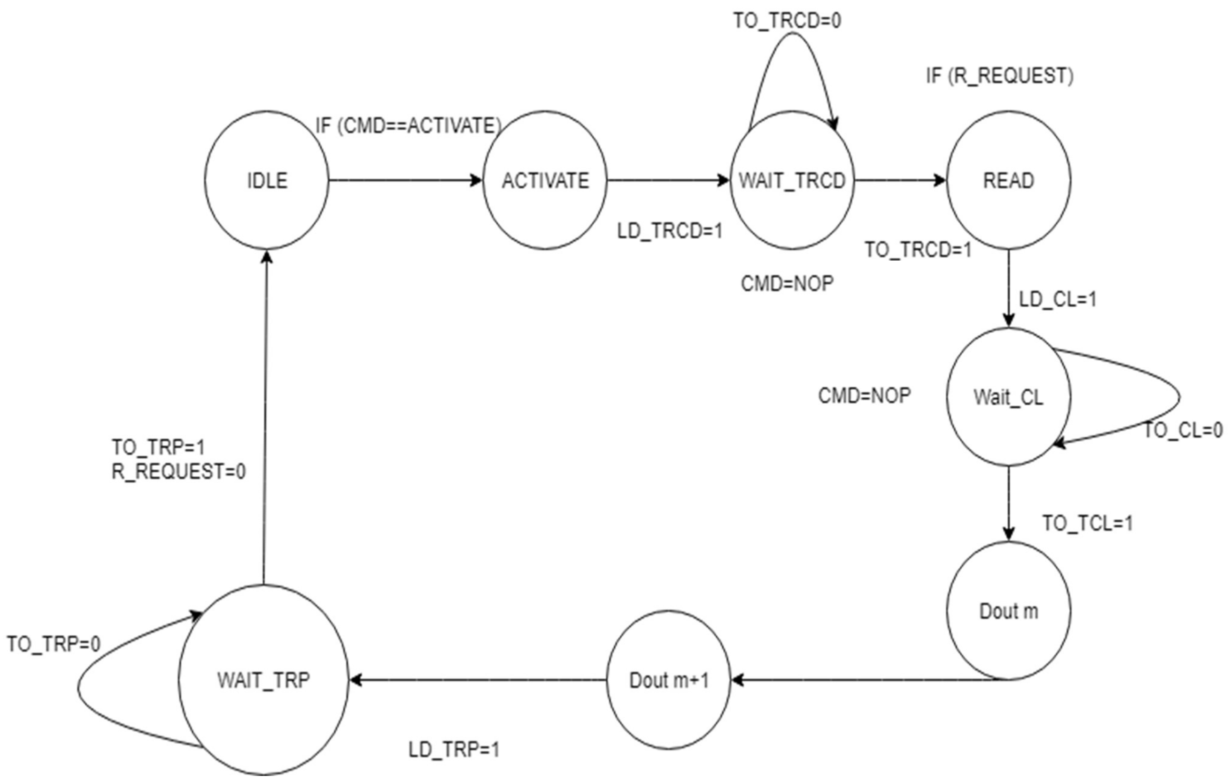


Generation of READY_REFRESH signal:

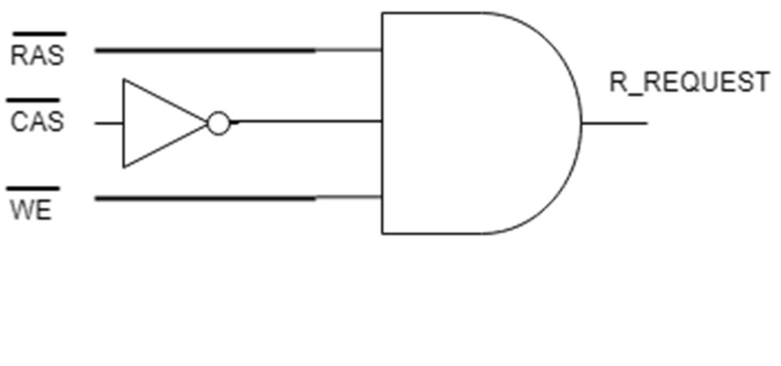


3. READ

- **Finite State machine**



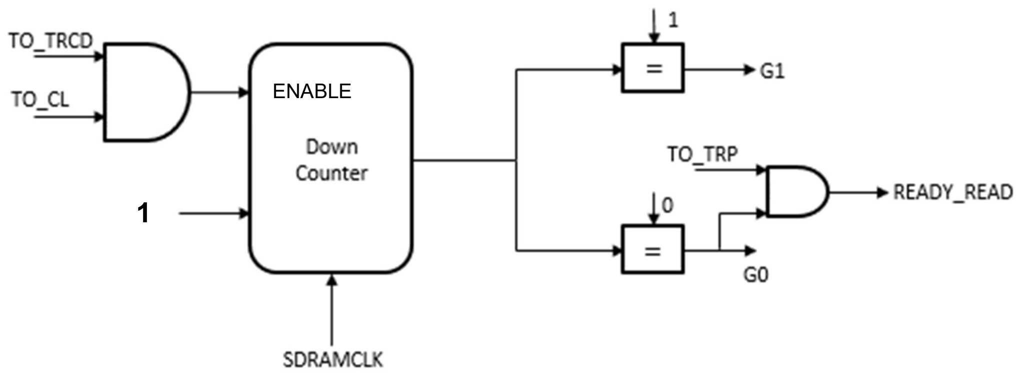
- **Generation of R_REQUEST signal:**



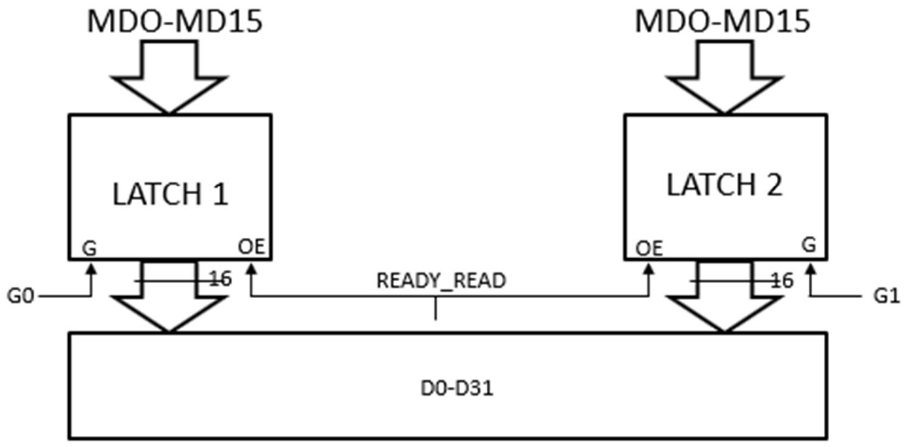
- **State transition table**

STATE	CONDITION	NEXT STATE
IDLE	If(CMD==ACTIVATE)	ACTIVATE
ACTIVATE	X	WAIT_TRCD
WAIT_TRCD	TO_TRCD=0	WAIT_TRCD
WAIT_TRCD	TO_TRCD=1	READ
READ	X	WAIT_CL
WAIT_CL	TO_CL=0	WAIT_TCL
WAIT_CL	TO_CL=1	Dout m
Dout m	X	Dout m+1
Dout m+1	X	WAIT_TRP
WAIT_TRP	TO_TRP=0	WAIT_TRP
WAIT_TRP	TO_TRP=1	IDLE

- **Generation of READY signal for Read:**

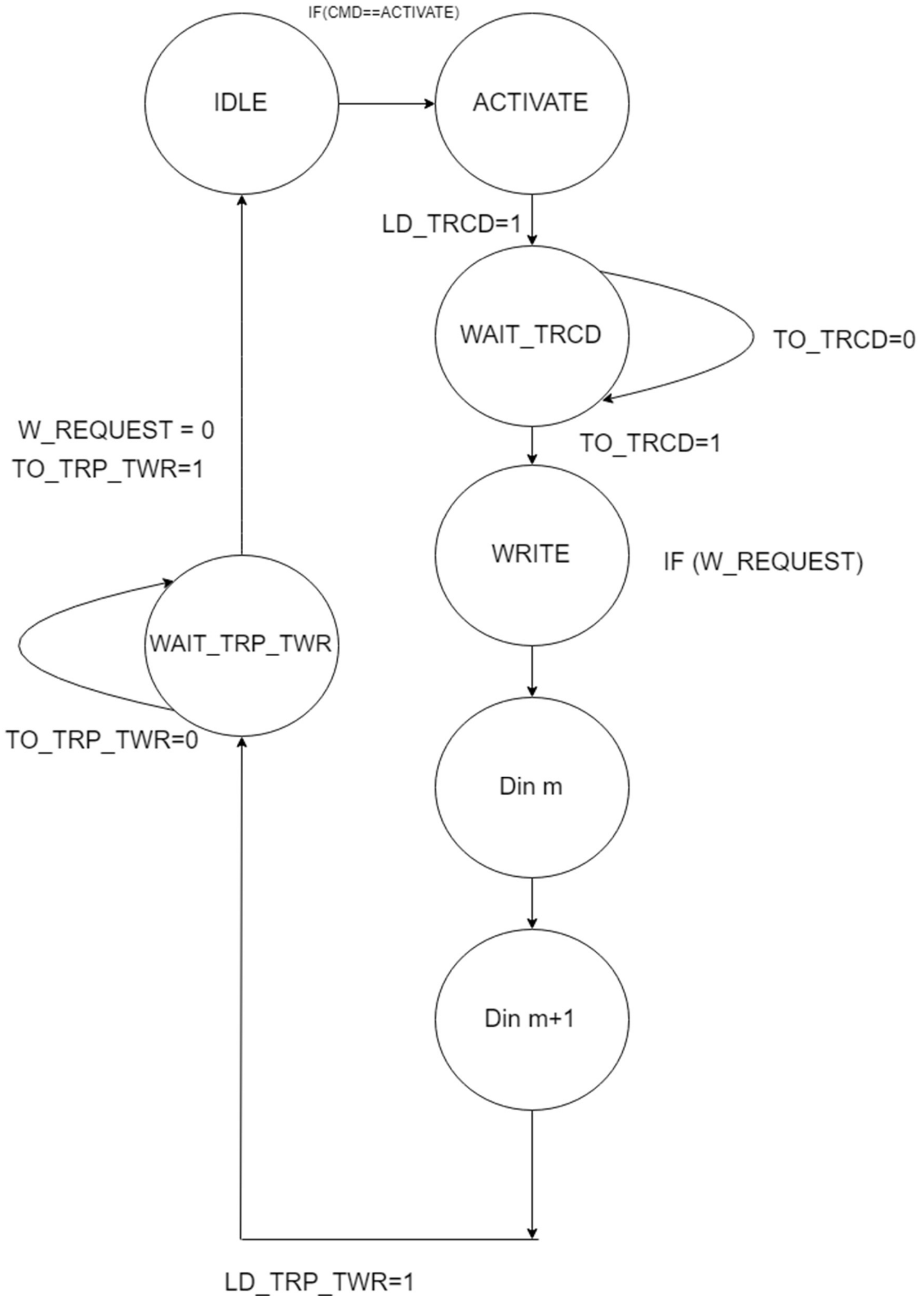


- **To output 32-bit data after 2 bursts:**



4. WRITE

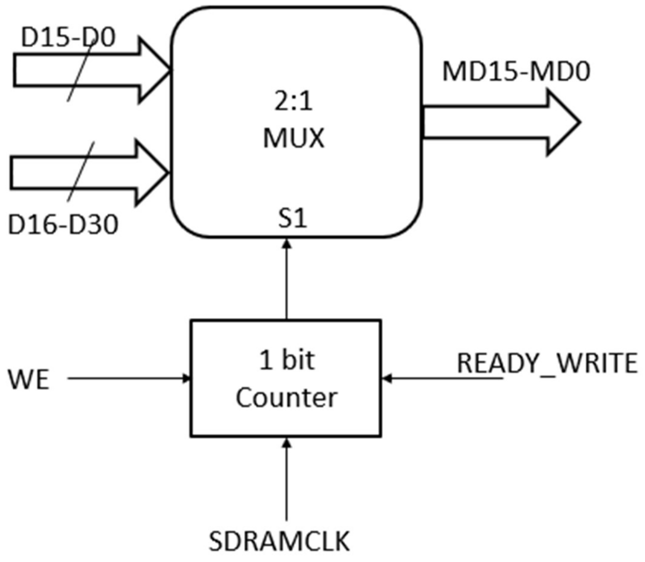
- **Finite state machine**



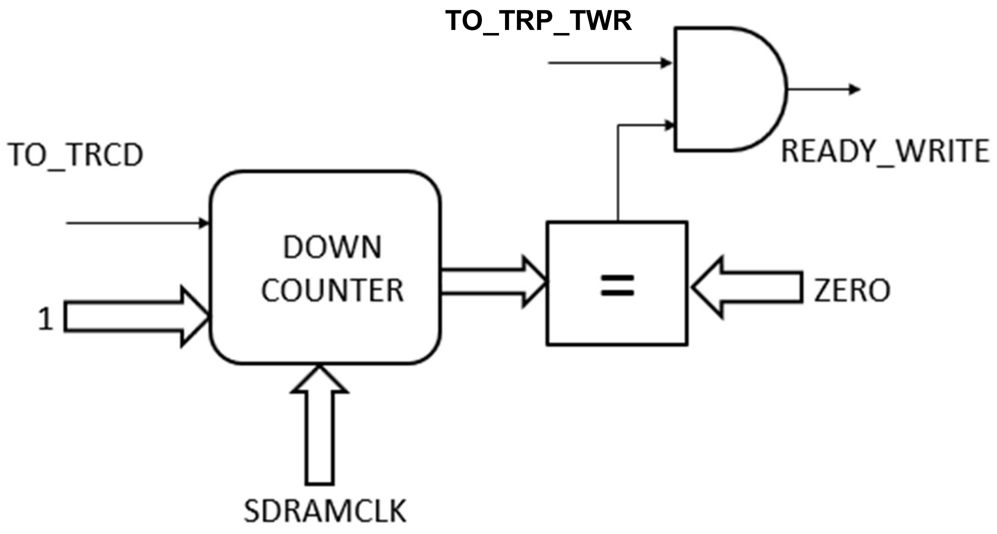
- **State transition table**

STATE	CONDITION	NEXT STATE
IDLE	If(CMD==ACTIVATE)	ACTIVATE
ACTIVATE	X	WAIT_TRCD
WAIT_TRCD	TO_TRCD=0	WAIT_TRCD
WAIT_TRCD	TO_TRCD=1	WRITE
WRITE	X	Din m
Din m	X	Din m+1
Din m+1	X	WAIT_TRP_TWR
WAIT_TRP_TWR	TO_TRP_TWR=0	WAIT_TRP_TWR
WAIT_TRP_TWR	TO_TRP_TWR=1	IDLE

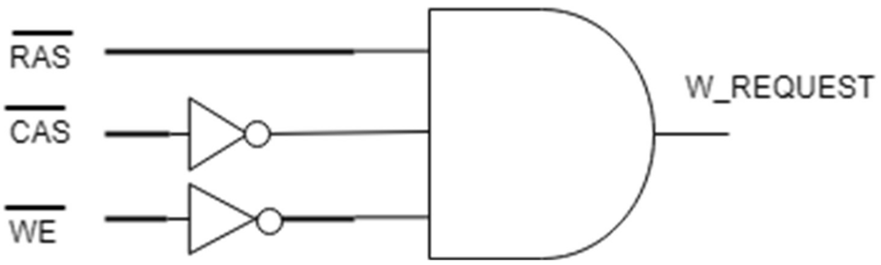
- **To write data to memory in 2 bursts:**



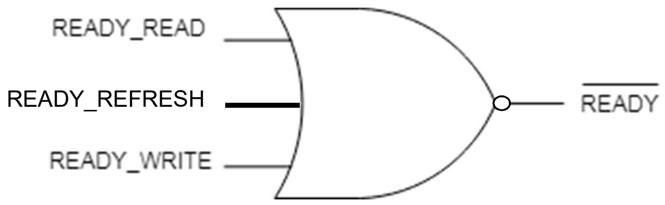
- **Generation of READY signal after write cycle:**



- **Generation of W_REQUEST**



READY SIGNAL GENERATION AFTER READ, WRITE OR REFRESH



EXTRA CREDIT

I. ADDING PROGRAMMABILITY:

For programmability we have used 6 registers, as rest all signals and parameters doesn't change on the basis of configuration. We are using 6 channels for 6 type of combinations. The combination is as follows.

MT48LC16M4A2	**MT48LC8M8A2**	**MT48LCM4M16A2**
BL=8	BL=4	BL=2
CL=2,3	CL=2,3	CL=2,3

- **Register mapping for variable parameters:**

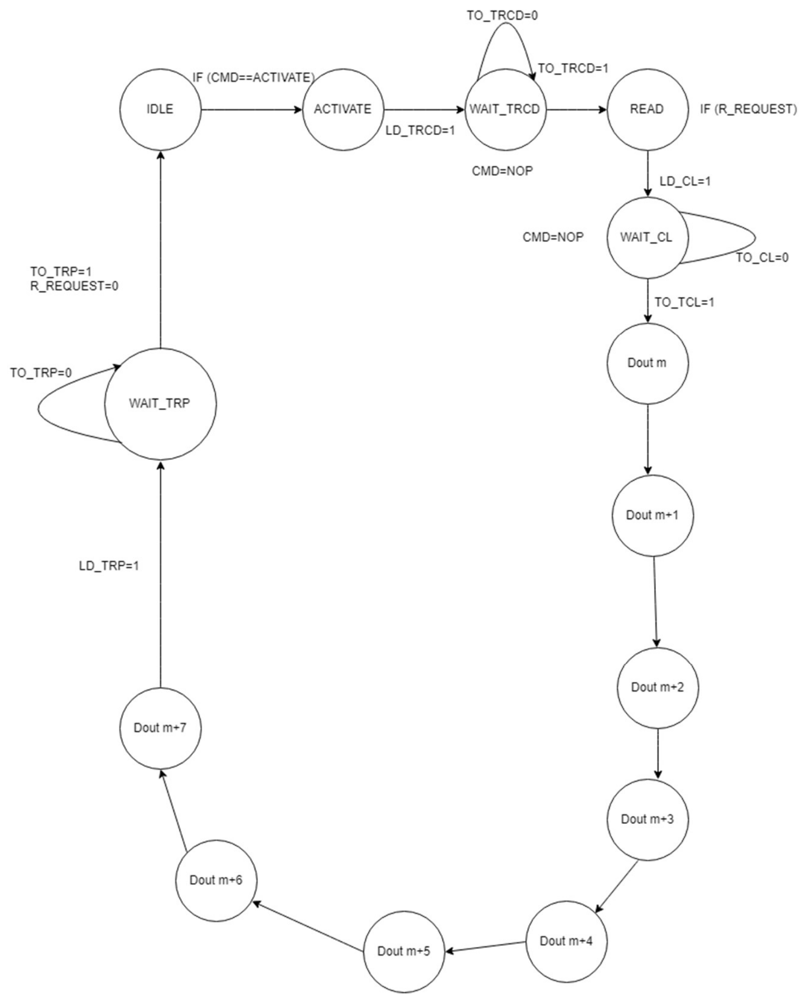
Offset	**MSB**	**LSB**
0	Burst Length	CAS Latency
2	TRP_1	TRCD
4	TCL	TRP

- **Timer values for wait states:**

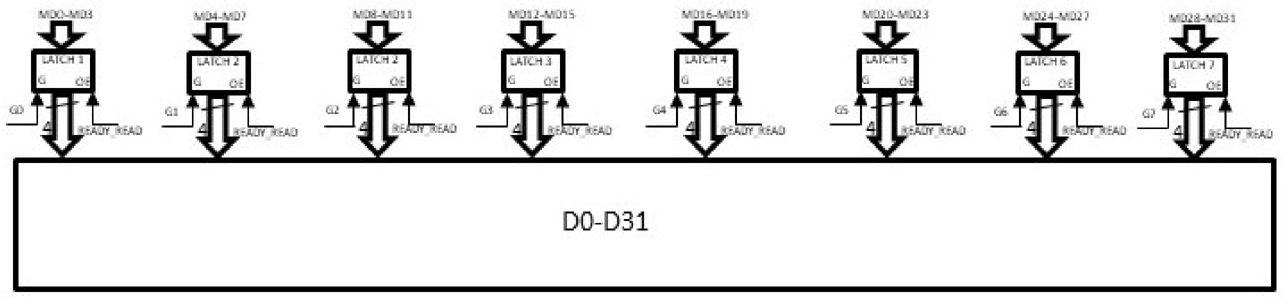
Timer	**Parameters**	**Delays**	
		CL=2	**CL=3**
T100usec	Wait 100us	13300	13300
TRFC_1	Refresh Cycle 1	9	9
TRFC_2	Refresh Cycle 2	9	9
TMRD	LMR delay	2	2
TRP_1	Pre-charge 1	2	3
TRCD	Row to column delay	2	3
TCL	CAS Latency	2	3
TRP	Pre-charge	2	3
TRP_TWR	Pre-charge + Write recovery time	4	4

II. A)MT48LC16M4A2(BL = 8)

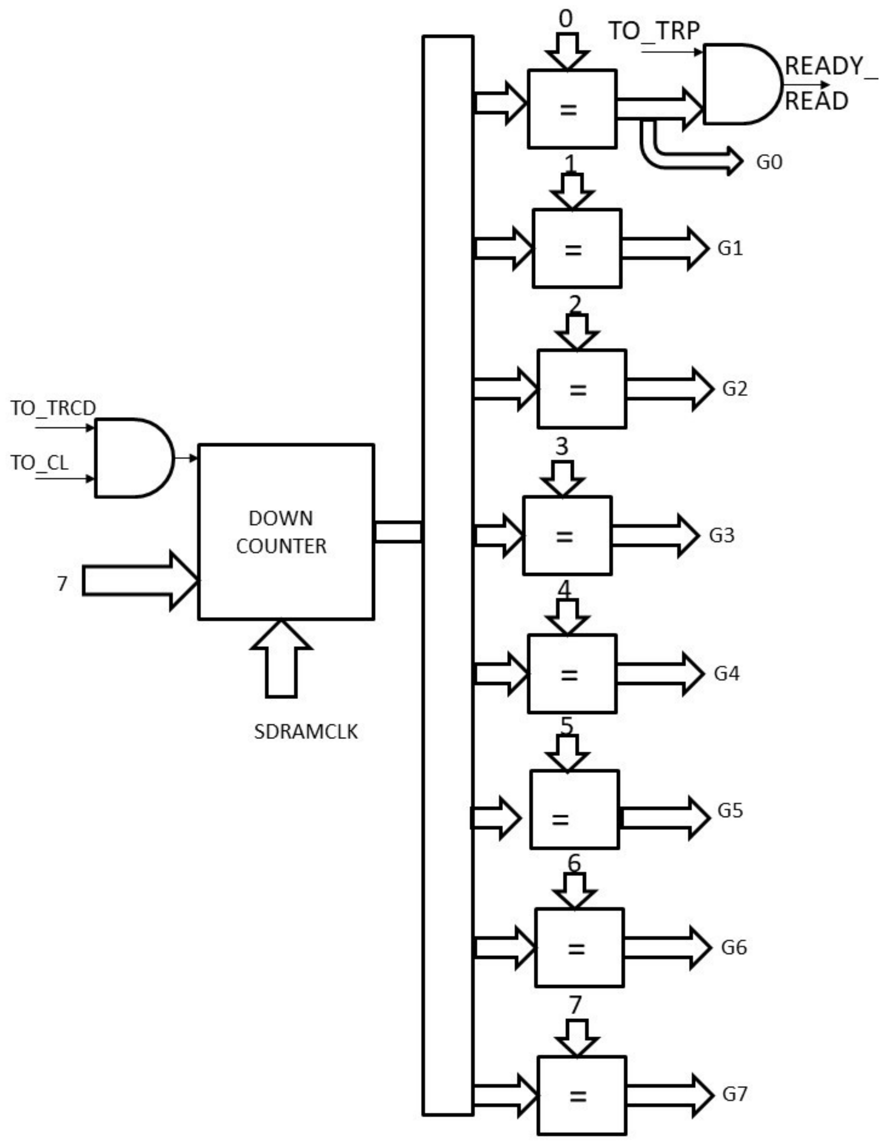
- **FSM for READ**



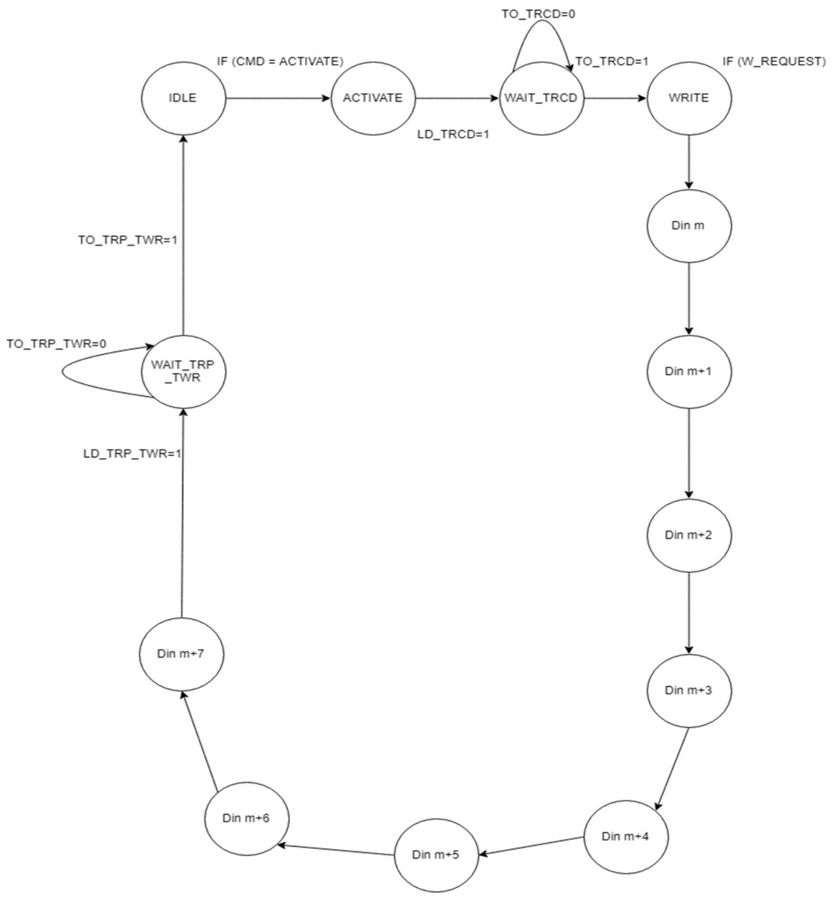
- **Data Read**



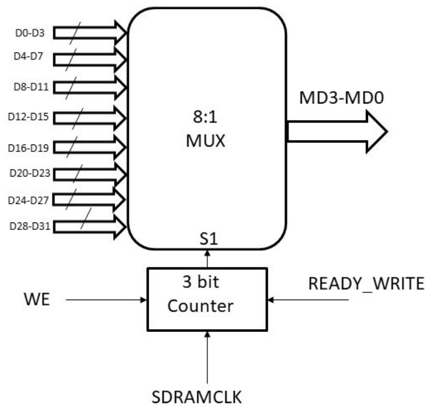
- **READY signal generation after READ:**



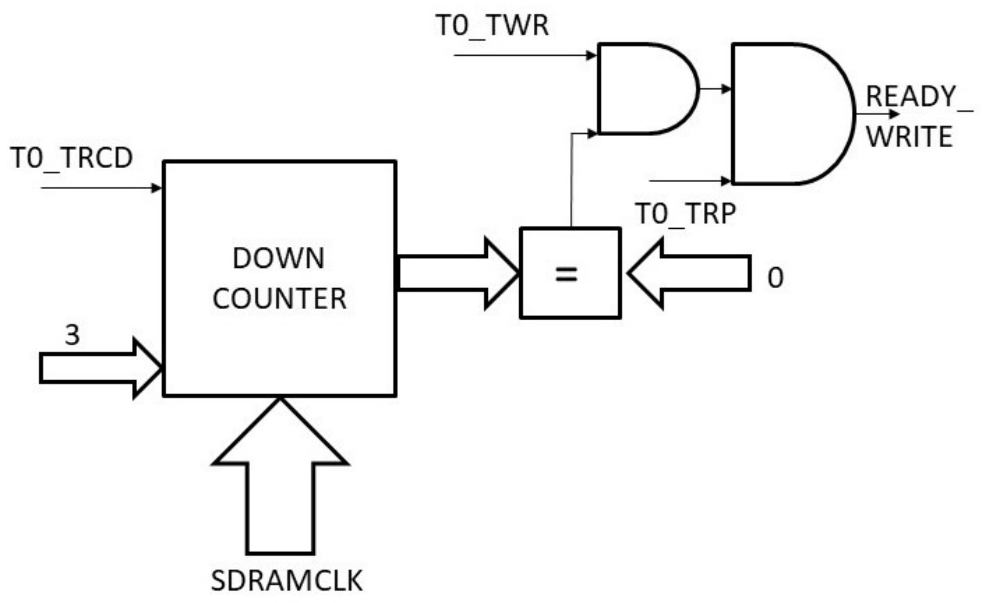
- **FSM for WRITE:**



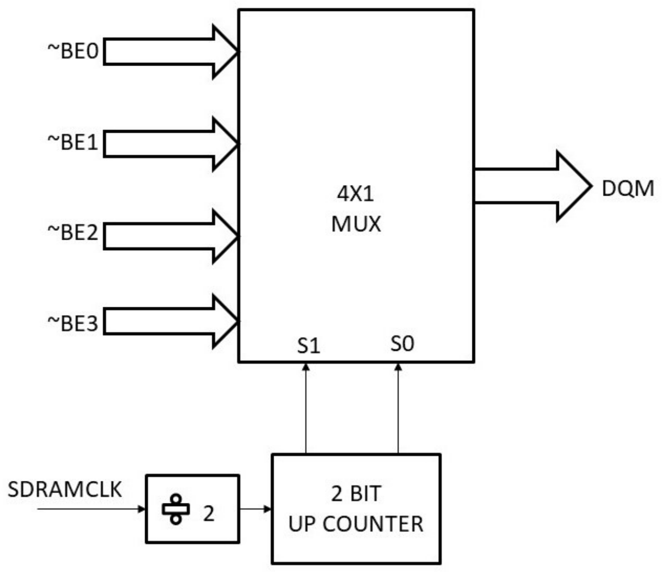
- **Data write:**



- **READY signal generation after WRITE:**

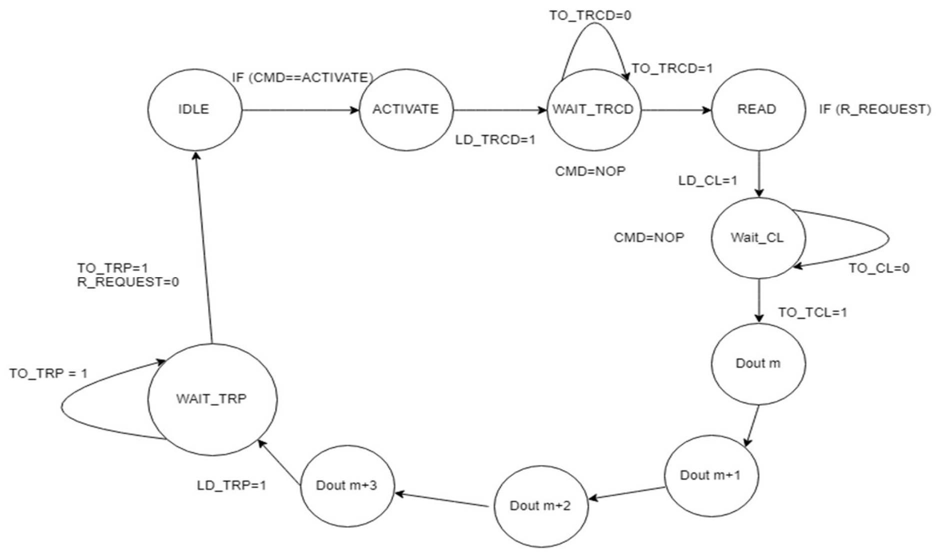


- **DQM Signal Generation:**

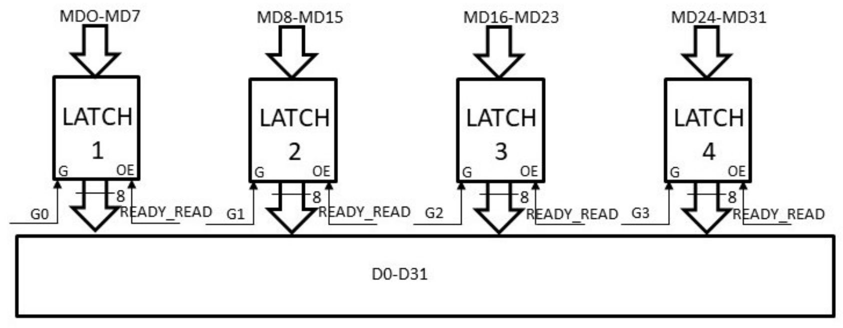


III. A)MT48LC8M8A2 (BL = 4)

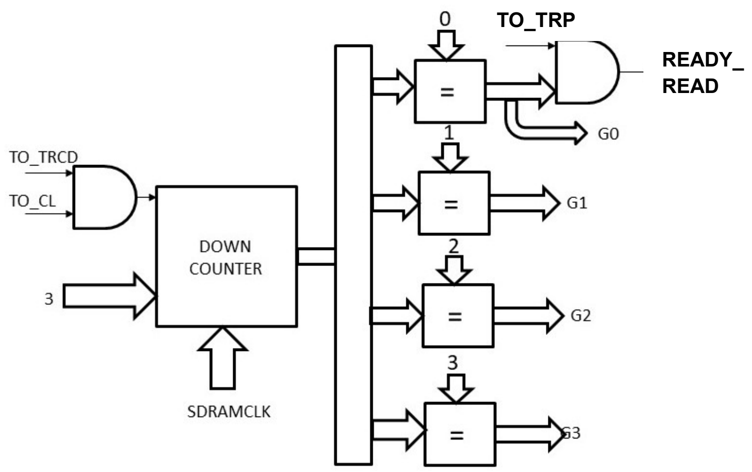
- **FSM for READD**



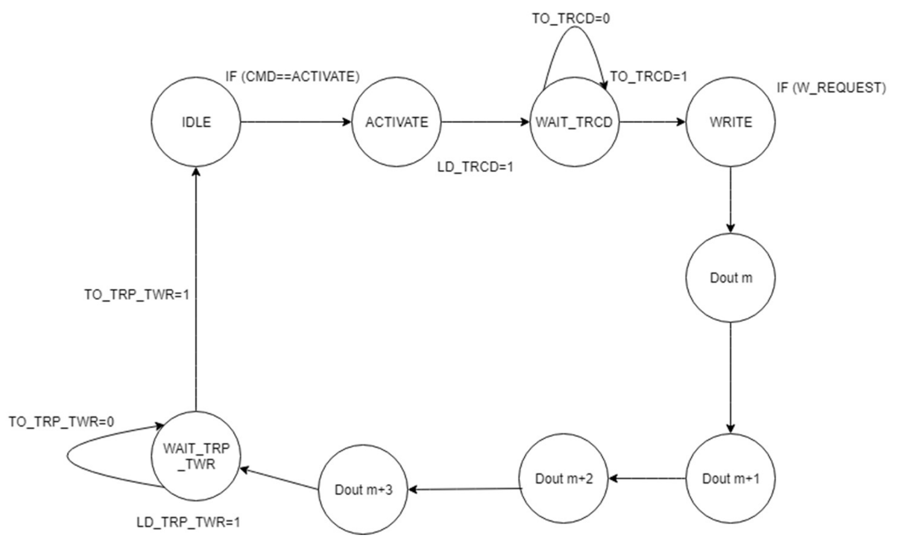
- **Data Read**



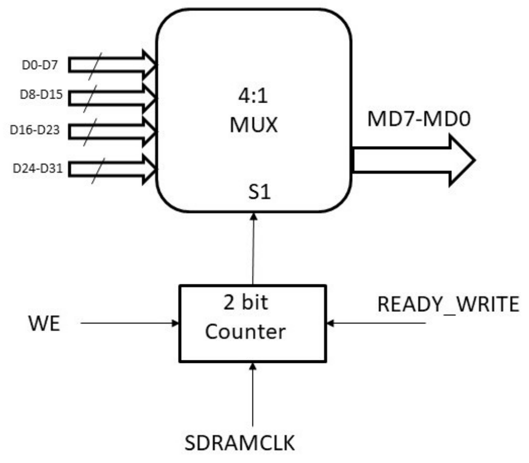
- **READY signal generation after READ:**



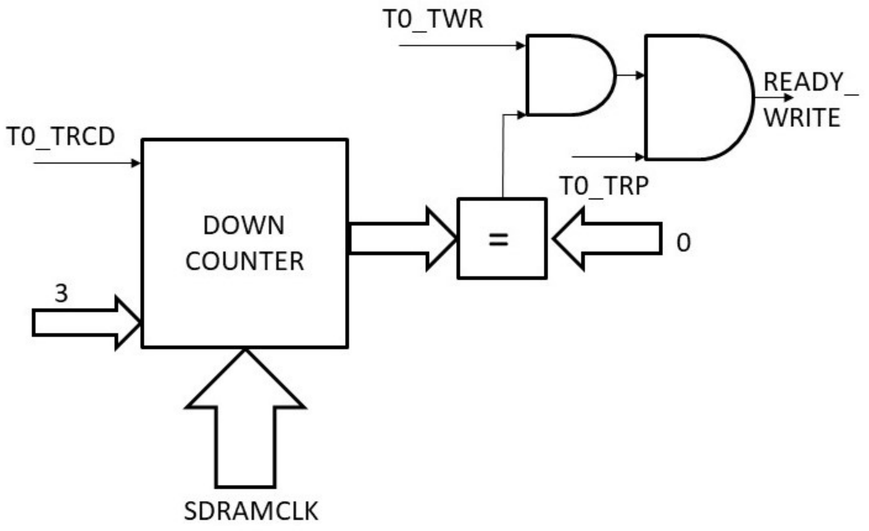
- **FSM for WRITE:**



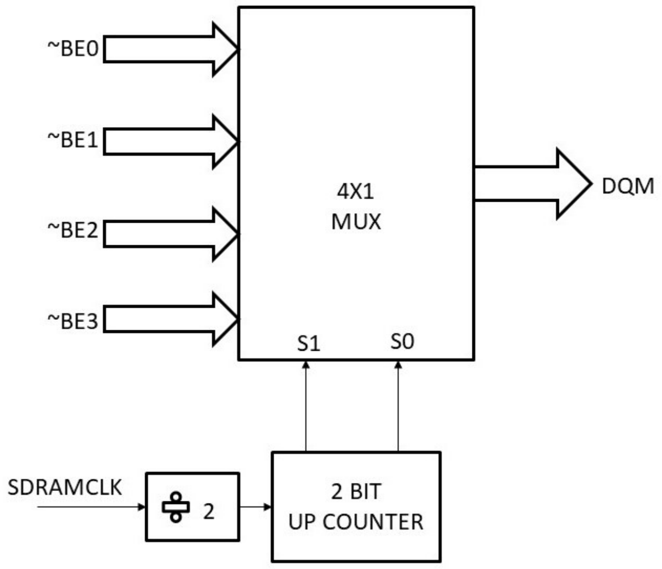
- **Data write:**



- **READY signal generation after WRITE:**

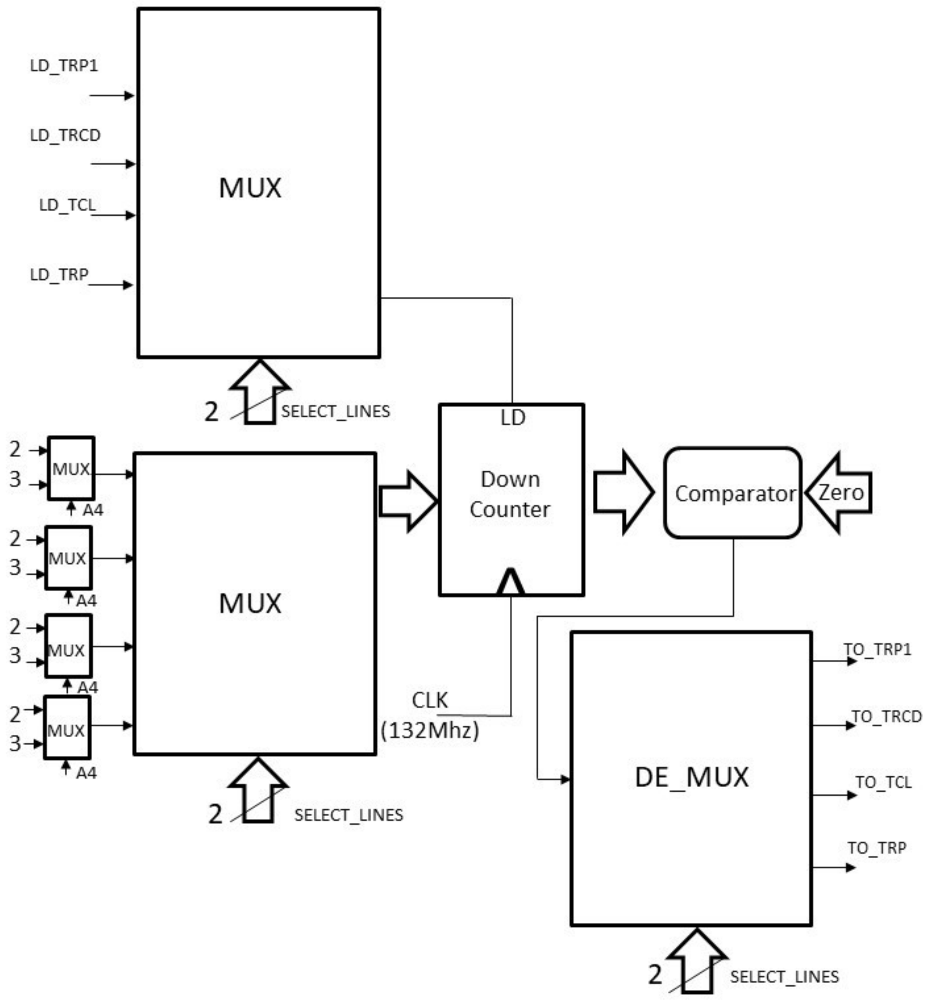


- **DQM Signal Generation:**



IV. PROGRAMMABLE CAS LATENCY:

Below mentioned logical diagram is used to load timing values based on CAS Latency, i.e. A4 bit of LMR register. These are the only timing values which vary with varying CL.



LD_TRP1
LD_TRCD
LD_TCL
LD_TRP
MUX
2
SELECT_LINES
2
3
MUX
A4
MUX
2
SELECT_LINES
LD
Down Counter
CLK (132Mhz)
Comparator
Zero
DE_MUX
TO_TRP1
TO_TRCD
TO_TCL
TO_TRP
2
SELECT_LINES