

Introduction:-

a) memory system:- memories are made up of Registers. Each Register in the memory is one storage location. Each location is identified by an address.
→ Each Register consists of storage element (Flip-flops or capacitors in semiconductor memories and magnetic domain in magnetic storage), each of which stores one bit of data. A storage element is called a cell.
→ The data stored in a memory by a process called writing and are retrieved from the memory by a process called reading.

→ memory can be classified as 'main memory' and 'Auxiliary memory' (secondary memory).

main memory:- The main memory is also known as the primary memory or core memory.

→ The instruction to be executed are read by the CPU from this memory.

→ The data on which the instruction has to be operate is also present only in this memory. → The output results and temporary results are stored in this memory.

→ All the processing is done through main memory, it should be Fast. Cost of this memory is also high.

Auxiliary memory:- The Auxiliary memory also known as secondary memory is connected as a peripheral device to the computer.

→ The information is stored in this in the form of magnetic energy.

→ The speed of data transfer in this memory is lesser compared to the main memory. It has the advantage of storing large volume of data.

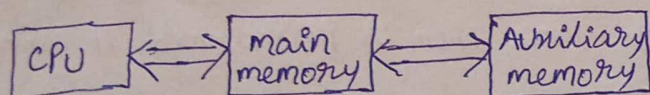


fig: memory organisation.

→ Ex: Hard disks, floppy disks.

→ In older days magnetic tape is used for memory for data storing. Now a days semiconductor memories are more popular and widely used.

→ semiconductor memories are small in size, less in weight, low cost, high speed, high reliability,

write operation:- The process of storing information into a memory is known as "write operation".

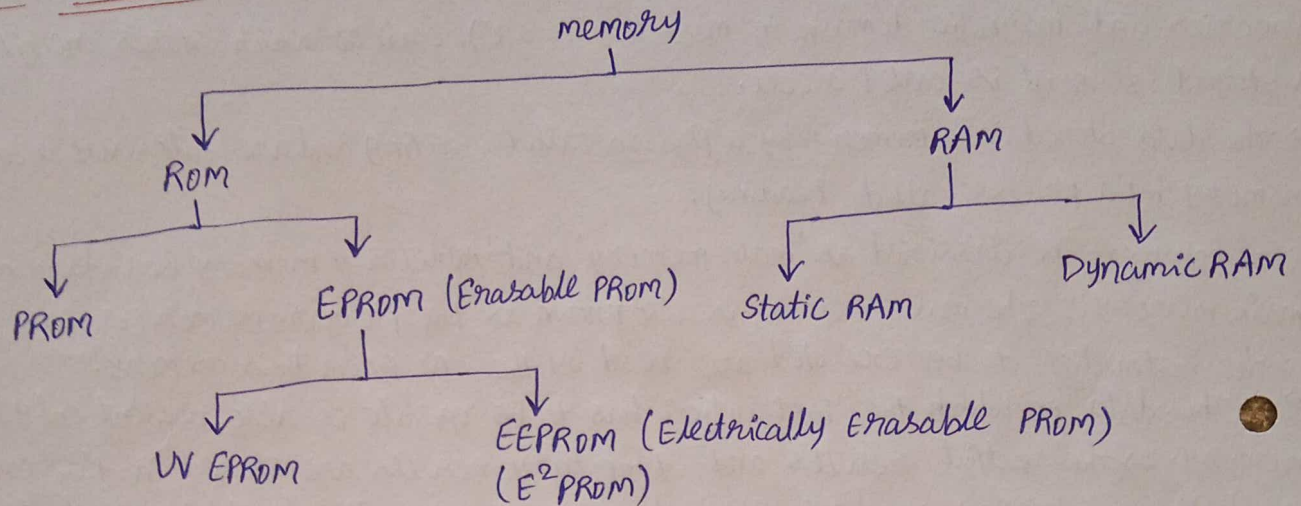
Read operation:- The process of retrieving the information from the memory is known as 'read operation'.

memory capacity:- the total number of bytes that can be stored in a memory is known as memory capacity.

1 kilo byte = 2^{10} bytes, 1 mega byte = 2^{20} bytes

1 Giga byte = 2^{30} bytes, 1 Tera byte = 2^{40} bytes.

1) memory structure :-



Rom:- Read only memory (Rom) are non-volatile devices because data stored in them is not lost when power is off.

→ Data permanently stored in the memory cells of an Rom.

Non-volatile memory:- The memory in which the stored data is not lost when the power is removed is known as non-volatile memory. Ex: Rom.

Prom:- programmable Read only memory (PROM) allows the user to store data by the process of programming. → once PROM is programmed, it cannot be changed. → It is a non-volatile memory.

Random Access memory (RAM):- It is also known as Read-write memory.

→ The data can be read and also write in RAM.

→ RAM is a volatile memory, Loss the stored data when power is off.

→ There are two types of RAM's. They are Static RAM, Dynamic RAM.

Rom

- 1) The data can only read but not written.
- 2) It is a non-volatile device
- 3) storage elements are diodes and transistors.
- 4) Data is permanently stored.

RAM

- 1) The data can read as well as write.
- 2) It is a volatile device
- 3) storage elements are transistors and capacitors.
- 4) Data is temporarily stored.

2) Static RAM :- memories that consists of circuits capable of Retaining their state as long as power is applied are known as static memories. (2)

→ These are random access memory (RAM) and combinedly called static RAM memories.

→ The static RAM's are built using MOS technology, but some are built using bipolar technology.

MOSFET RAM cell :-

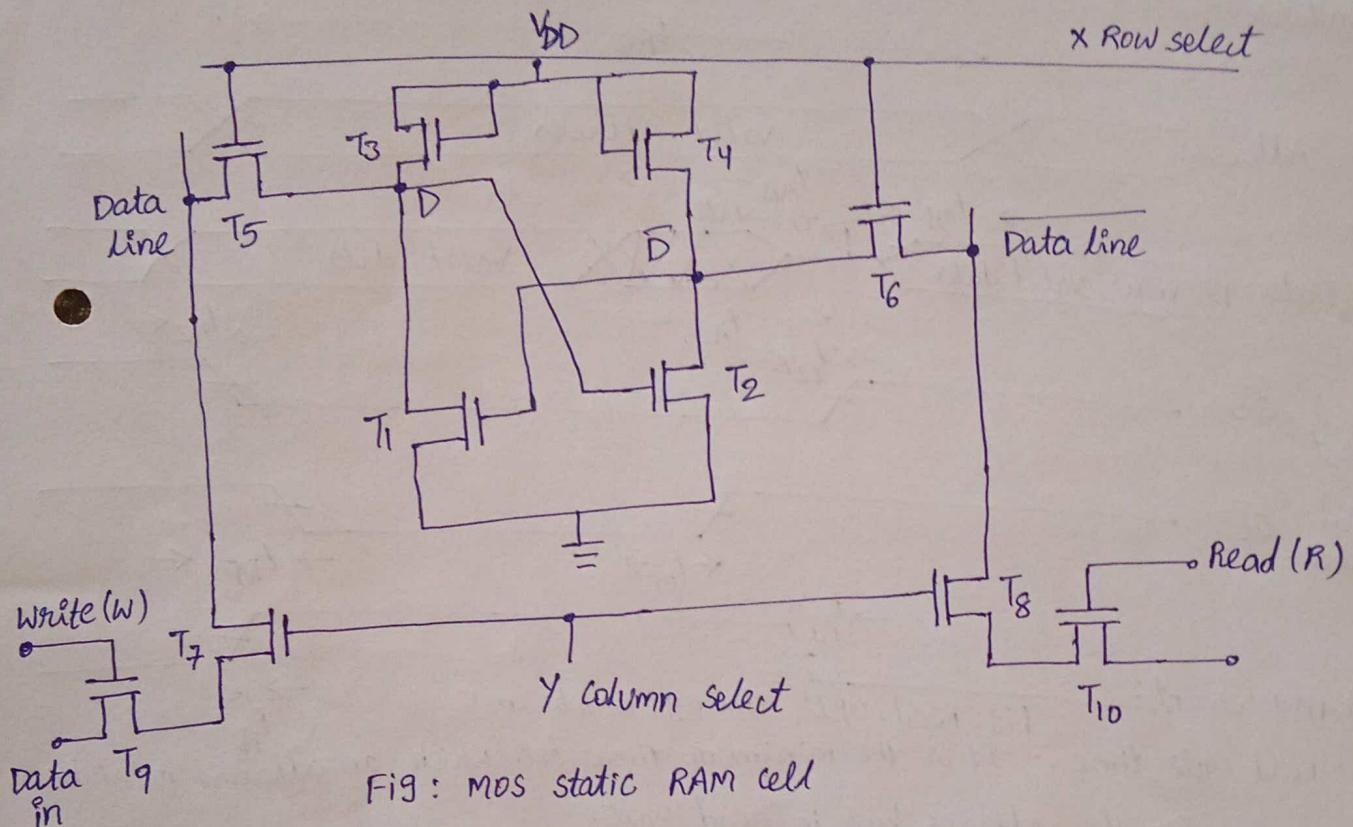


Fig: MOS static RAM cell

→ Enhancement mode MOSFET Transistors are used to make this RAM cell.

→ MOS Transistors T_1 and T_2 form the basic cross coupled inverters and T_3 and T_4 act as load Resistors for T_1 and T_2 .

→ X and Y lines are used for addressing the cell. When X and Y both are high, cell is selected.

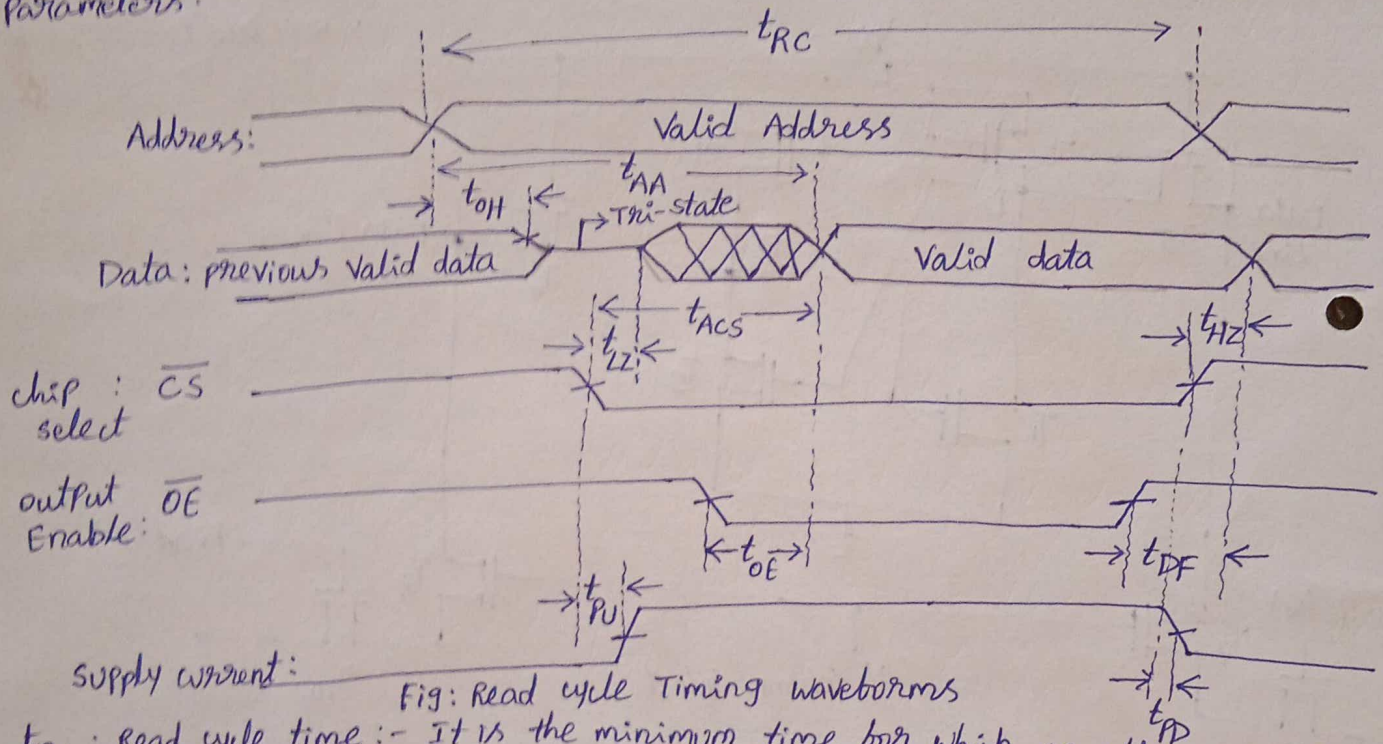
→ When $X=1$, T_5 and T_6 get ON and the cell is connected to the data and data line. → When $Y=1$, T_7 and T_8 are made ON. Due to this, Read or Write operation is possible.

Write operation :- Write operation can be enabled by making 'W' signal high. With write operation enabled, if data in signal is logic '1', node 'D' is also at logic '1'. → This turns ON T_2 and T_1 is cutoff. If new data on data in pin is Logic '0', T_2 will be cutoff and T_1 will be turned ON.

Read operation:- Read operation can be enabled by making 'R' signal high. With read operation enabled, T_{10} becomes ON. This connects the data line to the data out and thus the complement of the bit stored in the cell is available at the output.

SRAM Timing:-

i) Read cycle:- The timing diagram is drawn on the basis of different timing parameters.



t_{RC} : Read cycle time:- It is the minimum time for which an address must be held stable on the address bus, in read cycle.

t_{AA} : Address Access time:- It is the minimum specified time within which a valid new data is put on the data bus after an address is applied.

t_{OH} : output Hold from Address change:- It is the time for which a data from previously read address will be present on the output after new address is applied.

t_{LZ} : chip selection to output in tri-state:- It is the minimum time for which the data output will remain in tri-state after $\overline{CS}$ goes low.

t_{ACS} : chip select Access time:- It is the minimum time required for memory device to put valid data on the data bus after $\overline{CS}$ goes low.

t_{HZ} : chip deselection to output in tri-state:- It is the time within which

output returns to its high impedance state (tri-state) after $\overline{CS}$ goes high. (3)

t_{OE} : output enable to output delay:- It is the minimum time required for the memory device to put valid data on the data bus after $\overline{OE}$ goes low.

t_{DF} : output enable high to output float:- It is the time for which data will remain valid on the data bus after $\overline{OE}$ goes high.

t_{PU} : chip selection to power up time:- It is the time within which the memory device is powered up to its normal operating current after $\overline{CS}$ goes low.

t_{PD} : chip deselection to power down time:- It is the time for which the memory device will remain powered up after $\overline{CS}$ goes high.

ii) write cycle:- The write cycle for static RAM - timing diagram is drawn on the basis of different timing parameters.

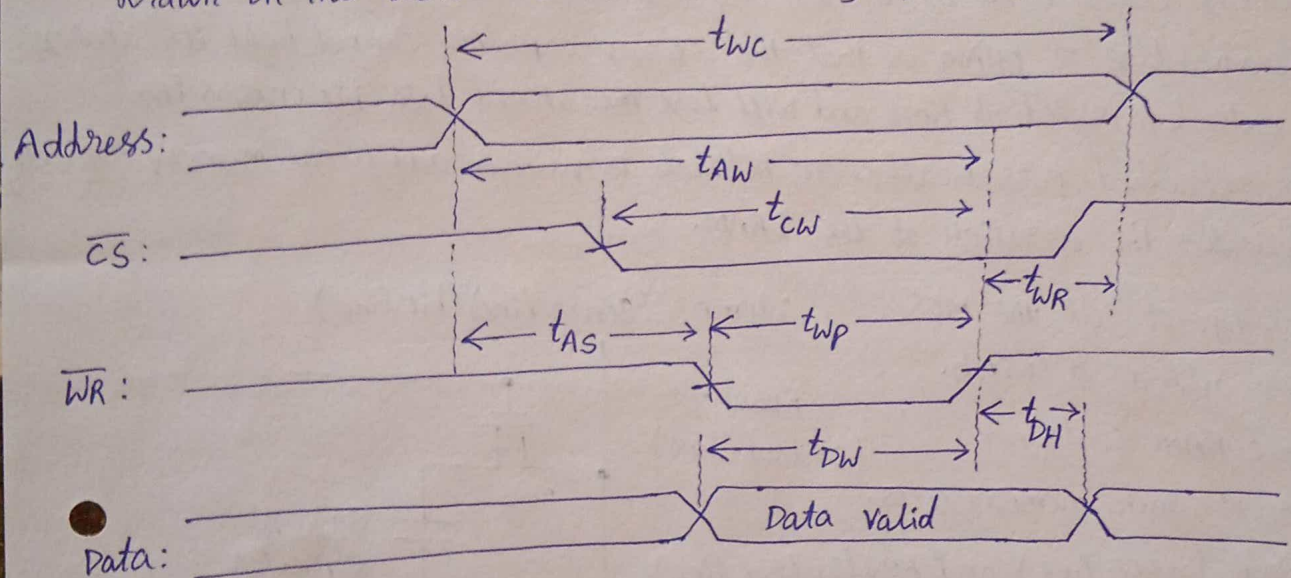


Fig: write cycle timing waveforms.

t_{WC} : write cycle time:- It is the minimum time for which an address must be held stable on the address bus, in write cycle.

t_{AW} : Address valid to end of write:- It is the time at which address must be applied on the address bus before $\overline{WR}$ goes high.

t_{WR} : write Recovery Time:- It is the time for which address will remain on address bus after $\overline{WR}$ goes high.

t_{AS} : Address setup time:- when address is applied, it is the time after which $\overline{WR}$ can be made low.

t_{CW} : chip selection to the end of write:- It is the time at which the $\overline{CE}$ must be made low to select the device before $\overline{WR}$ goes high.

t_{WP} : write pulse width:- It is the time for which $\overline{WR}$ goes low.

t_{DW} : Data valid to the end of write:- It is the minimum time for which data must be valid on the data bus before $\overline{WR}$ goes high.

t_{DH} : Data hold time:- It is the time for which data must be held valid after $\overline{WR}$ goes high.

3) Dynamic RAM (DRAM):-

→ Dynamic RAM stores the data as a charge on the capacitor.

→ The advantage of this type of cell is that it is very simple, thus allowing very large memory arrays to be constructed on a chip at a lower cost per bit.

→ The disadvantage of DRAM is that the storage capacitor cannot hold its charge over an extended period of time and will lose the stored data bit unless its charge is refreshed periodically. To refresh requires additional memory circuitry and complicates the operation of the DRAM.

→ In this type of cell, the MOS transistor acts as a switch.

→ A dynamic RAM contains thousands of such memory cells.

When column (Sense line) and Row (Control line)

go high, the MOSFET conducts and charges the capacitor.

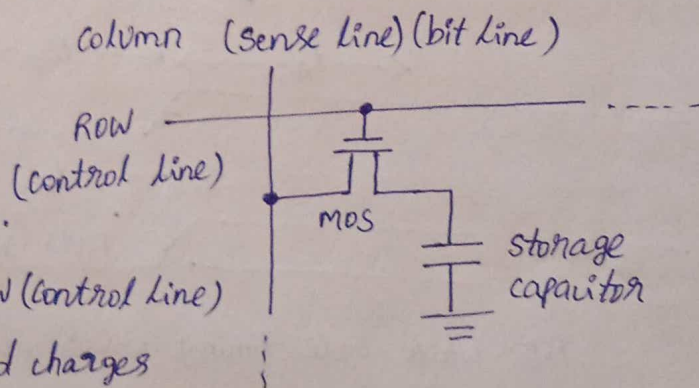


FIG: A MOS DRAM cell

→ When the column and Row lines go low, the MOSFET opens and the capacitor retains its charge. In this way, it stores 1 bit.

→ The dynamic RAM contains more memory cells as compared to static RAM per unit area.

→ Internal dynamic RAM circuit consists of three tri-state buffers: input buffer, output buffer and refresh buffer. Input & output buffers are enabled and disabled by controlling $R/\overline{W}$ line. When $R/\overline{W}$ line is low, input buffer is enabled and output buffer is disabled.

→ When $R/\bar{W}$ line is high, input buffer is disabled and output buffer is enabled. This operation is used to perform Read, Write and Refresh operations. (4)

Write operation:-

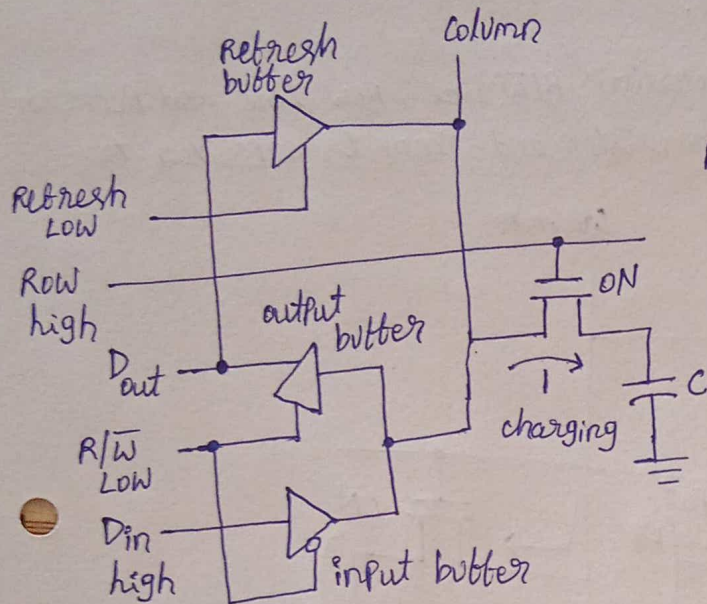


Fig: Writing a '1' into the memory cell

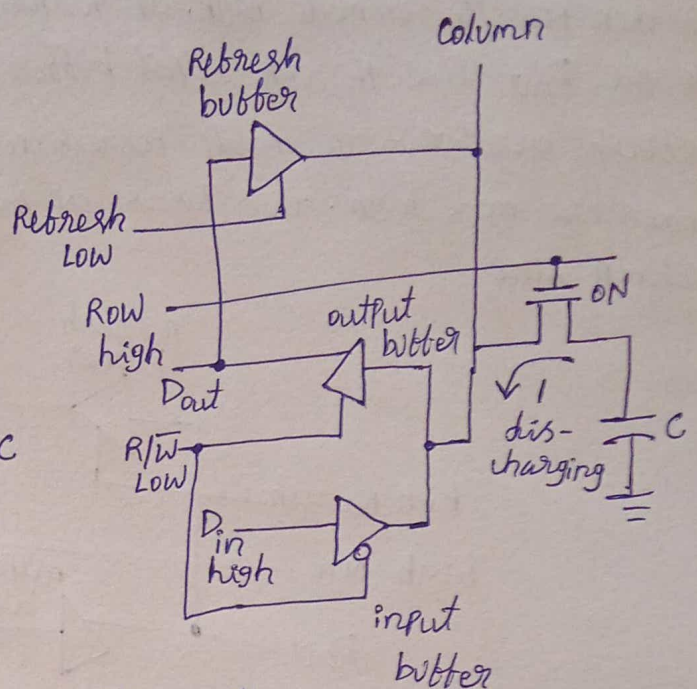


Fig: Writing a '0' into the memory cell

→ To enable write operations $R/\bar{W}$ line is made low which enables input buffer and disables output buffers. To write a '1' into the cell, the D_{in} line is high and the transistor is turned ON by a high on the Row line. This allows the capacitor to charge to a positive voltage.

→ When '0' is to be stored, a Low is applied to the D_{in} line. The capacitor remains uncharged, or if it is storing a '1', it discharges.

→ When the Row line is made low, the transistor turns off and disconnects the capacitor from the data line, thus storing the charge (either 1 or 0) on the capacitor.

Read operation:-

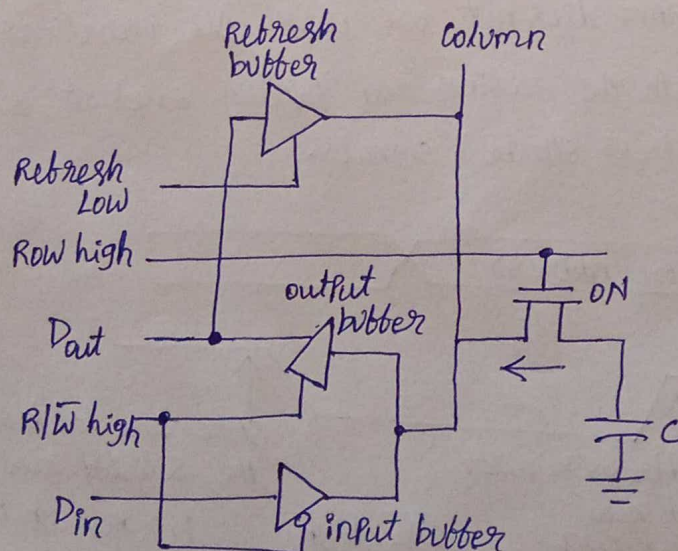


Fig: Reading a '1' from the memory cell.

→ To read data from the cell, the $R/\bar{W}$ line is made high, which enables output buffer and disables input buffer.

→ Then Row line is made high. It turns transistor ON and connects the capacitor to the Dout line through output buffer.

Refresh operation:- To enable Refresh operation $R/\bar{W}$ line, Row line and Refresh lines are made high. This turns ON transistor and connects capacitor to column line.

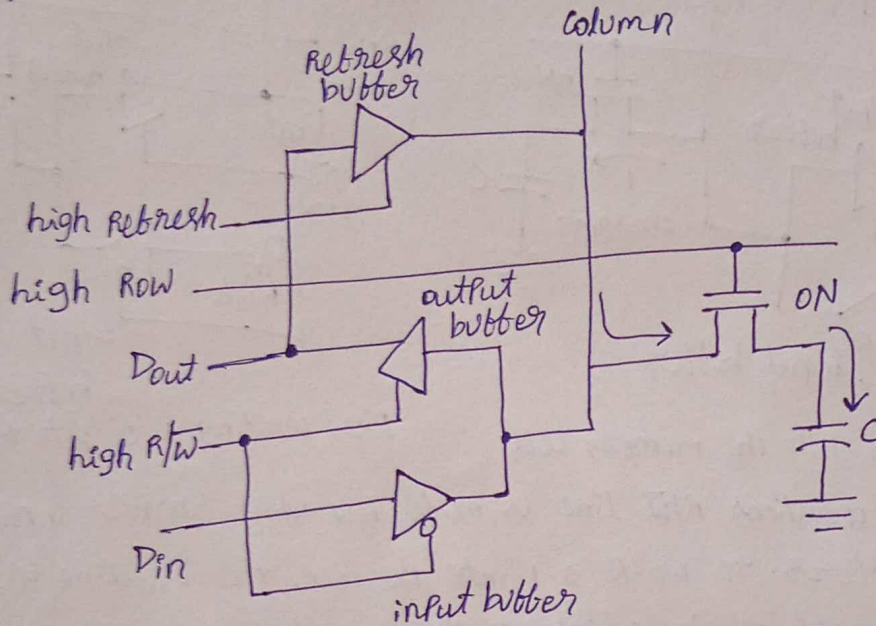
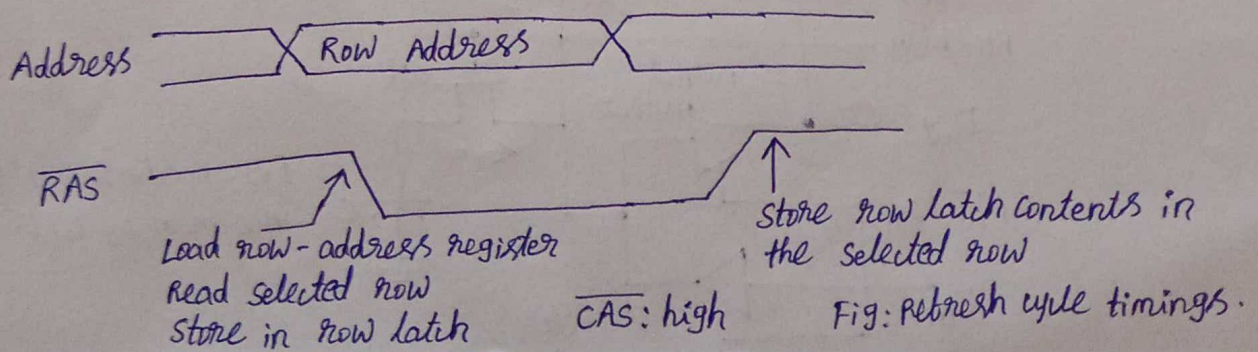


Fig: Refreshing a stored '1'.

→ As $R/\bar{W}$ is high, output buffer is enabled and the stored data bit is applied to the input of refresh buffer. → the enabled Refresh buffer then produces a voltage on column line corresponding to the stored bit and thus Refreshing the capacitor.

Dynamic RAM Timing:-

Refresh cycle:- Dynamic RAM does not use clock. All operations of DRAM are initiated and completed on both the rising and falling edges of $\overline{RAS}$ (Row Address strobe) and $\overline{CAS}$ (Column address strobe) signals.



→ This refresh cycle is used to refresh a row of DRAM without actually reading or writing any data at the external pins of DRAM chip. (5)

→ At the beginning of cycle a row address is applied to the multiplexed address inputs and $\overline{RAS}$ signal is activated. At the falling edge of the $\overline{RAS}$ signal DRAM stores the row address in an internal row address register and reads the selected row of the memory array into an on chip row latch.

→ When the $\overline{RAS}$ signal is deactivated, the contents of the row are written back from the row latch. To refresh an entire 64K x 1 DRAM, 256 such cycles are needed. The process of refreshing has to be repeated after every 4ms.

Read cycle:- Read cycle begins like refresh cycle and selected row contents are loaded into the row latch. Then column address is applied to the multiplexed address inputs and is stored in an on-chip column address register on the falling edge of the $\overline{CAS}$ signal.

→ The column address is used to select one bit of the just read row and the bit is made available on the D_{out} pin of DRAM.

→ A tri-state pin, D_{out} is output enabled as long as $\overline{CAS}$ is asserted. The entire row is written back into the array as soon as $\overline{RAS}$ is negated.

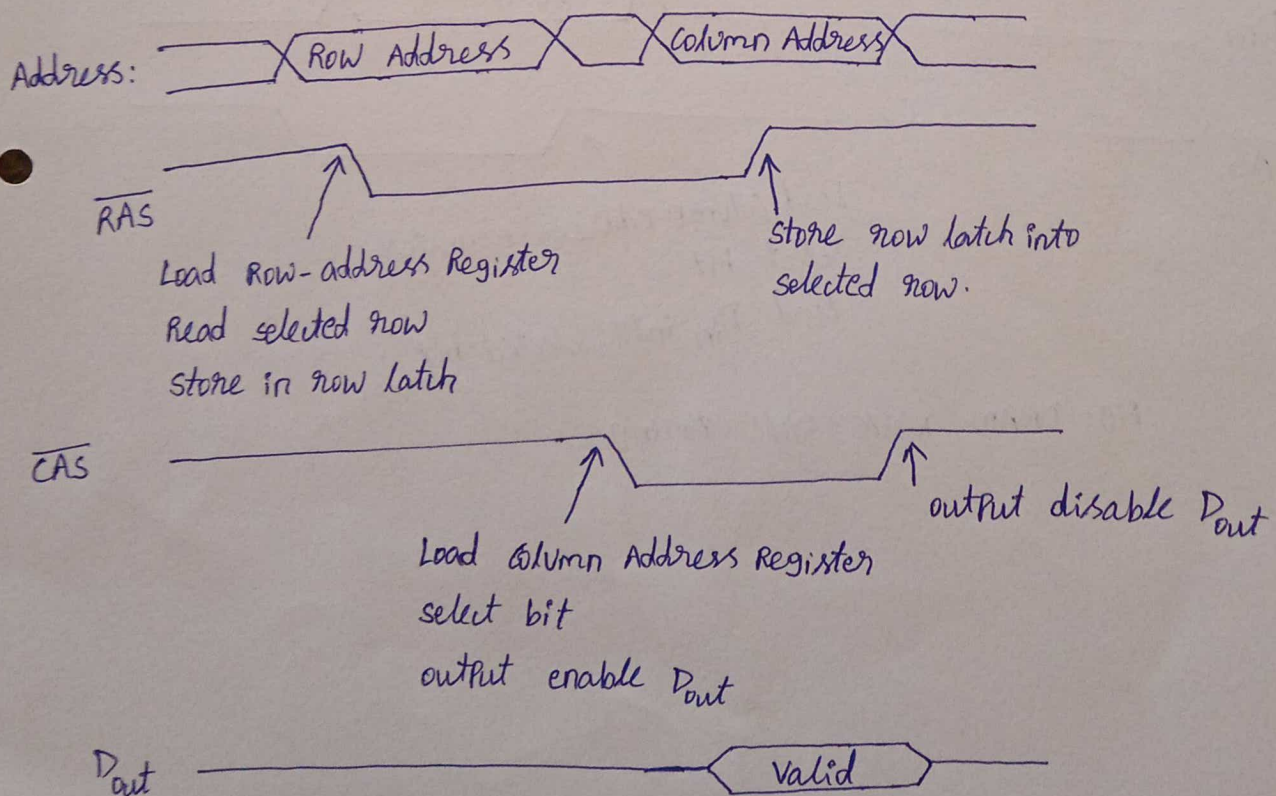


Fig: DRAM read-cycle timings.

Write cycle :- The write cycle begins like a refresh or read cycle. $\overline{WE}$ is asserted before $\overline{CAS}$ is asserted to select a write cycle.

→ Since $\overline{WE}$ is asserted, D_{out} is disabled and the bit from D_{in} input is written into the row latch at the bit position selected by column address.

→ At the rising edge of the $\overline{RAS}$ signal, the row latch contents are copied back and thus the newly written bit is also copied back.

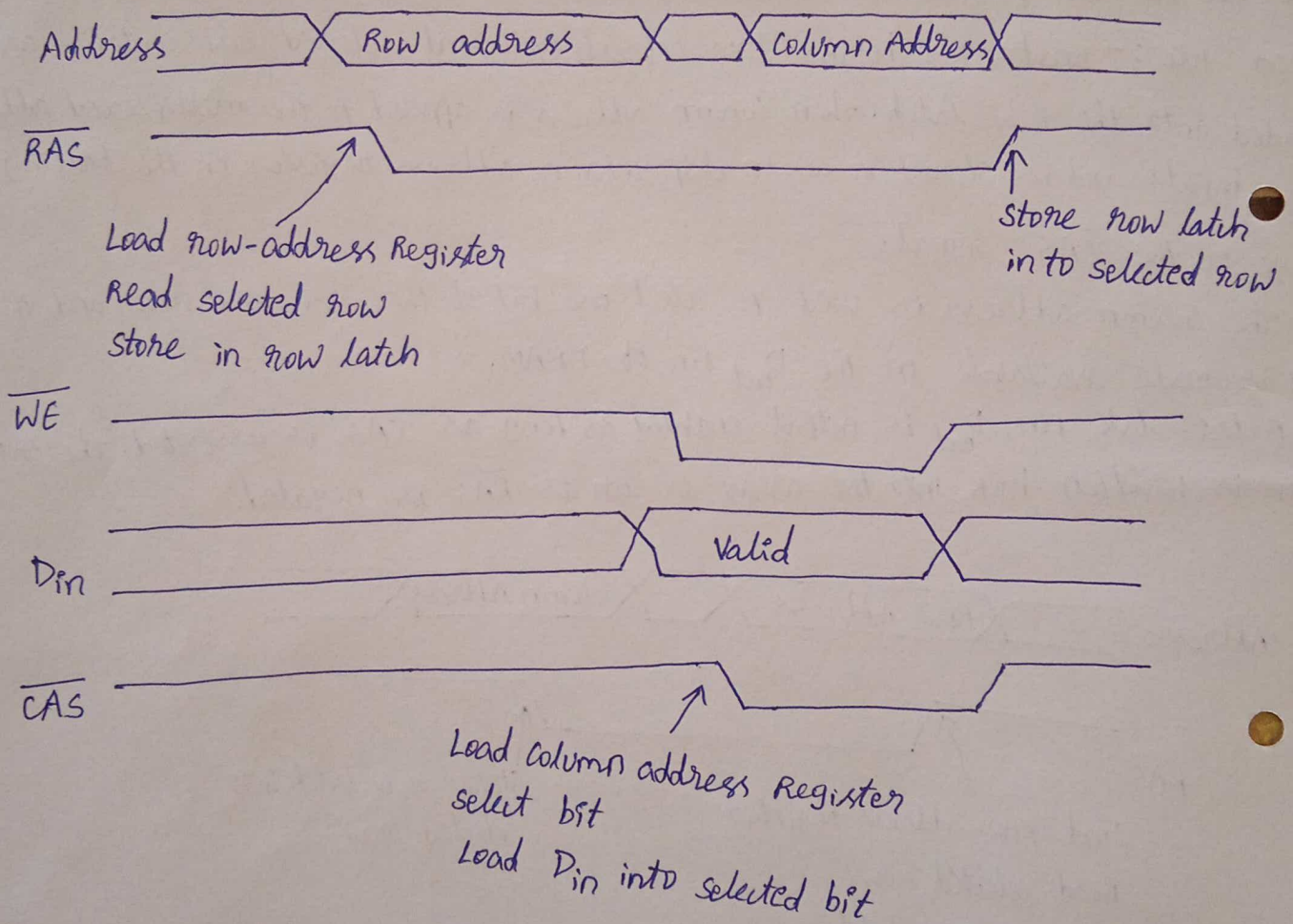


Fig: DRAM write-cycle timings.

Complex programmable Logic devices (CPLD's) Architecture:-

4) → complex programmable logic device (CPLD) is basically a single device containing multiple simple programmable logic devices (SPLD's) and providing more capacity for larger logic designs.

→ A CPLD consists multiple SPLD arrays with programmable interconnections. The CPLD's are internally organized varies with the manufacturer.

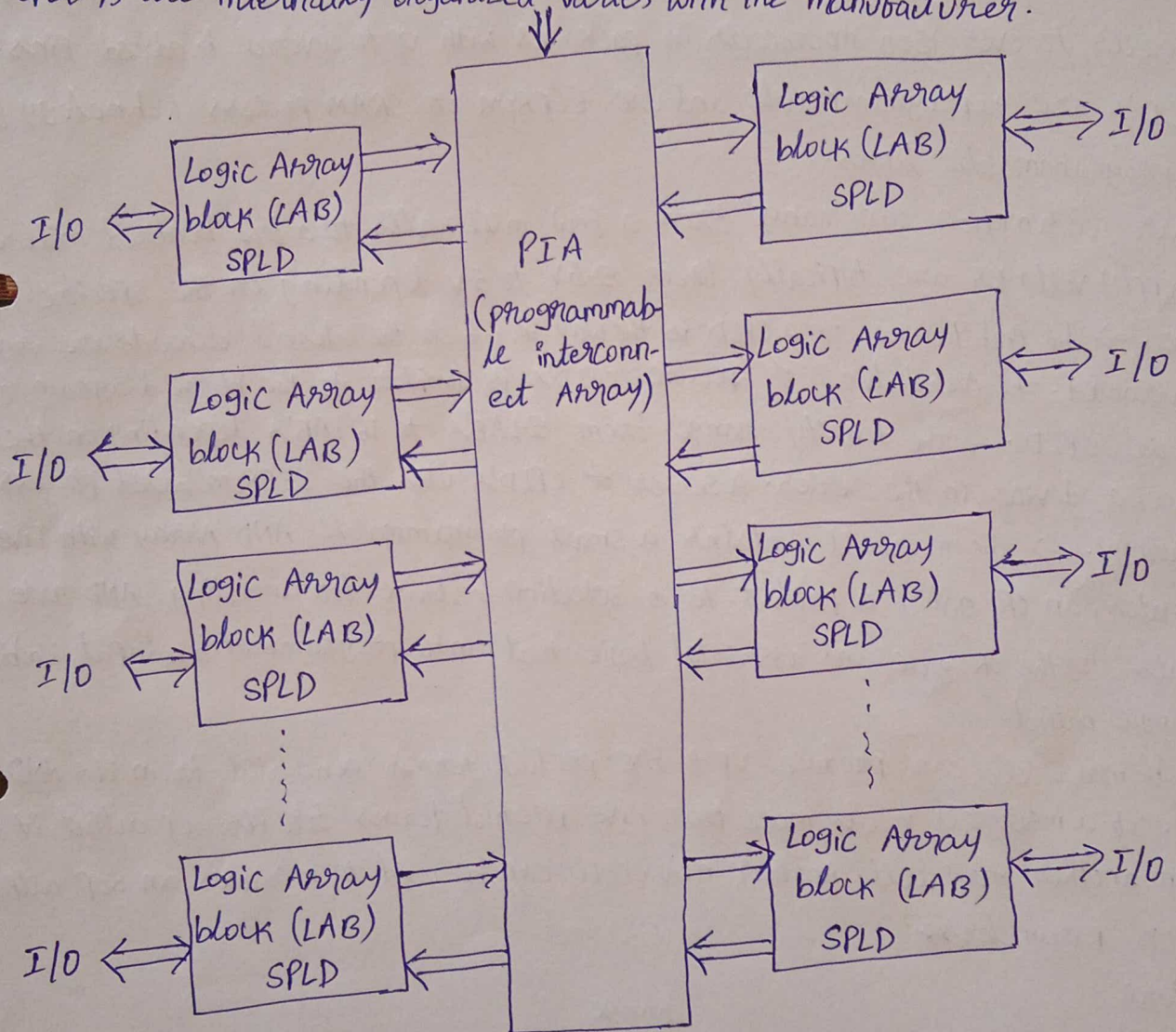


Fig: Basic block diagram of a generic CPLD

→ Each SPLD Array in a CPLD is a LAB (Logic Array block). Programmable interconnections are generally called the PIA (programmable interconnect Array) although some manufacturers, such as xilinx, use the term AIM (Advanced interconnect array).
(matrix)

→ the LAB's and the interconnections between LAB's are programmed using software. A CPLD can be programmed for complex logic functions based on the SOP structure of the individual LAB's (actually SPLD's).

→ Inputs can be connected to any of the LAB's and their outputs can be interconnected to any other LAB's via the PIA.

→ programmable logic manufacturers CPLD density can range from tens of macrocells to over 1500 macrocells in packages with up to several hundred pins.

→ CPLD's are reprogrammable and use EEPROM or SRAM process technology for the programmable links.

→ power consumption can range from a few milliwatts to a few hundred milliwatts.

DC supply voltages are typically from 2.5V to 5V depending on the specific device.

→ The Architecture of a CPLD is the way in which the internal elements are organized and arranged. The Architecture of specific CPLD's is similar to the block diagram of a generic CPLD.

→ The density ranges from 2 LAB's to 16 LAB's depending on the particular device in the series. A series of CPLD's uses the EEPROM based process technology.

→ The "macrocell" contains a small programmable AND array with five AND gates, an OR gate, a product-term selection matrix for connecting AND gate outputs to the OR gate and associated logic that can be programmed for input, combinational logic output.

→ Each macrocell can produce up to five product terms generated from its AND array. If a macrocell needs more than five product terms for its SOP output, it can use another macrocell. Another macrocell can be used to increase an SOP output.

→ CPLD Parameters:-

<u>Feature</u>	<u>Range</u>
1) Number of macrocells	10 - 1700
2) Number of LAB's	10 - 221
3) maximum operating Frequency	20.4 MHz - 400 MHz
4) Number of I/O's	10 - 1156
5) DC operating Voltage	1.8V, 2.5V, 3.3V, 5V.

→ CPLD manufacturers: Altera, xilinx, Lattice, Atmel

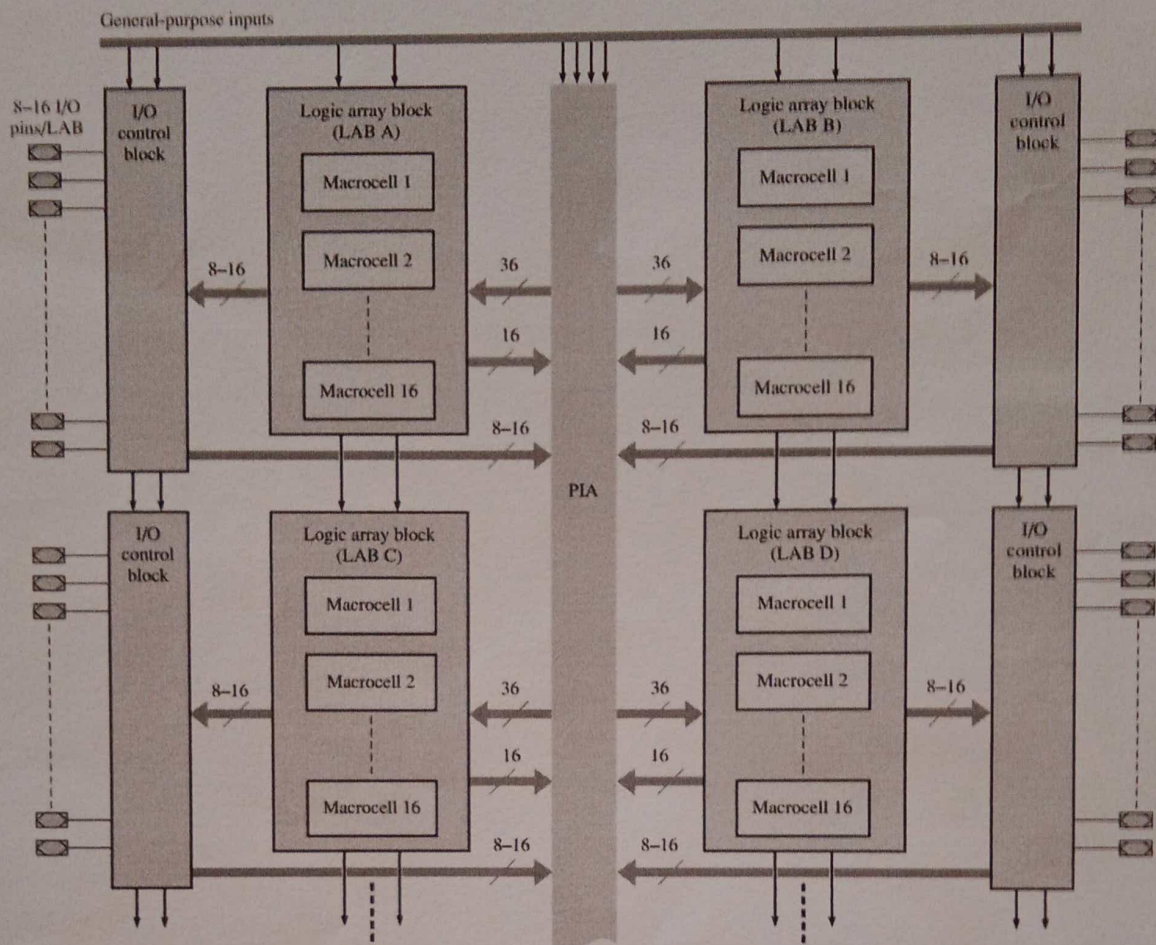


FIGURE 10-9 Basic block diagram of a typical CPLD.

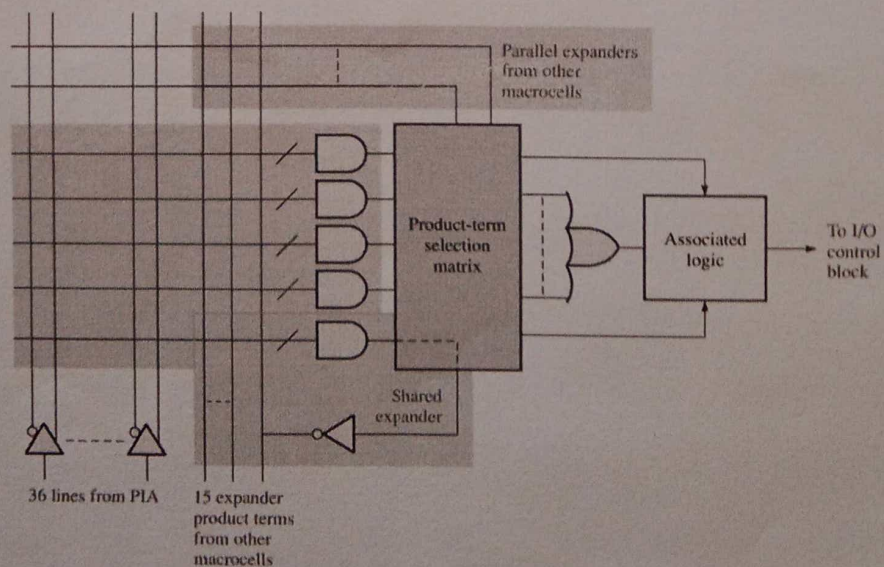


FIGURE 10-10 Simplified diagram of a macrocell in a typical CPLD.

2) Architecture of Field-programmable Gate Arrays (FPGA's) :-

- The three basic elements in an FPGA are the configurable logic block (CLB), the interconnections and the input/output blocks.
- The configurable logic blocks in an FPGA are many more of them.
- When the CLB's are relatively simple, the FPGA Architecture is called "Fine grained". When the CLB's are larger and more complex, the Architecture is called coarse grained.
- The distributed matrix of programmable interconnections provide both interconnections of the CLB's and connections to inputs and outputs. Large FPGA's can have tens of thousands of CLB's in addition to memory.
- FPGA's are reprogrammable and use SRAM or antifuse process technology for the programmable links.
- Densities can range from hundreds of logic modules to hundreds of thousands of logic modules in packages with up to over 1,000 pins.
- DC supply voltages are typically 1.8V to 5V depending on the specific device.
- An FPGA logic block consists of several smaller logic modules that are the basic building units. The configurable logic blocks (CLB's) within the global row/column programmable interconnects that are used to connect logic blocks.
- Each CLB's is made up of multiple smaller logic modules and a local programmable interconnect that is used to connect logic modules within the CLB.

i) Logic modules :- A logic module in an FPGA logic block can be configured for combinational logic, registered logic, or a combination of both.

- A Flip-Flop is part of the associated logic and is used for registered logic.
- Look-up table (LUT) based logic module is a type of memory that is programmable and used to generate SOP combinational logic functions. LUT consists of a number of memory cells equal to 2^n , where 'n' is the number of input variables.

ii) operating modes of a Logic module :- A logic module can be programmed for the following modes of operation: 1) Normal mode 2) Extended LUT mode etc

3) Arithmetic mode 4) Shared Arithmetic mode.

- 'Normal mode' is used primarily for generating combinational logic function. A logic module can implement one or two combinational output functions with its two LUT's.

→ In the normal mode, you are limited to 6-variable SOP Functions.

→ The 'Extended LUT mode' allows expansion to a 7-variable function, the multiplexer formed by the AND-OR circuit with a complemented input is part of the dedicated logic in a Logic module.

* → FPGAs are either nonvolatile because they are based on antifuse technology or they are volatile because they are based on SRAM technology.

→ SRAM based FPGAs include either a nonvolatile configuration memory embedded on the chip to store the program data and reconfigure the device each time power is turned back on or they use an external memory with data transfer controlled by a host processor.

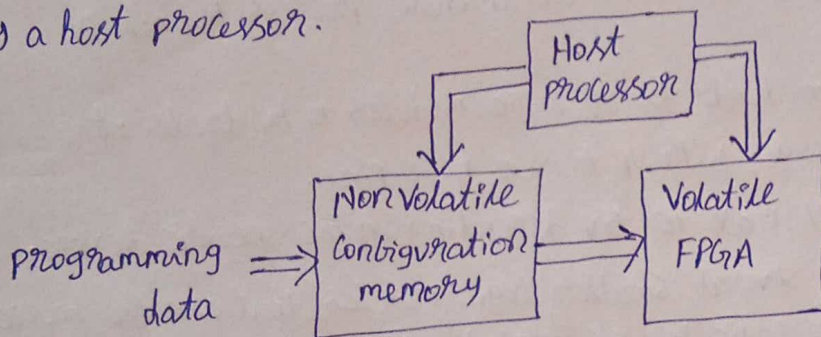


fig: Volatile FPGA with on-board memory and host processor.

→ FPGAs contain hard-core logic. A hard-core is a portion of logic in an FPGA that is put in by the manufacturer to provide a specific function and that cannot be reprogrammed.

→ If the embedded function has some programmable features, it is known as a soft-core function.

→ Hard cores are generally available for functions that are commonly used in digital systems, such as microprocessor, standard input/output interfaces and digital signal processors. More than one hard-core function can be programmed in an FPGA. The hard-core function is embedded in the user-programmed logic.

→ FPGAs containing either or both hard-core and soft core embedded processors and other functions are known as platform FPGAs because they can be used to implement an entire system without the need of external support devices.

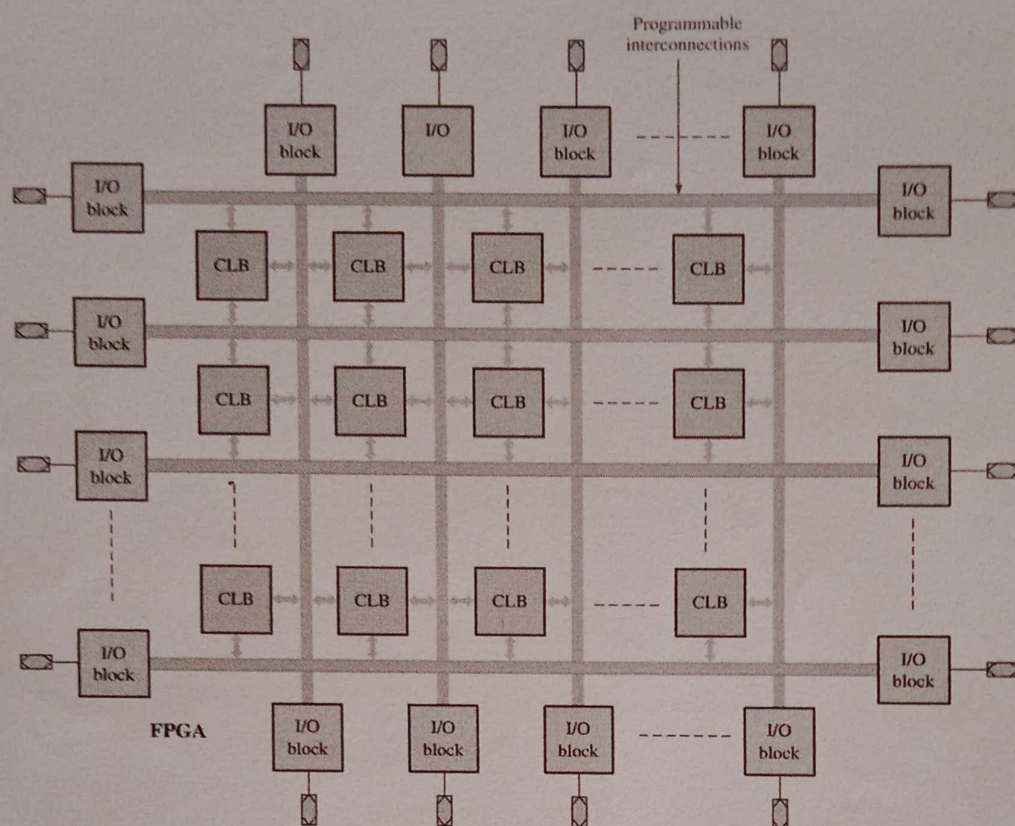


FIGURE 10-23 Basic structure of an FPGA. CLB is configurable logic block, also known as logic array block (LAB).

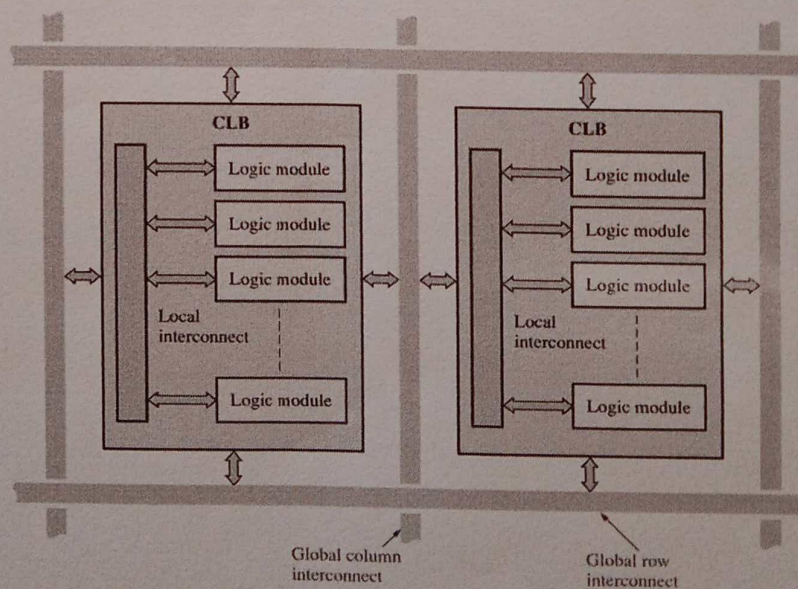


FIGURE 10-24 Basic configurable logic blocks (CLBs) within the global row/column programmable interconnects.

→ the FPGA contains embedded memory functions as well as digital signal processing (DSP) functions. DSP functions, such as digital filters are used in many systems.

→ the embedded blocks are arranged throughout the FPGA interconnection matrix and input/output elements (IOE's) are placed around the FPGA perimeter.

→ FPGA manufacturer: Altera, xilinx, Lattice, Atmel.

→ FPGA parameters: Embedded memory: 26 Kb - 63 Mb, Number of I/O's: 18-1200, DC operating Voltage: 1.8V, 2.5V, 3.3V, 5V.

6) Design and Analysis of Digital clock:- The digital counter is used to design the basic digital clock system. Digital clock that displays seconds, minutes and hours.

→ 1st a 60 Hz sinusoidal ac voltage is converted to a 60 Hz pulse waveform and divided down to a 1 Hz pulse waveform by a divide by 60 counter formed by a divide-by-10 counter followed by a divide-by-6 counter.

→ Both the seconds and minutes counts are also produced by divide-by-60 counters.

→ these counters count from 0 to 59 and then recycle to '0'; synchronous decade counters are used in this particular implementation.

→ Divide-by-6 portion is formed with a decade counter with a truncated sequence achieved by using the decoder count 6 to asynchronously clear the counter. The terminal count, 59 is also decoded to enable the next counter in the chain.

→ the hours counter is implemented with a decade counter and a Flip-Flop. Initially both the decade counter and the Flip-Flop are RESET, and the decade-12 gate and decade-9 gate outputs are High.

→ the decade counter Advances through all of its states from zero to nine, and on the clock pulse that recycles it from nine back to zero, the Flip-Flop goes to the set state ($J=1, K=0$). This illuminates a '1' on the tens-of-hours display. The total count is now ten (decade counter is in the zero state and the Flip-Flop is set).

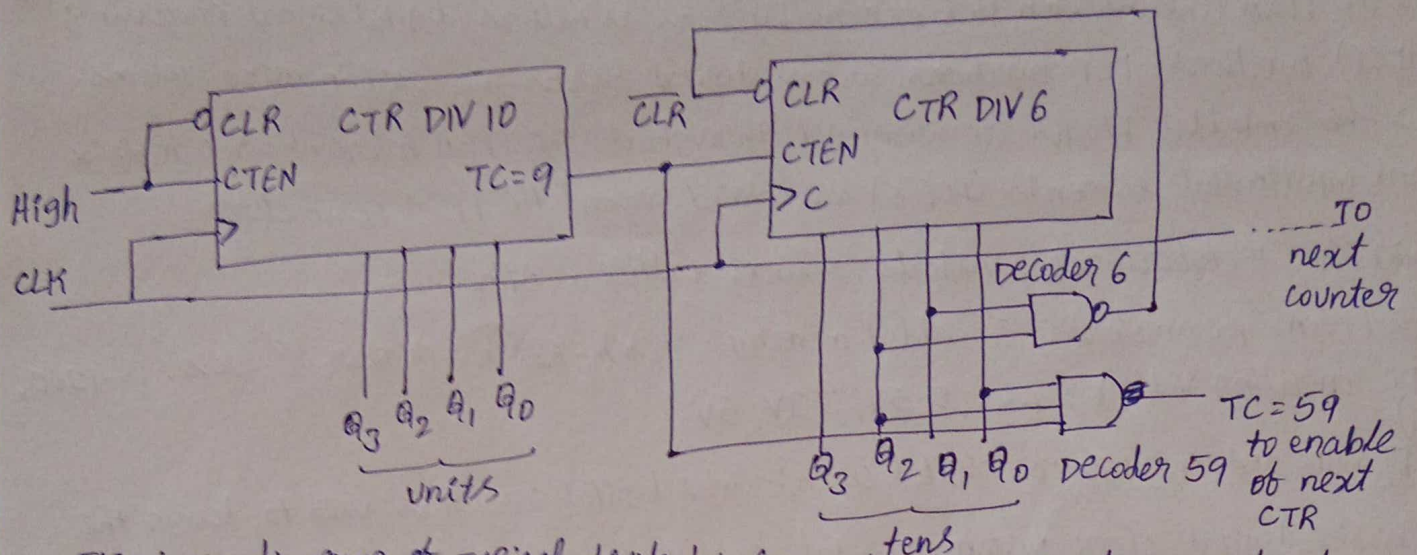


Fig: Logic diagram of typical divide-by-60 counter using synchronous decade counter.

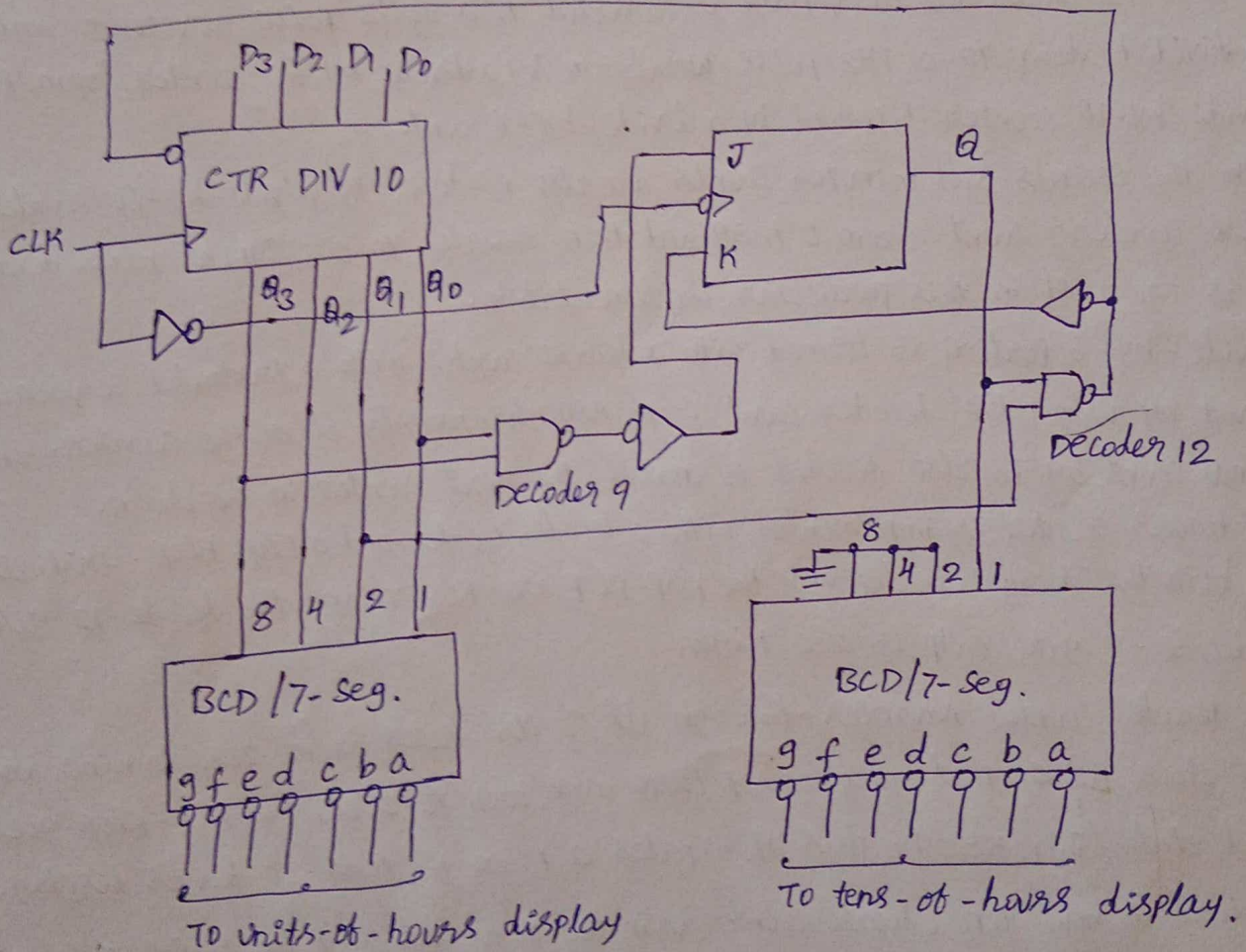


Fig: Logic diagram for hours counter and decoders.

→ The total count advances to eleven and then to twelve. In state 12 the Q_2 output of the decade counter is high, the Flip-Flop is still set, and thus the decode-12 gate output is low. → This activates the $\overline{LOAD}$ input of the decade counter. On the next clock pulse, the decade counter is preset to 0001 from the data inputs, and

- Explain how the hours counter is implemented
- Discuss the application of a counter in an automobile parking control system
- Describe how a counter is used in the process of parallel-to-serial data conversion

A Digital Clock

A common example of a counter application is in timekeeping systems. Figure 9-48 is a simplified logic diagram of a digital clock that displays seconds, minutes, and hours. First, a 60 Hz sinusoidal ac voltage is converted to a 60 Hz pulse waveform and divided down to a 1 Hz pulse waveform by a divide-by-60 counter formed by a divide-by-10 counter followed by a divide-by-6 counter. Both the *seconds* and *minutes* counts are also produced by divide-by-60 counters, the details of which are shown in Figure 9-49. These counters count from 0 to 59 and then recycle to 0; synchronous decade counters are used in this particular implementation. Notice that the divide-by-6 portion is formed with a decade counter with a truncated sequence achieved by using the decoder count 6 to asynchronously clear the counter. The terminal count, 59, is also decoded to enable the next counter in the chain.

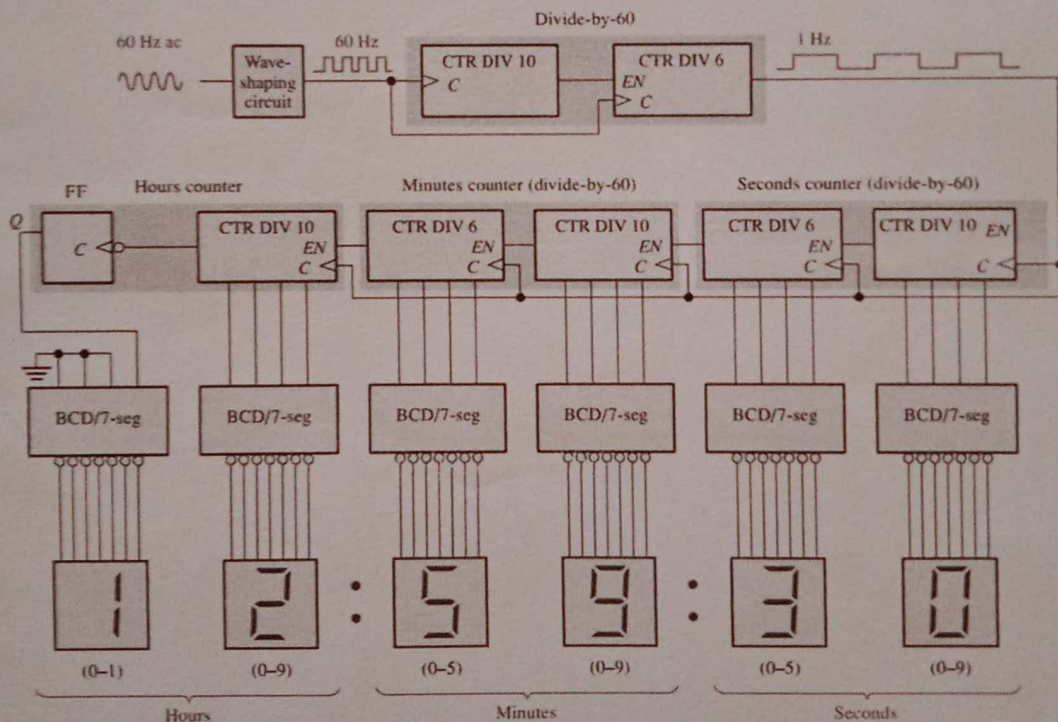


FIGURE 9-48 Simplified logic diagram for a 12-hour digital clock. Logic details using specific devices are shown in Figures 9-49 and 9-50.

The *hours* counter is implemented with a decade counter and a flip-flop as shown in Figure 9-50. Consider that initially both the decade counter and the flip-flop are RESET, and the decode-12 gate and decode-9 gate outputs are HIGH. The decade counter advances through all of its states from zero to nine, and on the clock pulse that recycles it from nine back to zero, the flip-flop goes to the SET state ($J = 1, K = 0$). This illuminates a 1 on the tens-of-hours display. The total count is now ten (the decade counter is in the zero state and the flip-flop is SET).

the Flip-Flop is reset ($J=0, K=1$). This logic always causes the counter to recycle from twelve back to one rather than back to zero. (2)

7) Traffic Light controller :-

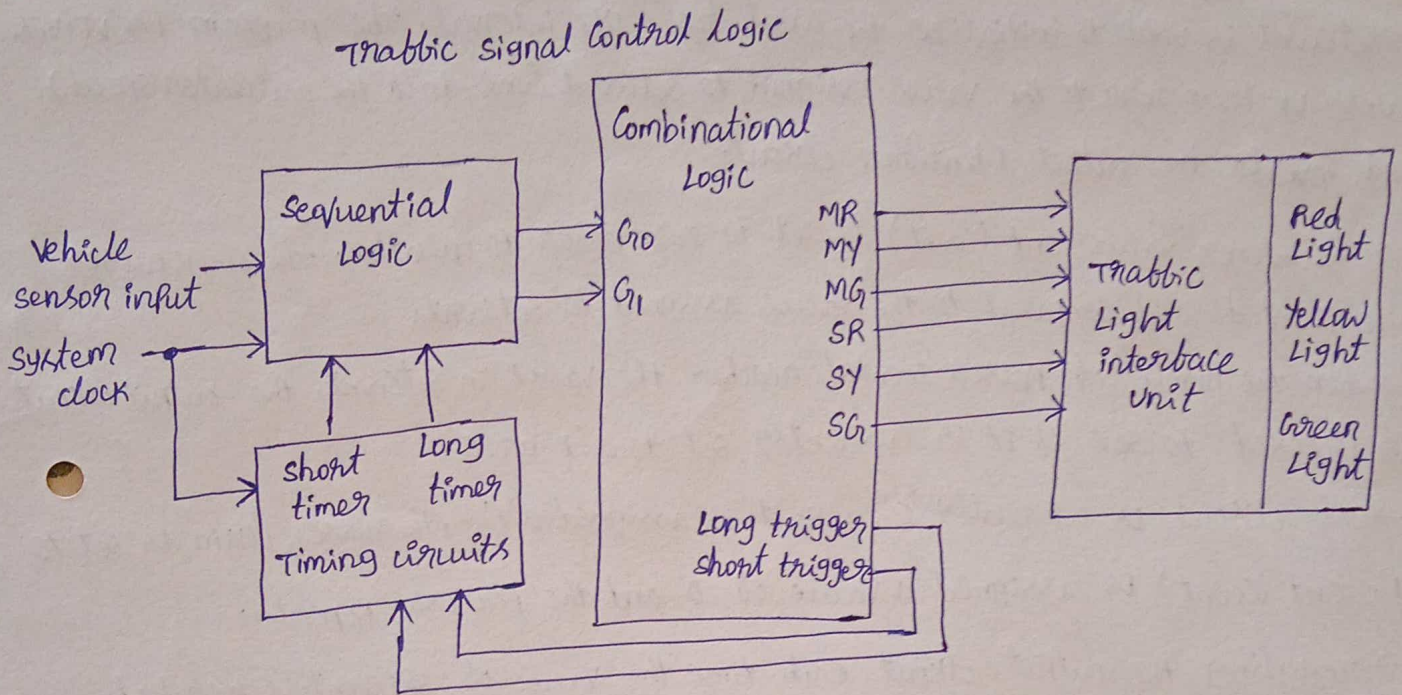


Fig: Block diagram of the Traffic signal controller

i) Timing circuits :- The timing circuits unit of the Traffic signal controller consists of a 25 sec timer and a 4 sec timer and a clock generator. one way to implement this unit is with two 555 timers configured as one-shots and one 555 timer configured as an astable multivibrator.

⇒ Another way to implement the timing circuits, external 24 MHz system clock is divided down to an accurate 1 Hz clock by the frequency divider. The 1 Hz clock is then used to establish the 25 sec and the 4 sec intervals by counting the 1 Hz pulses.

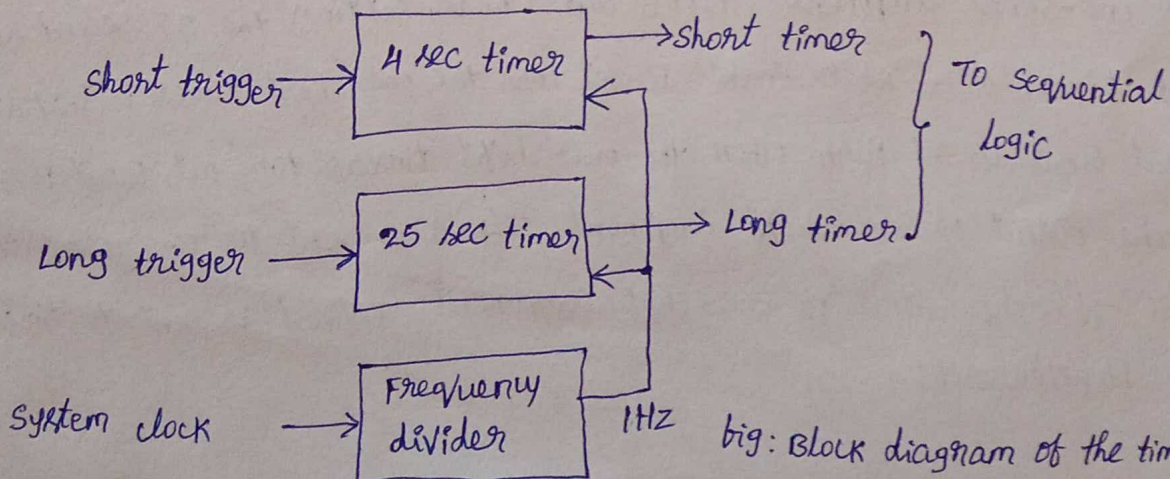


Fig: Block diagram of the timing ckt's.

ii) Frequency Divider: - The purpose of the Frequency divider is to produce a 1Hz clock for the timer circuits. The input clk in this application is a 24.00MHz oscillator that drives the program code.

→ setcount is used to initialize the count for a 1Hz interval. The program FreqDivide counts up from zero to the value assigned to setcount (one-half the oscillator speed) and inverts the output identifier clkout.

→ The integer value cnt (count) is set to zero prior to operation. The clock pulses are counted and compared to the value assigned to setcount.

→ When the number of pulses counted reaches the value in setcount, the output clkout is checked to see if it is currently set to a '1' or '0'.

→ If clkout is currently '0', clkout is assigned a '1'; otherwise, clkout is set to '0'. cnt (count) is assigned a value of '0' and the process repeats.

→ Toggling the output clkout each time the value of setcount is reached creates a 1Hz clock output with a 50% duty cycle.

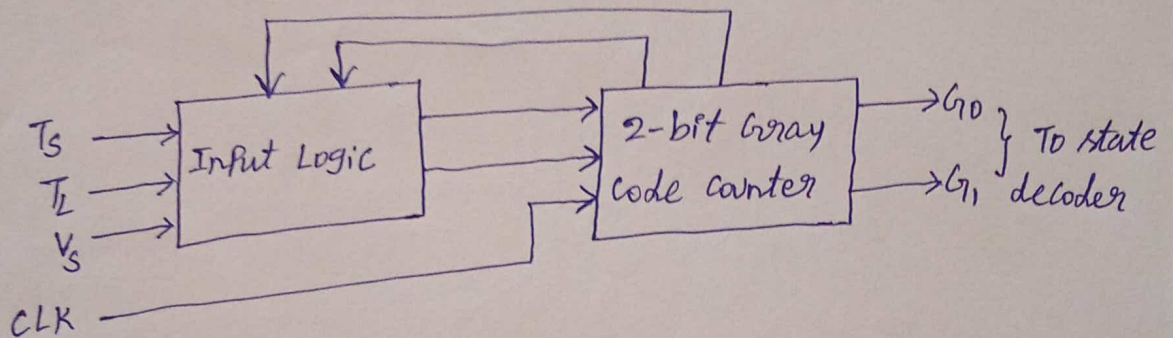
iii) Timer circuits: - The program Timer circuits uses two one-shot instances consisting of a 25 sec timer (TLong) and a 4 sec timer (TShort). The 25 sec and the 4 sec timers are triggered by long trigger and short trigger.

→ Count down timers driven by a 1Hz clock input (clk) replicate the one-shot components TLong and TShort.

→ The values stored in setcountLong and setcountShort are assigned to the Duration inputs of one-shot components TLong and TShort, setting the 25 second and 4-second timeouts. → When Enable is set Low, the one-shot timer is initiated and output Qout is set High. When the one-shot timers time out, Qout is set Low. → The output of one-shot component TLong is sent to Timer circuits identifier TL. The output of one-shot component TShort is sent to timer circuits identifier TS.

Sequential Logic:- The Sequential Logic unit controls the sequencing of the traffic lights, based on inputs from the timing circuits and the side street vehicle sensor. → The Sequential Logic produces a 2-bit Gray code sequence for each of the four states. (d)

The counter:- The Sequential Logic consists of a 2-bit Gray code counter and the associated input logic. The counter produces the four-state sequence on outputs G_0 and G_1 . → Transitions from one state to the next are determined by the short timer (T_S), the long timer (T_L), and vehicle sensor (V_S) inputs.



T_S : short timer (15s), T_L : Long timer (25s),

V_S : vehicle sensor for the side street.

Fig: Block diagram of the Sequential Logic.

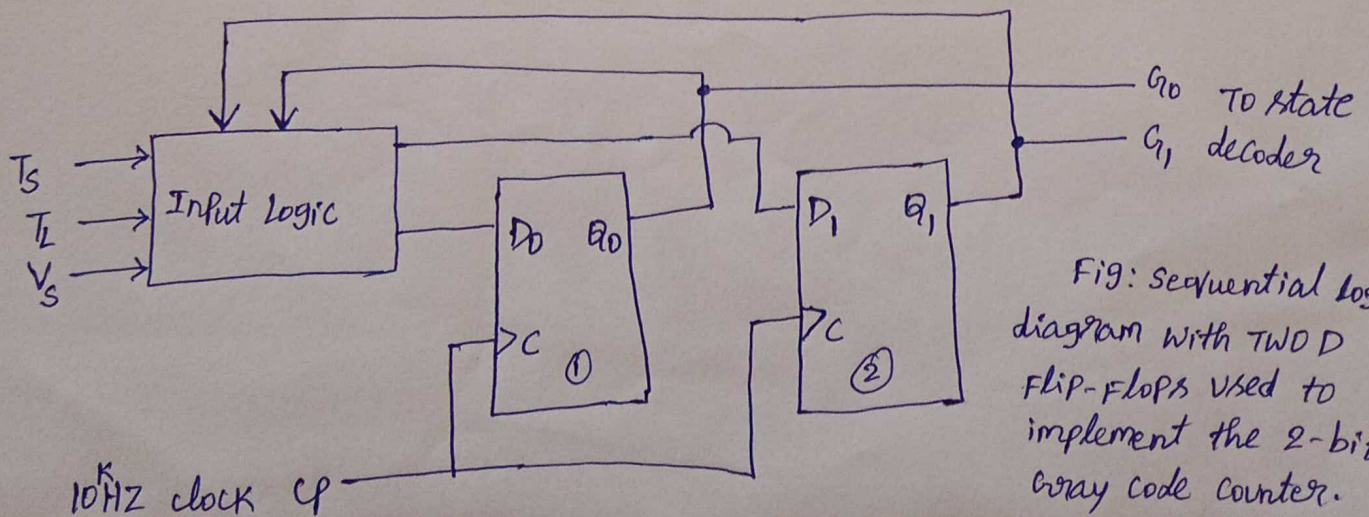


Fig: Sequential Logic diagram with TWO D Flip-Flops used to implement the 2-bit Gray code counter.

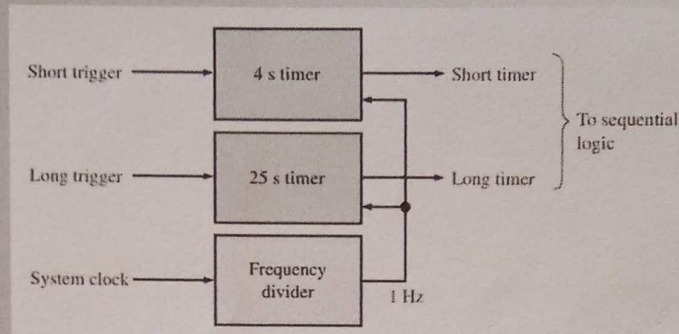


FIGURE 7-65 Block diagram of the timing circuits unit.

Exercise

1. Determine the values for the resistor and capacitor in a 25 s 555 timer.
2. Determine the values for the resistor and capacitor in a 4 s 555 timer.
3. What is the purpose of the frequency divider?

Controller Programming with VHDL

A programming model for the traffic signal controller is shown in Figure 7-66, where all the input and output labels are given. Notice that the Timing circuits block is split into two parts; the Frequency divider and the Timer circuits; and the Combinational logic block is divided into the State decoder and two logic sections (Light output logic and Trigger logic). This model will be used to develop the VHDL program codes.

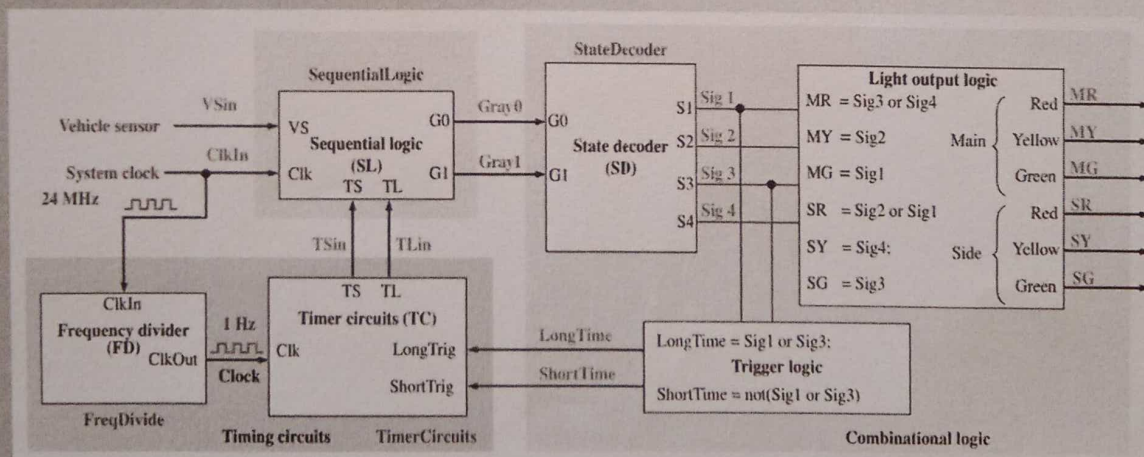


FIGURE 7-66 Programming model for the traffic signal controller.

Frequency Divider The purpose of the frequency divider is to produce a 1 Hz clock for the timer circuits. The input ClkIn in this application is a 24.00 MHz oscillator that drives the program code. SetCount is used to initialize the count for a 1 Hz interval. The program