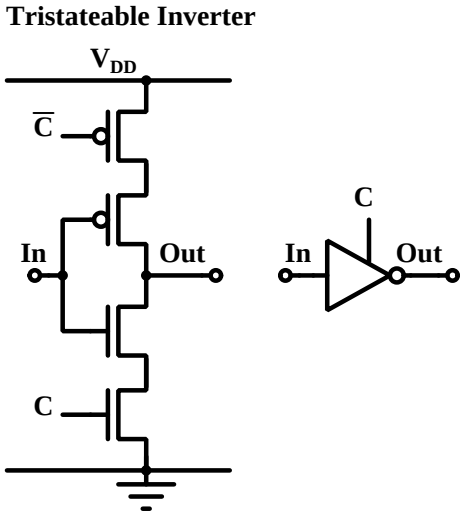
A Review of Digital Circuits

Dinesh Sharma

Department Of Electrical Engineering
Indian Institute Of Technology, Bombay

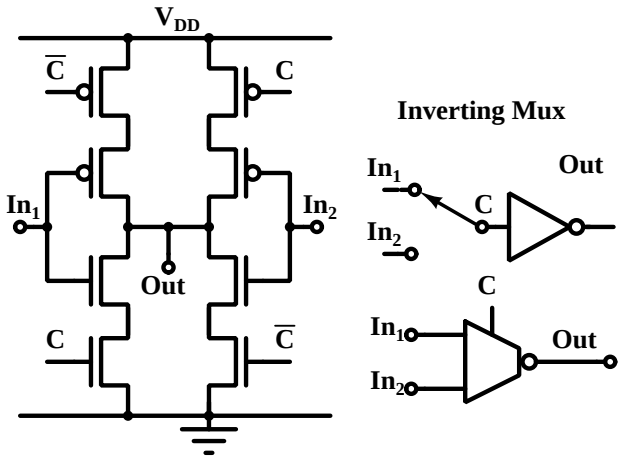
January 17, 2021

Multiplexers and demultiplexers for routing data



The figure on the left shows a tri-stateable inverter.

When C = '1', Out = $\overline{\text{In}}$.
When C = '0', Out is disconnected (Z state).

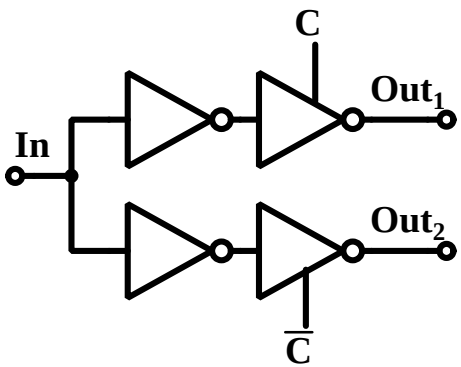


Using two tri-stateable inverters, we can make an inverting multiplexer.

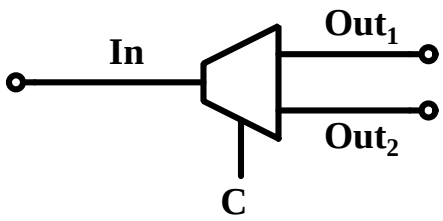
When C = '1', Out = $\overline{\text{In}_1}$, and
When C = '0', Out = $\overline{\text{In}_2}$

Multiplexers and demultiplexers for routing data

- A multiplexer selects one out of many signals and puts it on a single line.
- A de-multiplexer carries out the reverse operation - It takes a digital signal from a single line and copies it to a selected line out of multiple lines.



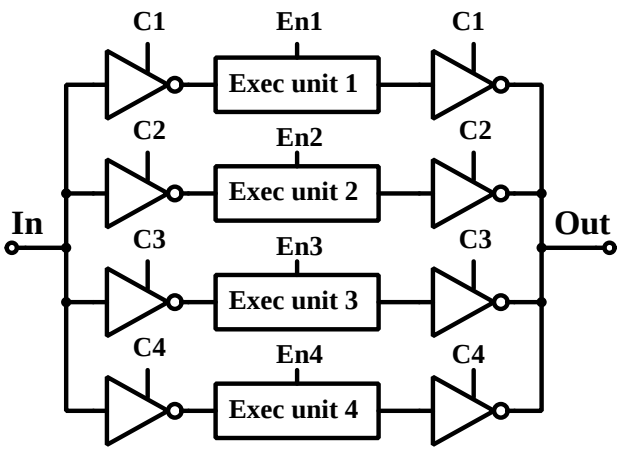
We can use two tri-stateable inverters with a common input enabled by complementary signals to implement a de-multiplexer.



We can add ordinary inverters if we do not want inversion, as shown in the figure here.

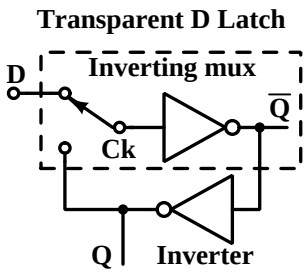
Multiplexers and demultiplexers for routing data

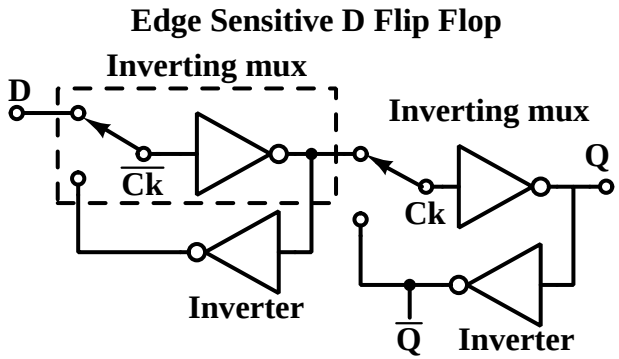
- Circuits shown in previous slides were ‘two way’ muxes and de-muxes.
- By adding a decoder and using its multiple outputs instead of C and $\overline{C}$, we can make a ‘2^n way’ mux or de-mux.



- The figure on the left shows an example of routing through an enabled circuit.
- In general, data paths will be multi-bit and therefore banks of demultiplexers/multiplexers will be required.

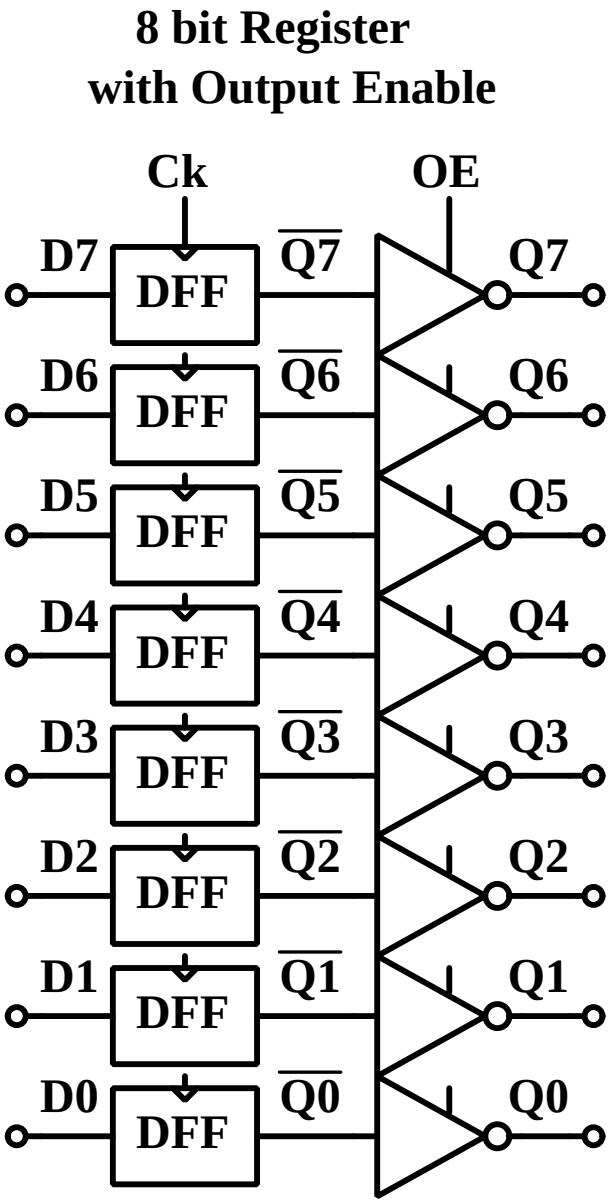
D Latches and Flip Flops





- The circuit on the left shows a transparent latch. When Ck is 'High', Q is just a buffered copy of D. Any change in D will result in a change in Q, so this is called the transparent state.
- When the clock is low, D is disconnected and the inverter form a latch. This is the latched state.
- The circuit at the left bottom uses two transparent latches with complementary clocks to form a master-slave flip flop. This is edge sensitive and transfers D to Q on a positive clock edge.

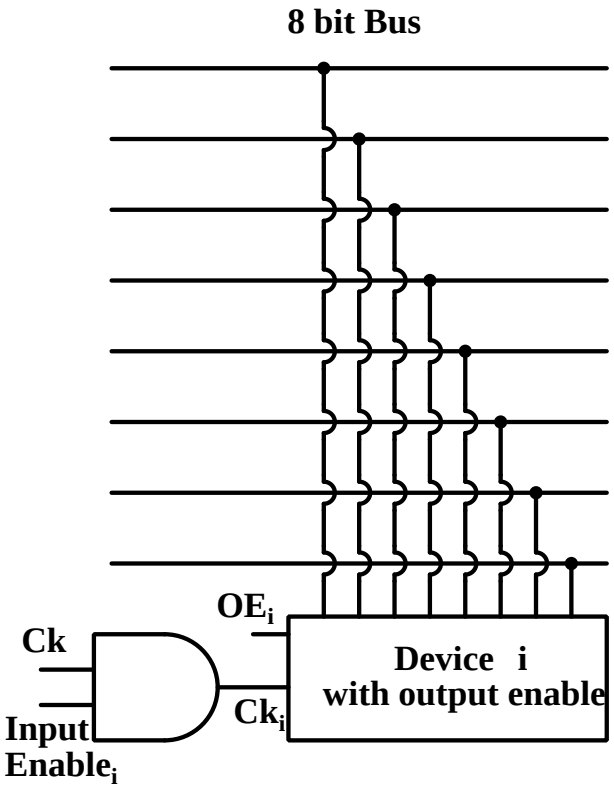
D Latches and Flip Flops



- We can stack multiple D flipflops to provide multi-bit storage.
- Circuit on the left uses 8 D flipflops with a common clock to make an 8 bit register.
- Outputs are driven by tri-stateable inverters, enabled by a common output enable signal.
- The register will drive its outputs only when enabled. Otherwise its outputs are disconnected.

Data buses

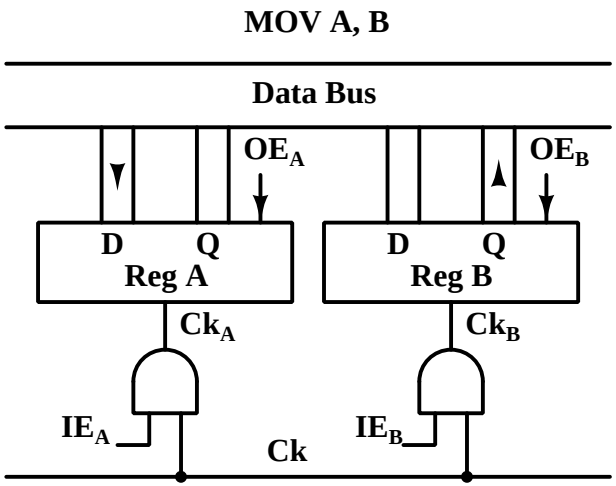
A bus is a common data path connecting multiple devices. An 8 bit bus is shown below.



- The bus needs a bus controller which provides the output enable and input enable signals.
- Each device drives the bus only when its output enable signal is ‘TRUE’.
- It latches its inputs only when the input enable signal is ‘TRUE’.
- The bus controller ensures that only one device has its outputs enabled at a given time.

Implementing a MOV instruction using a Bus

In a register file, D inputs as well as Q outputs of registers are connected to a common bus.



- The instruction MOV A, B copies the contents of register B to register A.
- To implement this instruction, the CPU makes OE_B 'TRUE', so the Bus is driven to the contents of B.
- The CPU makes IE_A 'TRUE'.
- The D inputs of A are connected to the bus, and hence to the Q outputs of B. When the clock comes, it will latch the contents of B into register A.

Semiconductor Memories

- Most electronic systems require storage and retrieval of data. Systems requiring a small amount of storage can use flipflops for this purpose.
- However, Flipflops are not efficient for large storage requirements. For large storage, we need specialised arrangements of storage elements called memories.
- Depending on their characteristics, memories are classified as
  - Static Random Access Memories (SRAM),
  - Dynamic Random Access Memories (DRAM),
  - Read Only Memories (ROMs)
  - Electrically Programmable Memories (EPROMs)
  - Electrically Programmable and Erasable Memories (EEPROMs)
  - Flash Memories
- Practically all memory systems these days use semiconductor memories. (Magnetic “cores” were used earlier for storage. Some memory terminology is acquired from those times).

Semiconductor Memories

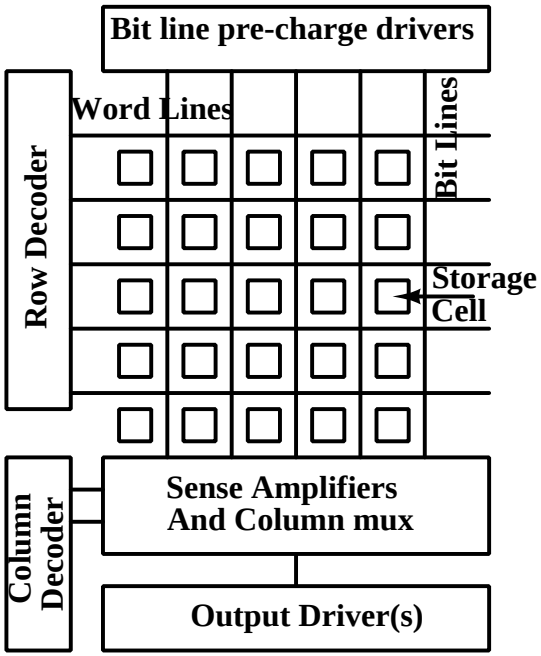
- Data stored in a memory can be accessed sequentially or in a random order.
- For Random Access, an address is provided to the memory, which then returns the stored content from that address or writes new data at this address.
- Conveying the address to the memory takes time, which may slow down memory access and affect system performance adversely.
- Many modern memory systems use a “burst mode” in which a start address is supplied, and then data is read or written from sequential addresses in a burst.
- The address is automatically adjusted in the memory by incrementing the initially supplied address with each read or write in a burst.
- While the storage mechanism varies from one type of memory to another, the internal organisation and the method for accessing the contents are very similar for all types of memories.

Semiconductor Memories

- At each unique address, the memory can store a single bit or multiple bits with a width of various sizes. Correspondingly, the memory is said to be bit oriented or byte or word oriented.
- The stored data may last only as long as power is applied to the memory. Such storage is called volatile.
- Some memories can retain their information even when power is removed. Such memories are called non-volatile.
- A memory must include circuits to decode the supplied address in order to activate a specific storage location associated with this address for read or write operation.
- A read or write operation must start only after address decoding is complete. Otherwise, a read or write may occur from a storage cell corresponding to a transient value of the changing address.
- This can result in data corruption.

Memory Organization

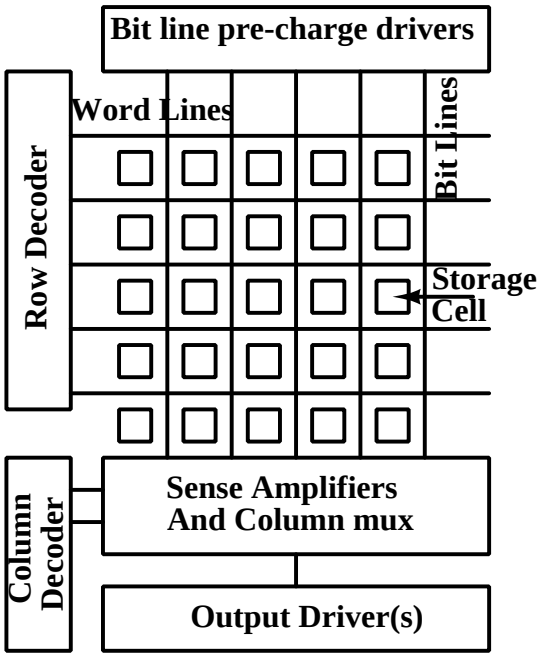
It is important to list the sub-units which are used in a typical memory. Details of their operation will vary depending on the type of memory, but the overall organisation is quite similar.



- Row/Column Address decoders provide the logic used for decoding the address in order to activate the selected memory location.
- Memory array is a two dimensional array in which the storage elements are arranged. The address decoder typically selects a row and a column to identify the memory cell being addressed.
- A sense amplifier is used in every column to amplify the small signal provided by the addressed cell and to amplify it to a rail-to-rail logic level

Memory Organization

The long bit lines in a memory would present a very heavy load for the tiny storage cells to drive. Therefore these are pre-charged to a voltage close to V_{DD}.



- An addressed storage cell, when connected to these lines will not change their voltage level if the connected node is at '1'.
- If the connected note is at '0', it will lower the voltage of the bit line by a small amount.
- The sense amplifier detects the presence or absence of this small voltage dip and amplifies it to a full scale '0' or '1'. For reliable operation, we run differential lines (bit and $\overline{\text{bit}}$) and connect these to Q and $\overline{Q}$ outputs of the storage cell.
- Output drivers provide sufficient drive to take the off-chip lines quickly to a '0' or a '1'.

Memory Timing parameters

The speed of a memory subsystem is characterized by several timing parameters.

Read Access time: This is the delay between presentation of an address to a memory and the availability of data at that location.

Row and Column Address strobes Often the address is specified in two halves – known as Row Address and Column Address. Each half of the address is latched into the memory using strobe signals. These strobes are known as Row Address Strobe (RAS) and Column Address Strobe (CAS) signals.

RAS/CAS times The Read Access time is broken into two halves. RAS to CAS delay (T_{RCD}) is the time gap required between the Row and column address strobes. CAS latency (CL) is the delay between Column strobe and availability of data.

Memory Timing parameters

Memory systems pre-charge long internal lines connected to a memory array and then sense the contents of a selected location by connecting it to the pre-charged lines and

Write recovery time: This is the interval needed between a write and the pre-charge operation. Since a pre-charge operation forcibly takes all bit lines to '1'.

Memory cycle time: This is the time required to complete successive read or write operations. This determines the rate at which one can issue new read or write requests to a memory. The read and write cycle times need not be the same, but most systems specify a single cycle time, which is the longer of the two.

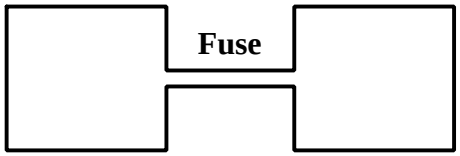
Types of storage Cells:ROM

Read only memories are those which are programmed just once with fixed data and which will be read many times but never altered.

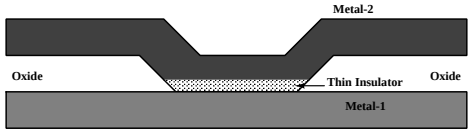
- In Read Only Memories, we use a permanently ON or OFF switch as a memory element.
- The switch may be implemented using MOS transistors. The transistor is turned ON or OFF permanently by connecting its gate to V_{DD} or ground at the time of chip fabrication using a mask. Such memories are called Mask Programmable ROMs.
- ROMs may also be made using poly-silicon fuses or anti-fuse structures as switches. These were discussed during our discussion on Field Programmable logic. A fuse is blown or an anti-fuse shorted to program data into the One Time Programmable (OTP) ROM or PROMs.

Types of storage Cells:ROM

Fuse and anti-fuse type structures can be used to connect or disconnect devices to implement ROM cells.



- A fuse structure uses a narrow neck in an interconnect, which can be blown by passing a heavy current through it.
- A memory cell can be programmed to ’0’ or ’1’ by connecting or disconnecting a node to ground directly or through a transistor.



- An anti-fuse structure uses a structure with a thin insulator between two connecting wires.
- The insulator can be shorted by applying a high voltage across it.
- This provides the means of connecting or not connecting a node to ground programmably.

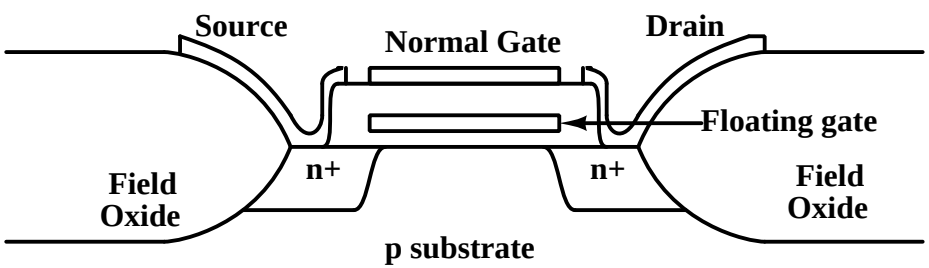
Types of storage Cells:EPROM

A variant of ROMs are Electrically Programmable ROMs or EPROMs. The data in these memories may be altered by using unusual operating conditions. The time taken to alter the contents of an EPROM is long, so the programming is carried out infrequently.

- We can implement permanently ON or OFF transistors by injecting charge into the gate region. Charge can be injected into the gate region by generating energetic carriers by avalanching and applying a field of appropriate sign between the gate and substrate.
- The injected charge into the semiconductor remains trapped in the insulating gate material, which changes the turn on voltage V_T. Thus an nMOS can be turned ON at zero gate bias by injecting holes into the gate region and making its V_T negative.
- Such memories are erased by exposure to Ultra Violet light, which releases the trapped charge from the gate oxide back into silicon.

Types of storage Cells:EEPROM

Erasure of EPROMs is through exposure to UV light, which requires a specially packaged device which will let in UV light during erasure through a quartz window. This can be avoided using electrically erasable programmable ROMs or EEPROMs.



EEPROMs use MOS structures which are programmed in a similar manner to EPROMs, but which permit erasure through application of a high field opposite to that used for injecting charge into the gate.

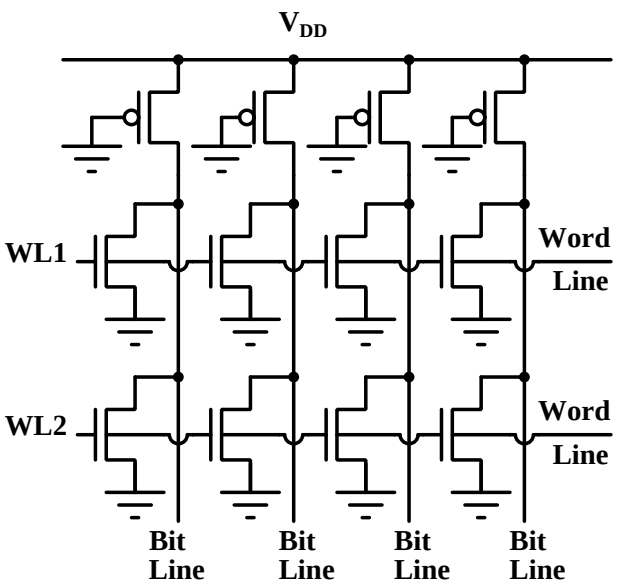
A floating gate MOS device, which uses an extra floating gate embedded in the oxide to store injected charge, can be used for this.

Types of storage Cells:EEPROM

- The process of high field injection and removal of oxide charge results in some damage to the MOS device.
- So the process of programming and erasing can be carried out for a limited number of times, after which the repeated charge injection and removal damages the device to such an extent that it cannot be programmed or erased any more.
- The number of times that a device can be programmed and erased is called the endurance of an EEPROM.

EEPROM configurations: NOR EEPROM

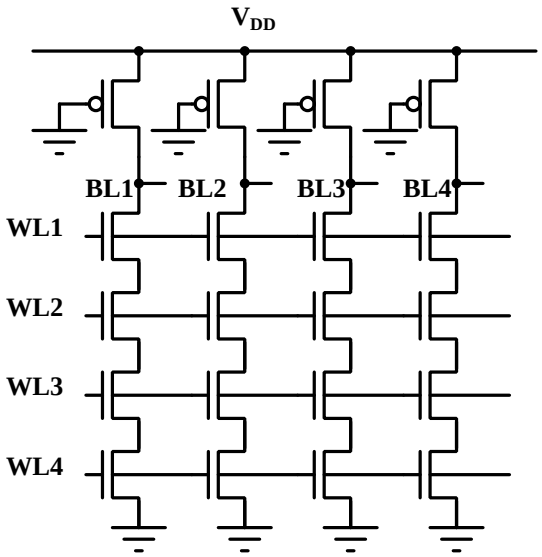
To address and read/write EEPROMs, programmable transistors can be connected in different ways.



- In practically all memory devices, there are perpendicular interconnect lines called Word lines and bit lines.
- In the NOR configuration of EEPROMs, bit lines are pulled up using permanently ON pMOS devices, as in pseudo nMOS logic.
- A programmable nMOS transistor is connected to ground at each crossing of a word line and a bit line.
- If the nMOS is ON, the bit line is pulled down to zero. If it is OFF, the bit line remains high due to the pull up transistor.

EEPROM configurations: NAND EEPROM

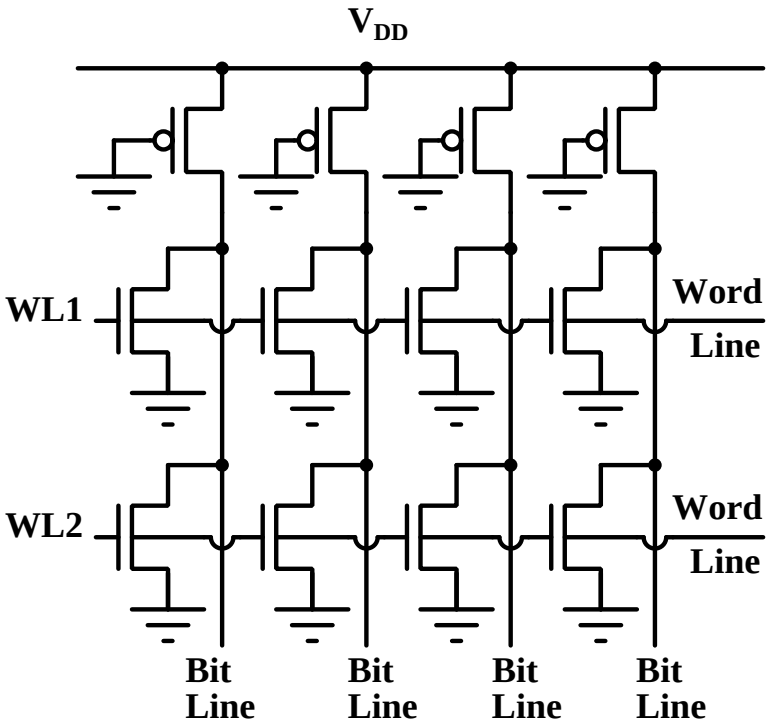
In a NAND EEPROM also, we have a pseudo nMOS like arrangement of transistors, with pMOS pull ups for each bit line.

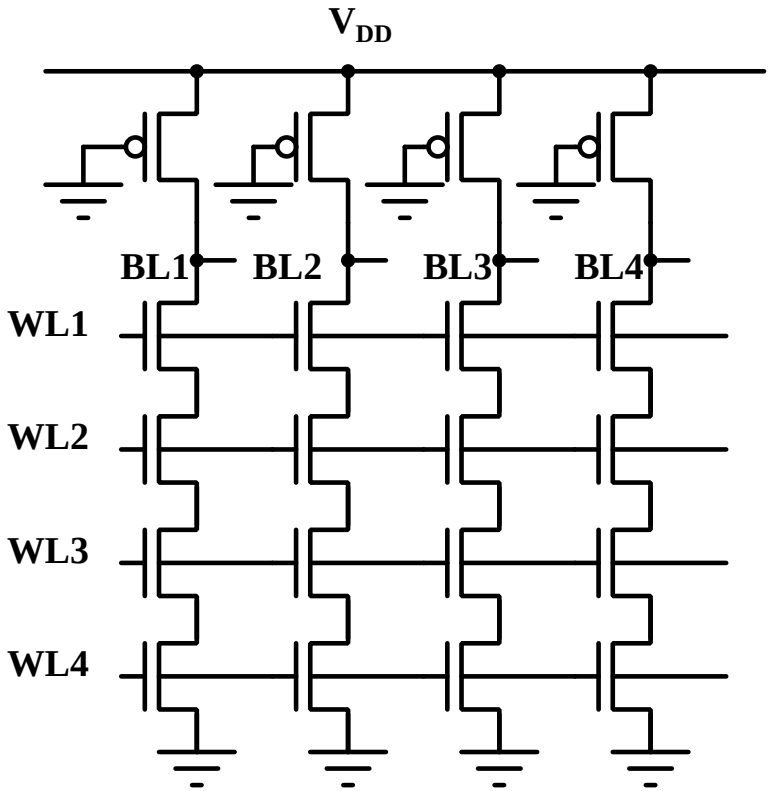


- nMOS transistors are all connected in series in a particular column.
- All word lines except the selected one are kept at ’1’, while the selected world line is kept at ‘0’.
- If the selected transistor in ON, it pulls the bit line output lower. Otherwise it remains high.
- Unlike pseudo nMOS, we need not pull the output all the way down to logic ‘0’. This enables us to put a lot of nMOS transistors in series.

The selected column (bit line) is multiplexed to a sense amplifier, which brings the output to a proper level for logic ‘0’ or ‘1’.

EEPROM configurations: NOR and NAND



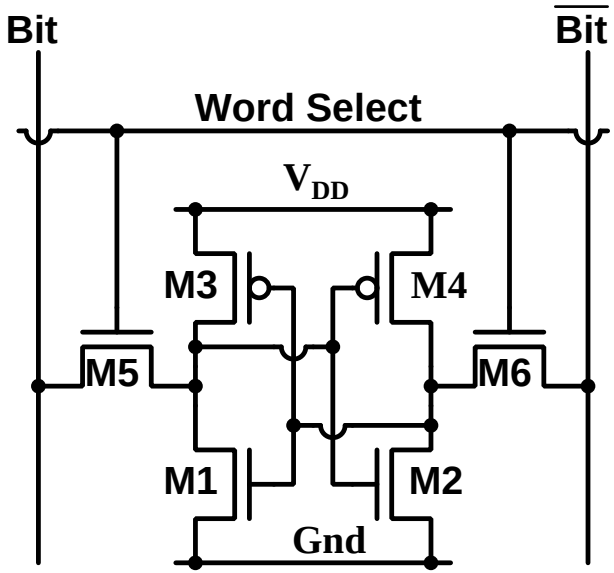


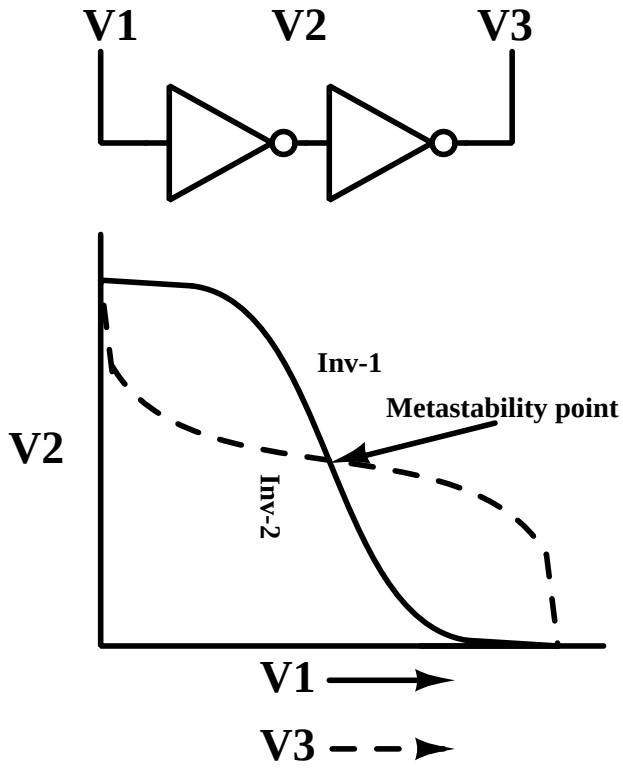
- NAND EEPROMs are much denser, because these don't need a contact window for every nMOS transistor.
- NOR is faster, because discharge is through a single transistor.
- Practically all thumb drives etc. use a NAND configuration these days.

Read/Write memories

- Read/write memories provide comparable times for read and write operation.
- These are suitable for applications where memory has to be modified frequently – such as data storage for a micro-processor based system.
- Static Read-Write memory (SRAM) uses a simple latch to store the data. The latch is implemented as two cross connected inverters and is accessed through pass transistors.
- Data in static Read Write memory will be retained as long as power is applied. This kind of storage is called volatile storage.
- Dynamic read-write memory (DRAM) stores data on a small sized capacitor. It consists of the capacitor and a single coupling transistor.
- DRAM is much denser, but needs frequent refreshing of the data, since the charge stored on the capacitor can leak away.

SRAM cell



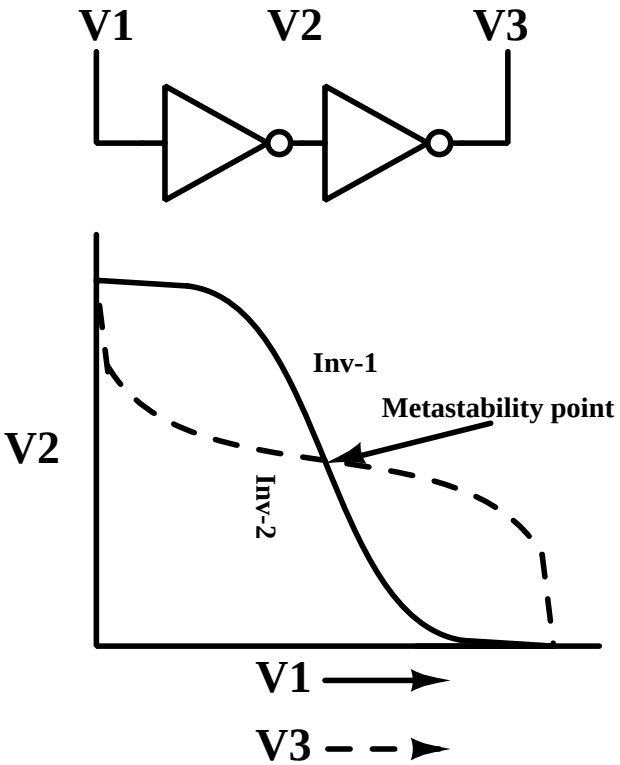


The 6 Transistor RAM cell contains cross connected inverters coupled to the bit and $\overline{bit}$ lines through pass transistors.

The pass transistors are enabled by the word select line.

SRAM cell: Butterfly diagram

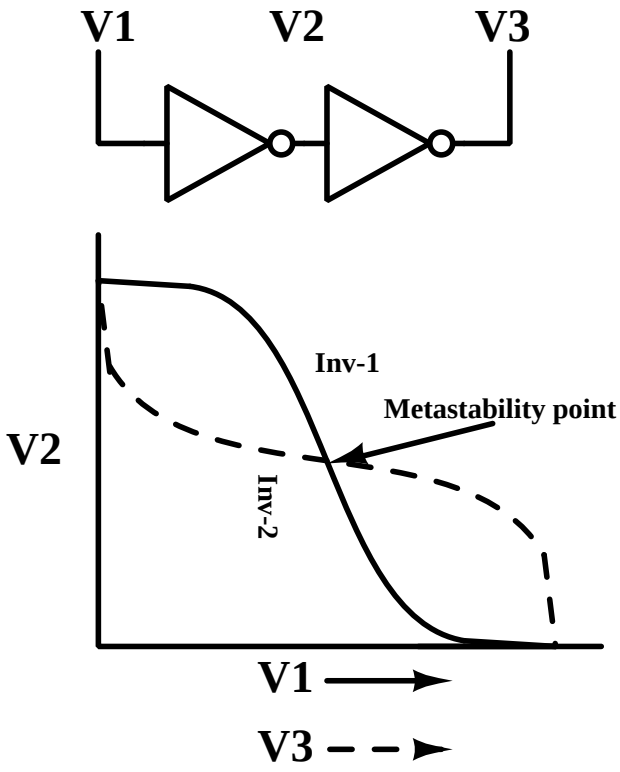
If we remove the feedback connection in the latch used in an SRAM cell, we get the two inverters as shown below.



- The solid line curve is the transfer characteristic of the first inverter (V2 vs V1).
- The dashed line curve is the transfer characteristic of the second inverter, with its input voltage V2 along the Y axis and its output voltage V3 along the X axis.
- This curve is known as a Butterfly diagram due to its shape.

In the latch, V3 is shorted to V1. So the static characteristics of the latch are satisfied where the solid line and the dashed line curves cross (V1=V3).

SRAM cell: Stable and Metastable points

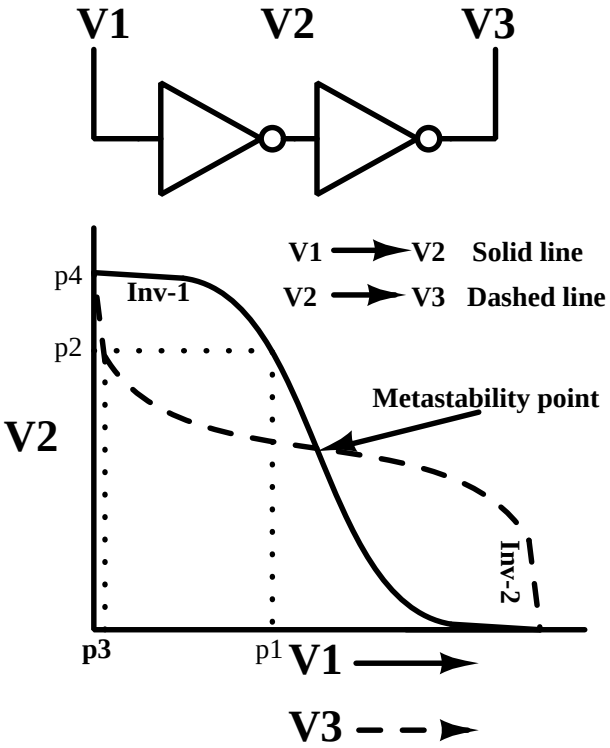


- The solid and dashed line curves cross at 3 points.
- At the left most point, $V1 = 0$, so $V2 = V_{DD}$ by the solid line curve.
- For $V2 = V_{DD}$, $V3 = 0$. by dashed line curve. So $V1 = V3$.
- At the right most point, $V1 = V_{DD}$, so $V2 = 0$ (solid line curve). For $V2 = 0$, $V3 = V_{DD}$ (dashed line curve). So $V1 = V3$ again.

$V1 = V3$ at the middle point as well. However, this is an unstable point as the least bit of perturbation from this point leads to one of the two stable points described above.

SRAM cell: Stable points

Suppose the latch is stable at the left crossing point and we perturb the voltage at V1.

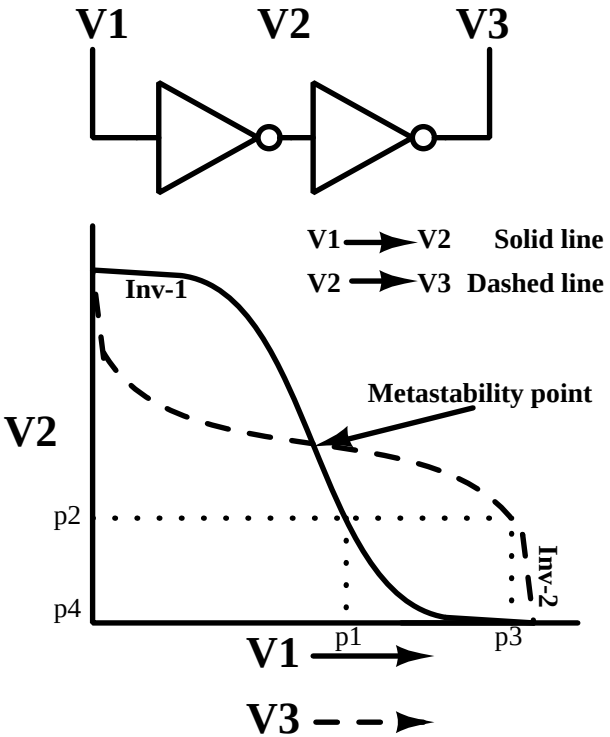


- If we perturb V1 to the point p1, the output of Inv1 will be p2 (solid line transfer curve for Inv1).
- With its input at p2, the output of Inv2 will be p3 (dashed line transfer curve for Inv2).
- But the output of Inv2 is shorted to input of Inv1 in the latch. With its input at p3, the output of Inv1 will be p4 (solid line transfer curve for Inv1).

In fact, perturbing V1 to any point to the left of the middle crossing will quickly recover to the left crossing point with $V1 = 0$ and $V2 = V_{DD}$.

SRAM cell: Stable points

What happens if the latch is stable at the right crossing point and we perturb the voltage from this point?



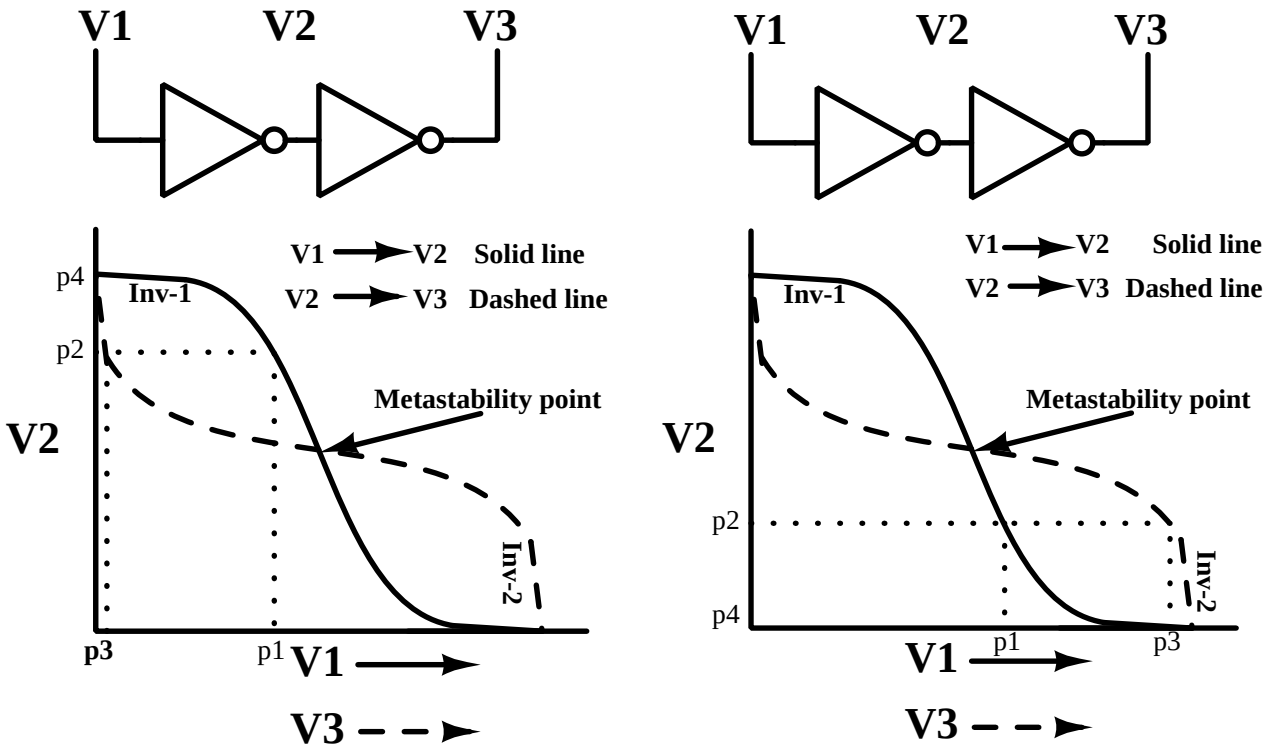
- If we perturb V1 to the point p1, the output of Inv1 will be p2 (solid line transfer curve for Inv1).
- With its input at p2, the output of Inv2 will be p3 (dashed line transfer curve for Inv2).
- But the output of Inv2 is shorted to input of Inv1 in the latch. With its input at p3, the output of Inv1 will be p4 (solid line transfer curve for Inv1).

In fact, perturbing V1 to any point to the right of the middle crossing will quickly recover to the right crossing point with $V1 = V_{DD}$ and $V2 = 0$.

SRAM cell: metastable point

What about the middle crossing point?

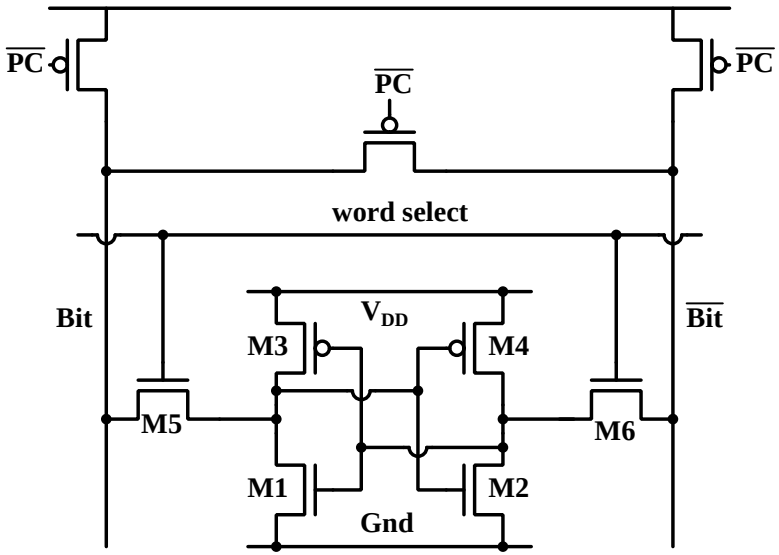
Network equations are satisfied here – The output voltage of Inv1, when applied to Inv2 gives back the same voltage.



However, perturbing even by a small amount to left will make the system go to the left crossing point and even by a small amount to the right will make the system go to the right crossing point.
So this point is not stable and is called the metastable point.

SRAM cell: Read cycle

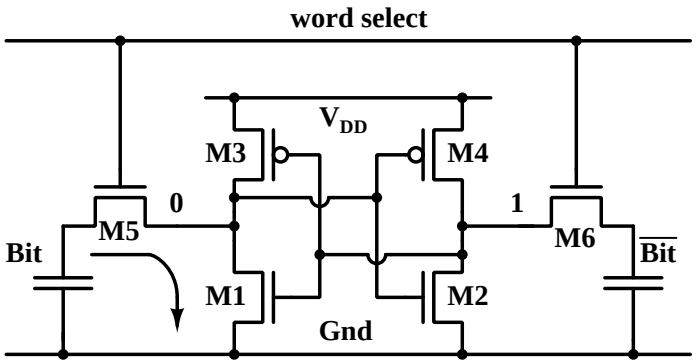
In order to read the memory, the address is placed by the processor on the address lines. The row and column addresses are decoded by the memory.



- The row decoder makes the Word Select line of the addressed row 'high'.
- On receiving the $\overline{Rd}$ command, the memory generates a pre-charge pulse ($\overline{PC}$) to precharge the Bit and $\overline{\text{Bit}}$ lines to 'high'.

The column address decoder connects the Bit and $\overline{\text{Bit}}$ lines of the selected column to the sense amplifier. Depending on the data stored in the cell, one of the outputs of the latch will be 'high' while the other will be 'low'.

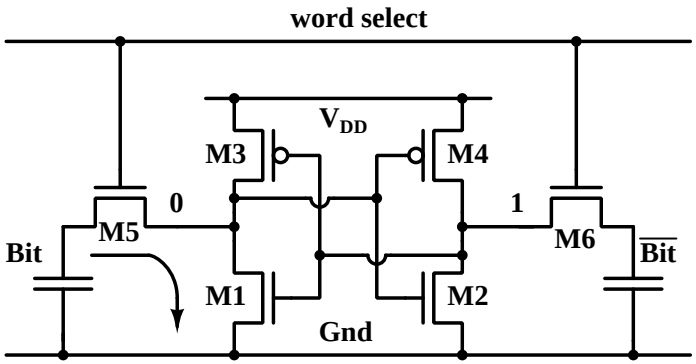
SRAM cell: Read cycle



- One of the outputs of the latch is high, while the other is low.
- Bit and $\overline{Bit}$ lines are shown here as capacitors.

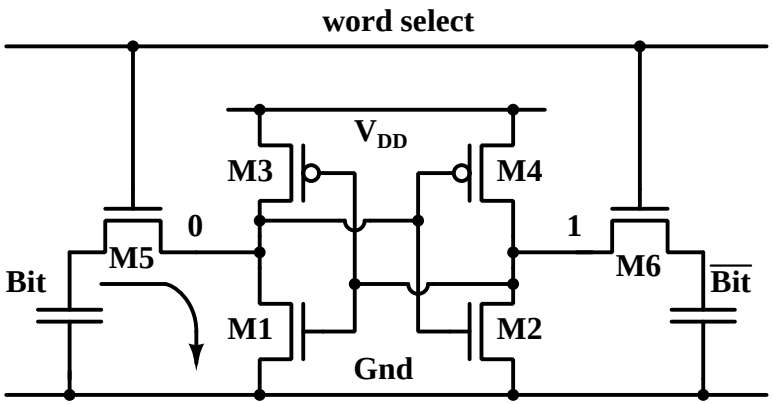
- The high node is unchanged, but the low node starts discharging the Bit or the $\overline{Bit}$ line (whichever is connected to it)
- This creates a voltage difference between the Bit and $\overline{Bit}$ lines.
- The sense amplifier amplifies the voltage difference and produces a digital ‘0’ or ‘1’ at its output.
- This digital data is then buffered to the output pad by a tristateable driver, which is activated only during read cycles.

SRAM cell: Read upset



- At the end of the cycle, the word select line returns to 'low', disconnecting the latch from Bit and $\overline{Bit}$ lines.
- The low node, which would have been pulled a little high due to connection with the pre-charged Bit or the $\overline{Bit}$ line is restored back to 0V by the feedback action as discussed earlier.
- The geometry of the access transistors has to be carefully determined to ensure that the low node is not pulled to too high a voltage when it is connected to the pre-charged Bit or $\overline{Bit}$ line.
- If the voltage of the low node goes too high when it is connected to the pre-charged Bit or $\overline{Bit}$ line, it may zap to a '1' value rather than going back to '0'.
- This is called a read upset.

SRAM cell: Read upset



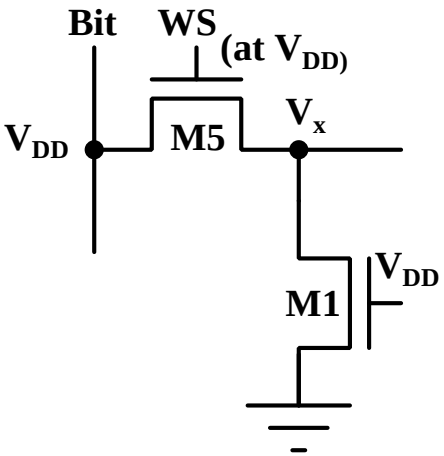
- consider the memory cell with a '0' stored in it. M3 and M2 are OFF.
- We find the condition that the voltage at the drain of M1 remains below V_{Tn}. Then M2 will remain OFF during the entire read cycle.

Since M2 remains OFF, The voltage at the gate of M1 will remain at V_{DD} for the entire cycle.

Since the drain and gate voltages of M5 are equal (and at V_{DD}), it is in saturation.

The gate of M1 is at V_{DD} while its drain is below V_{Tn}, so M1 remains in linear regime.

SRAM cell: Read upset



Equating currents through M5 and M1:

$$\frac{K_{n5}}{2}(V_{DD}-V_x-V_{Tn})^2 = K_{n1}\left((V_{DD}-V_{Tn})V_x - \frac{1}{2}V_x^2\right)$$

We define $\beta \equiv K_{n1}/K_{n5}$. This gives:

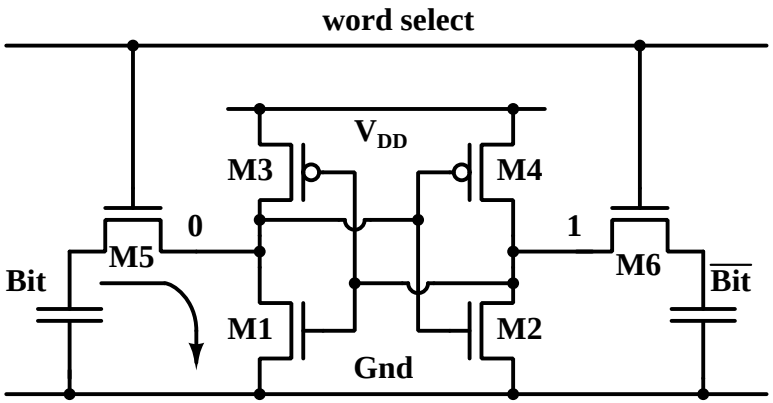
$$(V_{DD}-V_x-V_{Tn})^2 = 2\beta V_x(V_{DD}-V_{Tn}-\frac{1}{2}V_x)$$

In the limiting case, $V_x = V_{Tn}$. Then:

$$(V_{DD}-2V_{Tn})^2 = 2\beta V_{Tn}(V_{DD}-\frac{3}{2}V_{Tn}) = \beta V_{Tn}(2V_{DD}-3V_{Tn})$$

$$\text{Hence,} \quad \beta = \frac{(V_{DD}-2V_{Tn})^2}{V_{Tn}(2V_{DD}-3V_{Tn})}$$

SRAM cell: Read upset



$$\beta = \frac{(V_{DD} - 2V_{Tn})^2}{V_{Tn}(2V_{DD} - 3V_{Tn})}$$

If we take $V_{Tn} = V_{DD}/5$, we get

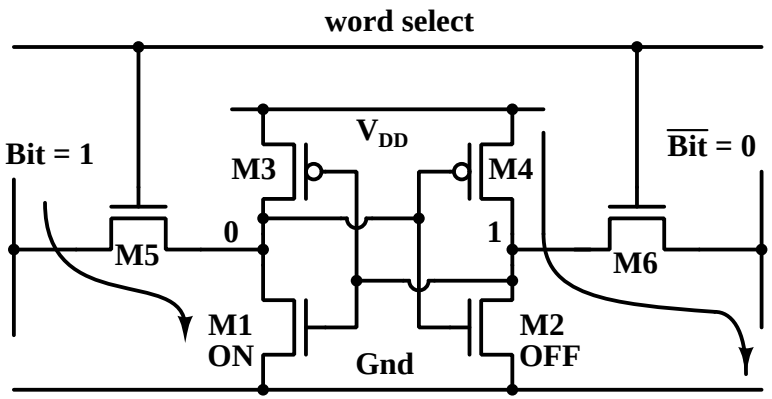
$$\beta = \frac{(5V_{Tn} - 2V_{Tn})^2}{V_{Tn}(10V_{Tn} - 3V_{Tn})} = \frac{9V_{Tn}^2}{7}V_{Tn}^2 = 9/7 = 1.2857$$

- Thus the value of β should be ≥ 1.3 to keep V_x below V_{Tn}.
- Notice that the geometry ratio of M5 and M1, as well as the maximum voltage rise at V_x is fixed by this consideration.
- This must be kept in mind while discussing the write operation.

SRAM cell: Write cycle

During the write cycle,

- The address is placed by the processor on the address lines. and the row and column address are decoded by the memory.
- When the word line goes HIGH, the access transistors are turned on for **all** cells in this row.
- When $\overline{\text{Wr}}$ is asserted, the Bit and $\overline{\text{Bit}}$ lines are driven to Data and $\overline{\text{Data}}$ respectively **in the selected column only**.

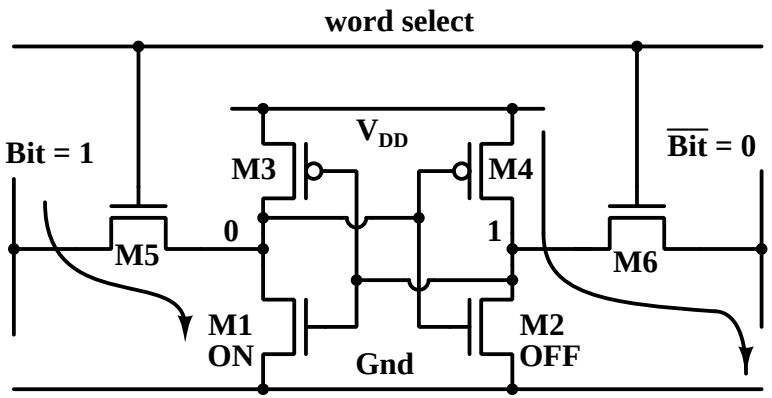


Thus, in the selected column of the selected row, the cell goes through a write cycle.

For all the remaining columns in the selected row, cells go through a read cycle.

SRAM cell: Write cycle

When the word line goes HIGH, the access transistors are turned on for all cells in this row.

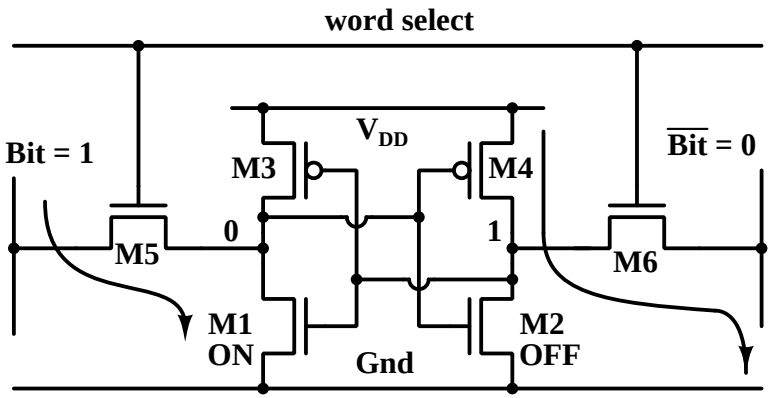


- In the selected column, the complementary values of Data and $\overline{Data}$ are copied to the latch through the pass transistors.
- Once the voltage at the node connected to the pass transistors passes the meta-stability point, the positive feed back ensures that it will go all the way to '0' or '1'.

After the write process is complete, the Word line is brought low again.

SRAM cell: Write cycle

- Considerations of Read upset fix the ratio between the pass transistor and pull down transistor width.
- The voltage rise at the node which is at '0' is limited to $\approx V_{Tn}$, with the bit line at '1'.
- Then how do we write a '1' to the cell – which requires the node to be taken above $\approx V_{DD}/2$?

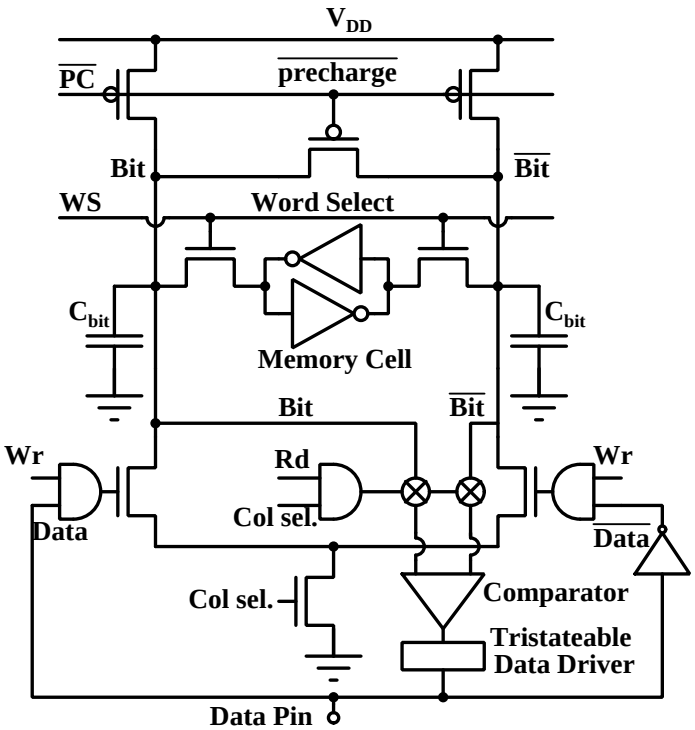


- The major work for writing a '1' has to be done at the 'high' node which must be pulled down to well below $V_{DD}/2$.
- This consideration fixes the geometry ratio of M4 and M6.

Notice that M1, M3 and M5 should have the same size as M2, M4 and M6 respectively.

SRAM cell: Full Data path

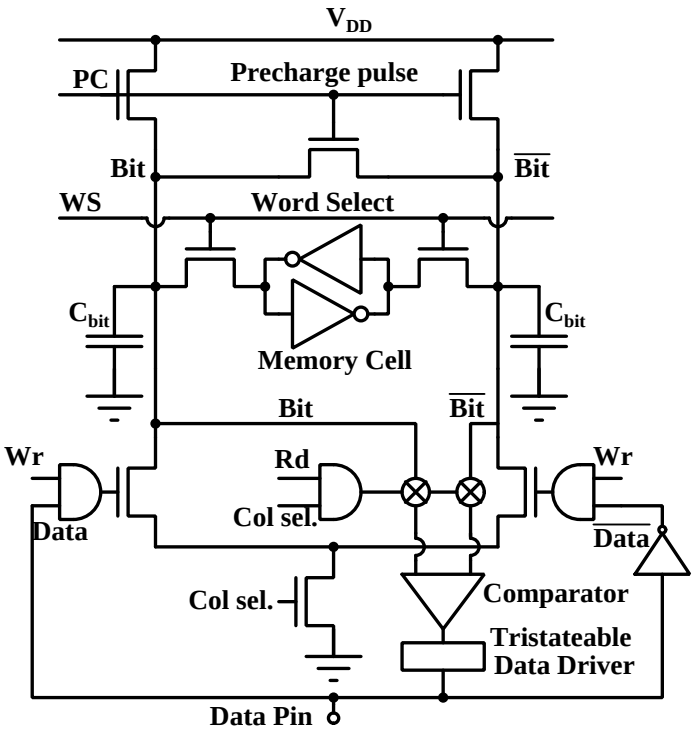
The data path during read and write cycles is shown below.



- All 3 pMOS transistors on the top turn on during pre-charge. The weak shorting transistor ensures that the voltages on Bit and $\overline{\text{Bit}}$ are equal.
- The pass gates connected to Bit and $\overline{\text{Bit}}$ lines are activated only if this is a read cycle and this column has been selected.
- When activated, these feed the sense amplifier/comparator, which then drives a tri-stateable buffer to the data pin.

SRAM cell: Alternate Bit line pull up

We can use nMOS transistors instead of pMOS to pre-charge the Bit and $\overline{\text{Bit}}$ lines. The pre-charge pulse should now be a positive pulse.

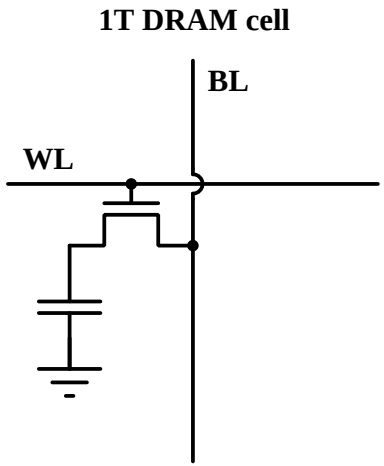


- Mobility of n channel transistors is higher and the source follower configuration has lower output impedance. So charging is much faster.
- However, the maximum voltage to which the Bit and $\overline{\text{Bit}}$ lines can be charged is now $V_{DD} - V_{Tn}$.
- Transistor geometries have to be adjusted due to lower voltage on the Bit and $\overline{\text{Bit}}$ lines.

Lower voltage on Bit and $\overline{\text{Bit}}$ lines is actually an advantage for the design of sense amplifiers, whose common mode range does not have to go all the way to V_{DD}.

DRAM cell

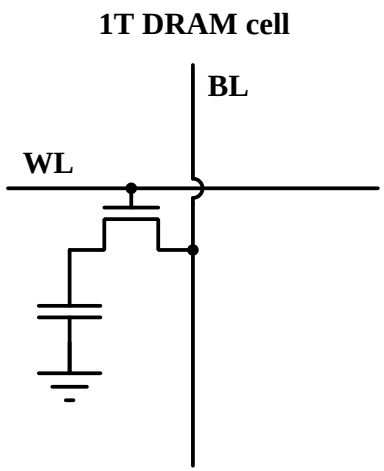
- DRAM uses a single transistor per cell, so it is much denser than SRAM.
- The storage is on a capacitor – so it needs periodic refreshing to avoid data loss due to leakage.
- To store sufficient charge on the capacitor, special technological steps are required. So it is not process compatible with other CMOS circuits.



- In the read cycle, the bit line is pre-charged 'high'.
- If the capacitor stores a '0' (it is discharged), it pulls the voltage on the bit line a little lower due to charge sharing.
- If it stores a '1', the voltage on the bit line remains unchanged.

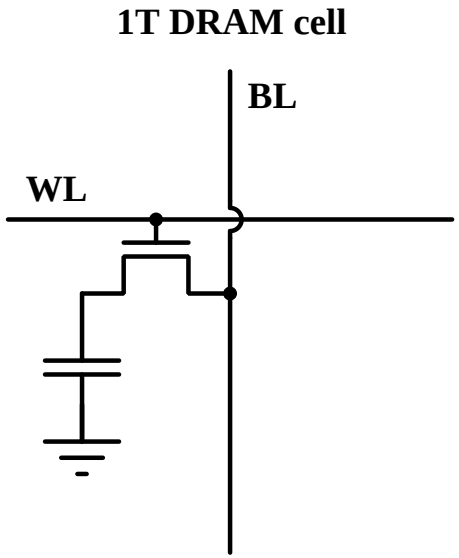
Presence or absence of a voltage change is detected by the sense amplifier and driven to the output as in the case of an SRAM.

Destructive Read in a DRAM cell



- During a read cycle, the storage capacitor is connected to the bit line in the entire word line.
- Since the storage capacitor is much smaller than the bit line capacitor, it gets charged to ‘1’ during charge sharing.
- All cells which had a ‘0’ stored have their data destroyed!
- So the Read operation takes place on the entire row, and the read data needs to be written back to the entire row.
- Thus a read cycle is always followed by an internal write operation for the whole row.
- This also refreshes the data in the entire row, restoring any leaked charges.

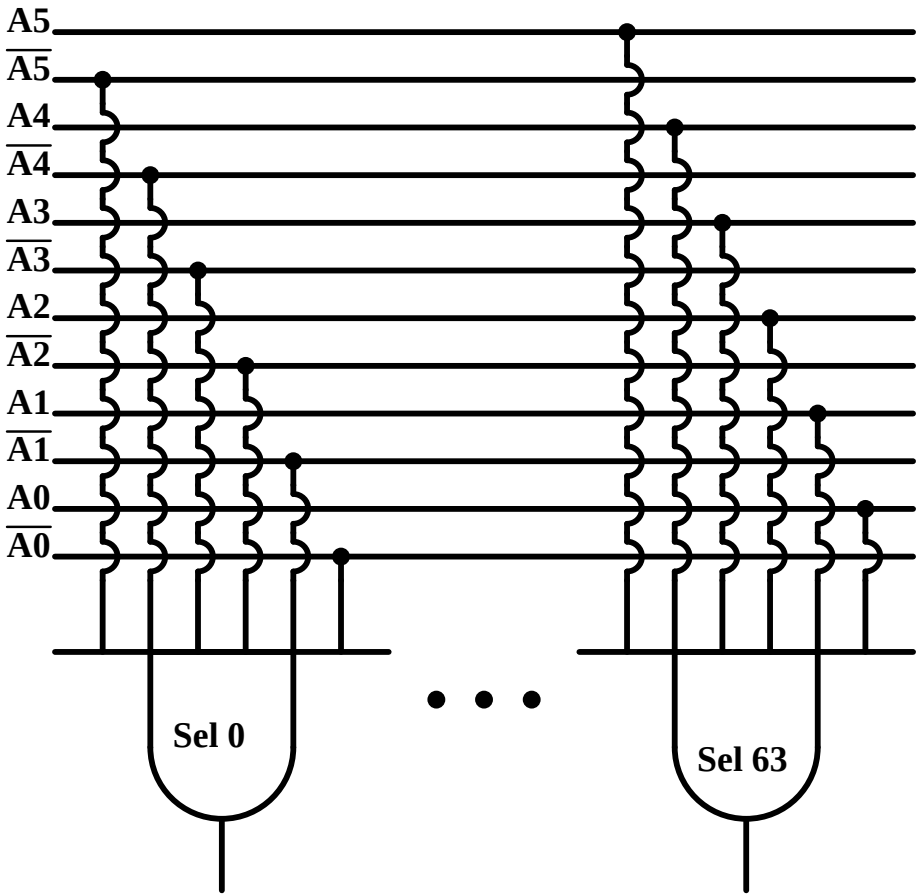
Write Cycle in a DRAM cell



- During a write cycle, the bit line is driven to the data value in the selected column.
- When the word select signal comes, the data is copied to the storage capacitor.
- The columns which are not selected by the current address go through a read cycle.
- This includes the refresh operation.
- So the data is refreshed for the entire word line whenever it is accessed for a read or a write operation.
- Banks of DRAM memories need a controller, which periodically performs a dummy read on every row in order to refresh the data stored in the memory.
- Modern DRAM chips have refresh circuitry on chip, which internally do the refresh operation if some word lines are not being accessed. These memories are also called SDRAMs.

Address decoding in Memories

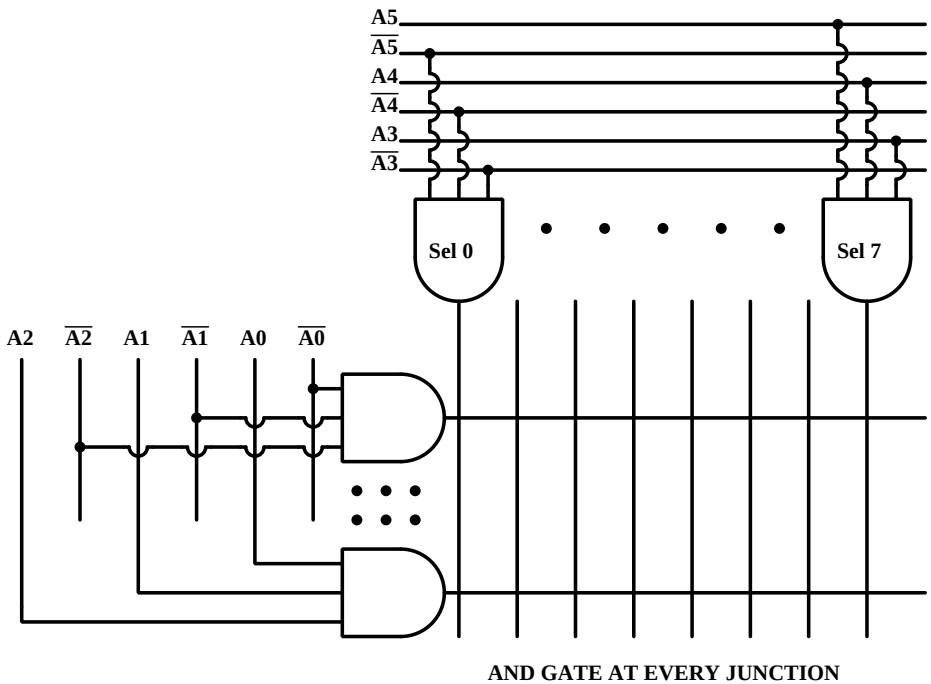
A one step decoder for 6 bit row or column address is shown below.



This kind of decoding is not practical.

Address decoding in Memories

A more practical approach decodes different groups of memory separately. We separately decode 3 bits each in this example and then combine the select outputs using AND gates.



Semi-custom Design

- Some applications can afford to compromise on speed, power and complexity in order to achieve a quick design and fabrication time and lower costs.
- These use a design style known as semi-custom design.
- In semi-custom design, part of the design and fabrication is already done for us. We take this pre-fabricated template and customize it to perform the functions that our VLSI needs to perform.
- This approach has several advantages. The prefabricated part can be processed and kept ready for customization.
- Then the time to take an application to the market is defined only by the remaining processing which is necessary after customization.

Semi-custom Design

- Different applications can use the same pre-fabricated template.
- While each application may have a relatively small market, the template will be made in very large numbers, making it economically competitive.
- Obviously, the pre-fabricated template itself is not customized for a particular final circuit. So it will not be an optimal design for a given application.
- However, most applications do not require absolutely the best performance.
- In such cases, the time to market and cost can be drastically reduced by using semi-custom design.

Field Programmable Gate Arrays

- From a time to market point of view, it is preferable to have the customization done as late as possible.
- However, then a higher fraction of the design and fabrication cycle is non-specific to the actual design.
- In Field Programmable Gate Array (FPGA) based design, customization is done *after* the product has actually been delivered to the end user.
- That is what the term Field Programmable refers to.
- The actual VLSI template which can support this late customization is quite complex.
- The area of the chip is much higher and the speed much lower compared to what could have been achieved by a custom designed circuit.

Field Programmable Gate Arrays

- System clock speeds possible with FPGAs are of the order of hundreds of MHz, whereas custom circuits can go to clock speeds of about 4 GHz.
- To implement a given function, the silicon area consumed by an FPGA may be an order of magnitude higher than the area consumed by custom designed circuit performing the same function.
- However, since many applications can use the same FPGA, the FPGA chip is produced in very large quantities, which provides economies of scale.
- Thus, for designs which are not produced in very large quantities, FPGA based design can be the most cost effective solution, in spite of the overhead in complexity, power consumption and delay.

Semi-Custom Design Procedure

There are two components of Semi-custom design technique.

1. Customization techniques which will be used for adapting the “fabric” to implement the final application on the given template.
2. The design of the basic template or the “fabric” which can be customized as late as possible and which can be used by a large variety of applications.

During fabrication, transistors are defined much before the interconnects.
Therefore, to be able to customize the circuit as late as possible, customization of a pre-fabricated template is commonly done by changing the interconnects.

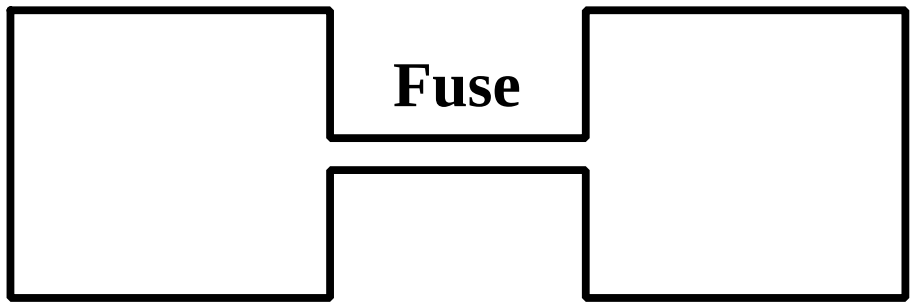
Customizing Interconnects

Customization of interconnects is done by placing programmable interconnect devices at appropriate points in the pre-fabricated template. These include:

- Fuses: Here the connection is a short by default and can be converted to an open by blowing a fuse.
- Anti-fuses: Here the connection is open by default, but can be converted to a short by breaking down an insulator.
- transistor based interconnects: Here a connection is made or broken using transistor switches. The state of the transistor is defined by a memory.

Use of Fuses to Customize Interconnect

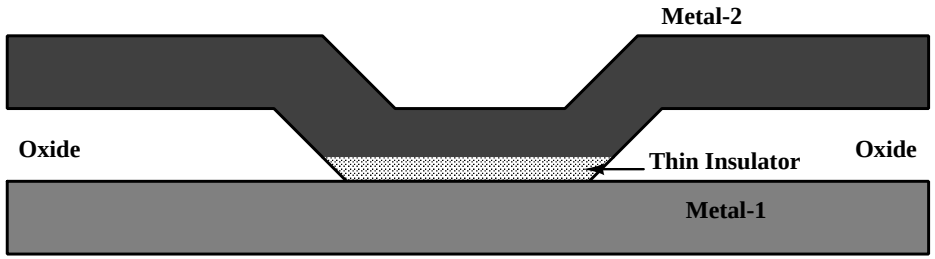
Customization of interconnects can be done through programmable removal of an existing connection. These work like fuses in electrical circuits.



- The connection to be customized is connected in the template through a narrow neck like structure.
- The narrow part of the interconnect is capable of passing normal operating currents of the circuit.
- However, during customization, one can pass a pulse of heavy current through this, which ‘blows’ the fuse and disconnects the wires leading up to the fuse.

Use of Anti-Fuses to Customize Interconnect

In these structures, there is no connection between the programming nodes in the un-programmed state of the interconnect. The current path is interrupted through a thin layer of insulator.



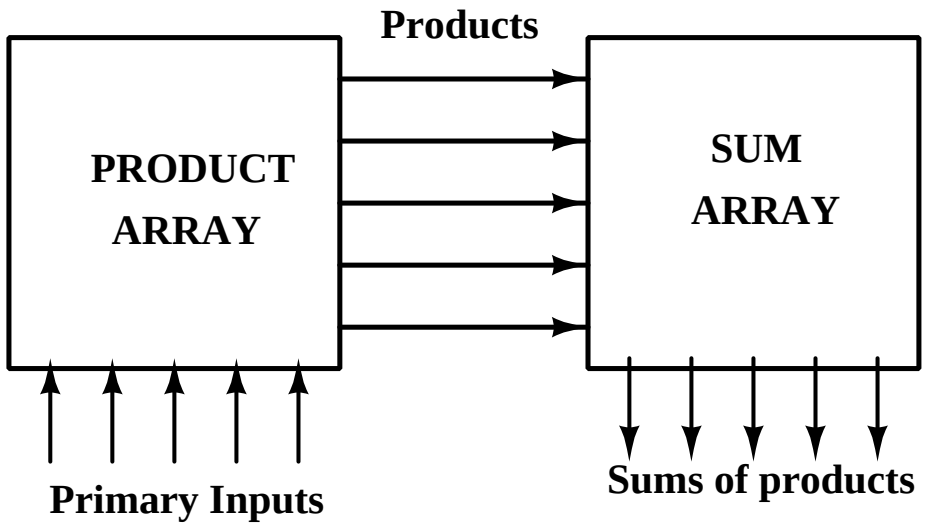
By applying a high voltage pulse, this insulator can be broken down and a short created across it.

This action is opposite to that of a fuse. Therefore such structures are called anti-fuses.

Re-configurable Logic with Programmable Logic Arrays

Logic functions can be expressed in a generic sum of products form.

- We need a circuit which can be re-configured to implement any logic expressed as a sum of products.
- One way is to use a customisable array which produces different product terms and another customisable array which sums the products so produced.



Use of Pseudo-nMOS for Programmable Logic Arrays

- We need an architecture where transistor geometries are not changed with logic and only interconnect needs to be programmed.
- CMOS logic is not convenient for implementing this architecture.
- This is because simultaneous configuration of the p channel pull up network and the n channel pull down network will be needed.
- Pseudo NMOS gates are more suitable in this case, because the pull up is just a single grounded gate pMOS whose geometry remains the same for all logic.

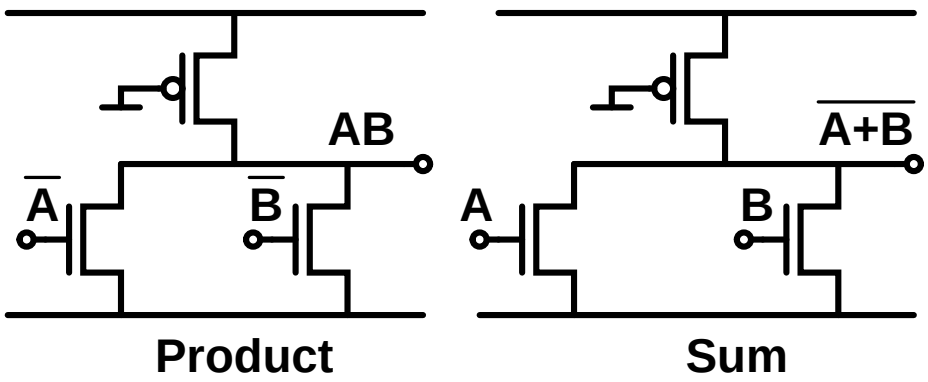
Use of Pseudo-nMOS for Programmable Logic Arrays

- Pseudo NMOS circuits are ratioed.
- Implementing sums is not a problem, since this is done with NOR type gates.
- Transistor geometry remains the same in NOR gates irrespective of how many transistors are put in parallel.
- However, implementing products presents a problem, since NAND gates connect nMOS transistors in series.
- The geometry of series connected pull down devices depends on the number of devices connected in series and hence, on the specific logic being implemented.

Use of Pseudo-nMOS for Programmable Logic Arrays

How to implement the product function without changing transistor geometry?

We can use the expression :
$A \cdot B = \overline{\overline{A} + \overline{B}}$.
Now the product of A and B can be implemented as the NOR of $\overline{A}$ and $\overline{B}$, which does not use series connected transistors.



By adding inverters at the input and output as required, we can implement products or sums using only NOR type of circuits, which use constant geometry NMOS transistors in parallel.

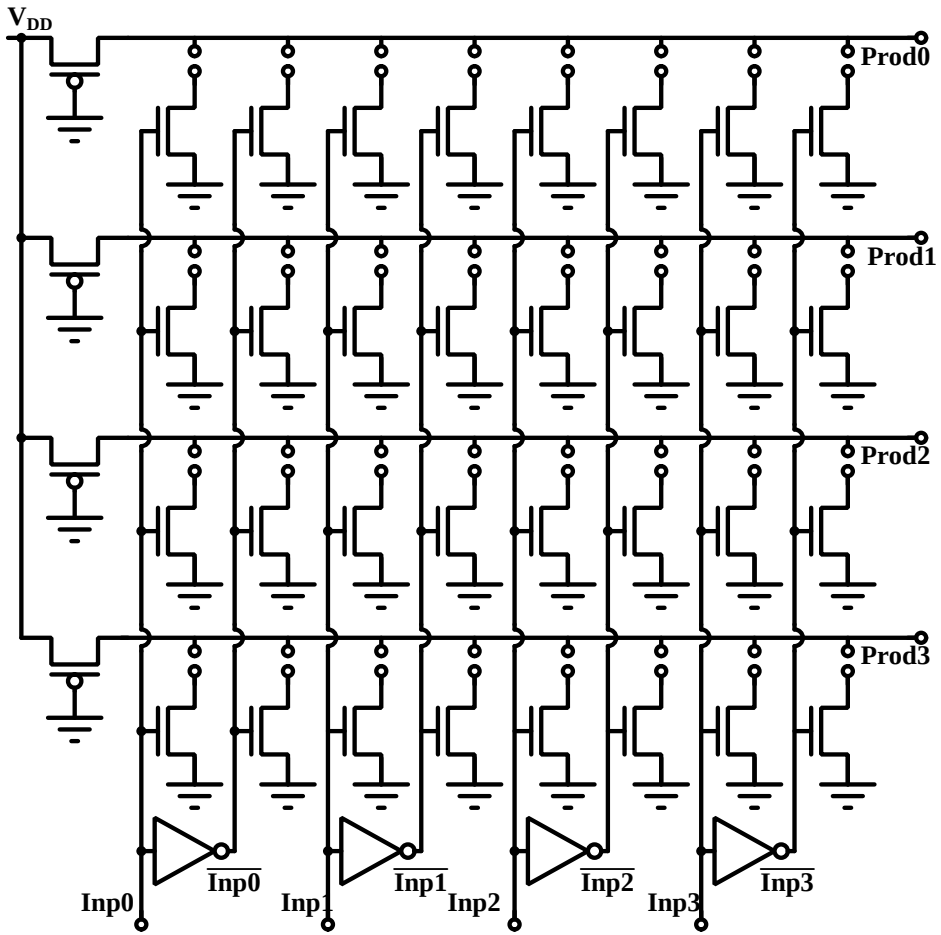
Programmable Logic: Product Array

Suppose we want to generate p distinct products of n inputs.

- We generate complements of all inputs.
- We place pull down nMOS transistors in a matrix of $2n$ columns and p rows.
- Each row is pulled up by a pMOS transistor with grounded gate.
- The row corresponds to the output of a NOR circuit which may have up to n pull down nMOS transistors in parallel.
- Each column is driven by an input or its complement. So there are $2n$ columns.

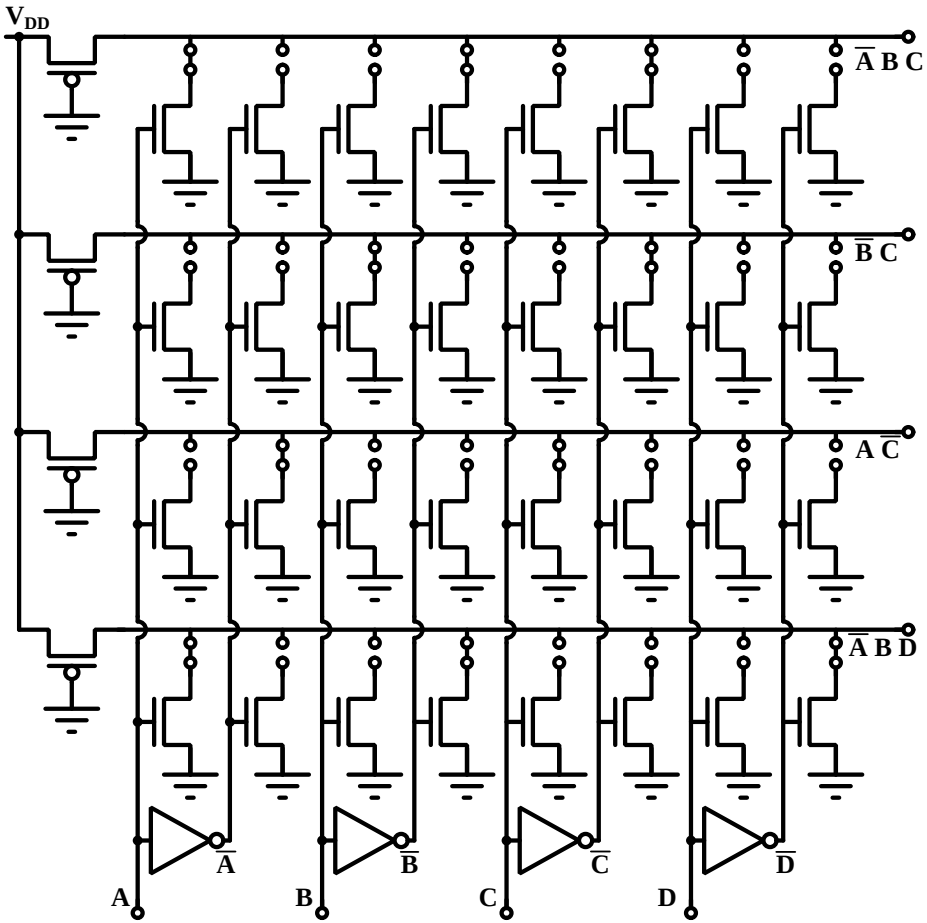
Programmable Logic: Product Array

- Each row generates a distinct product.
- By connecting links selectively at drains of nMOS transistors in any row, chosen transistors can be included in the NOR configuration.
- Connecting a transistor includes the complement of the input driving its gate in the product term.



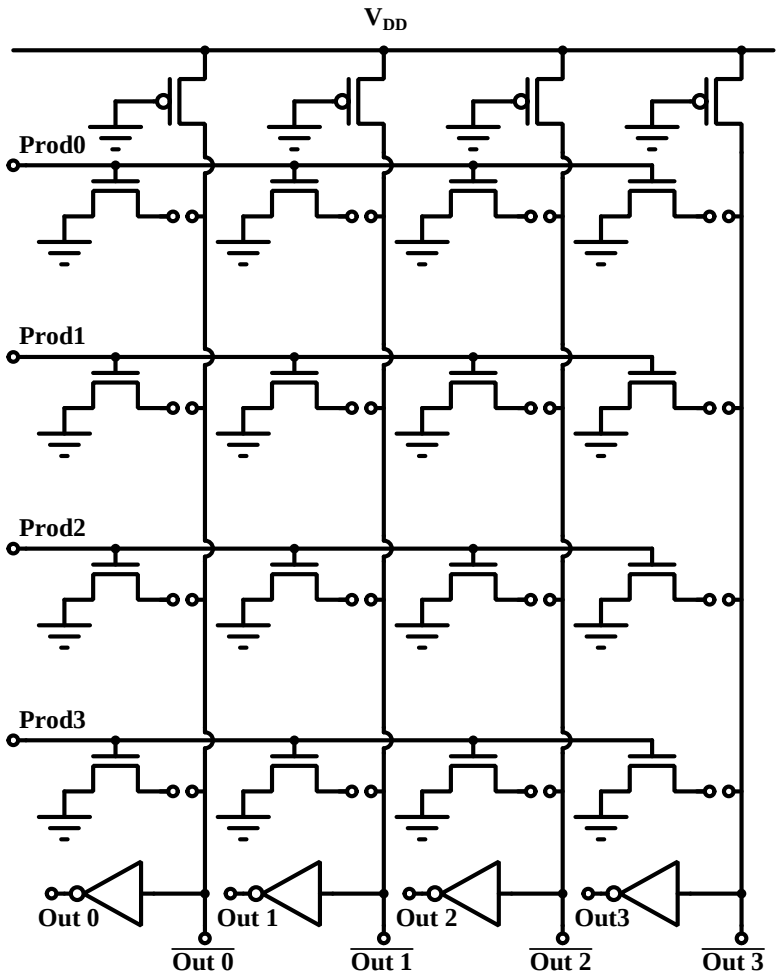
Programmable Logic: Product Array

- In the example shown on the right, we have generated products $\overline{A} \cdot B \cdot C$, $\overline{B} \cdot C$, $A\overline{C}$ and $\overline{A} \cdot B \cdot D$.
- To generate $\overline{A} \cdot B \cdot C$, links from drains of transistors with gates connected to the columns with $A, \overline{B}$ and $\overline{C}$ are connected to the top product row.
- Other products are generated similarly.

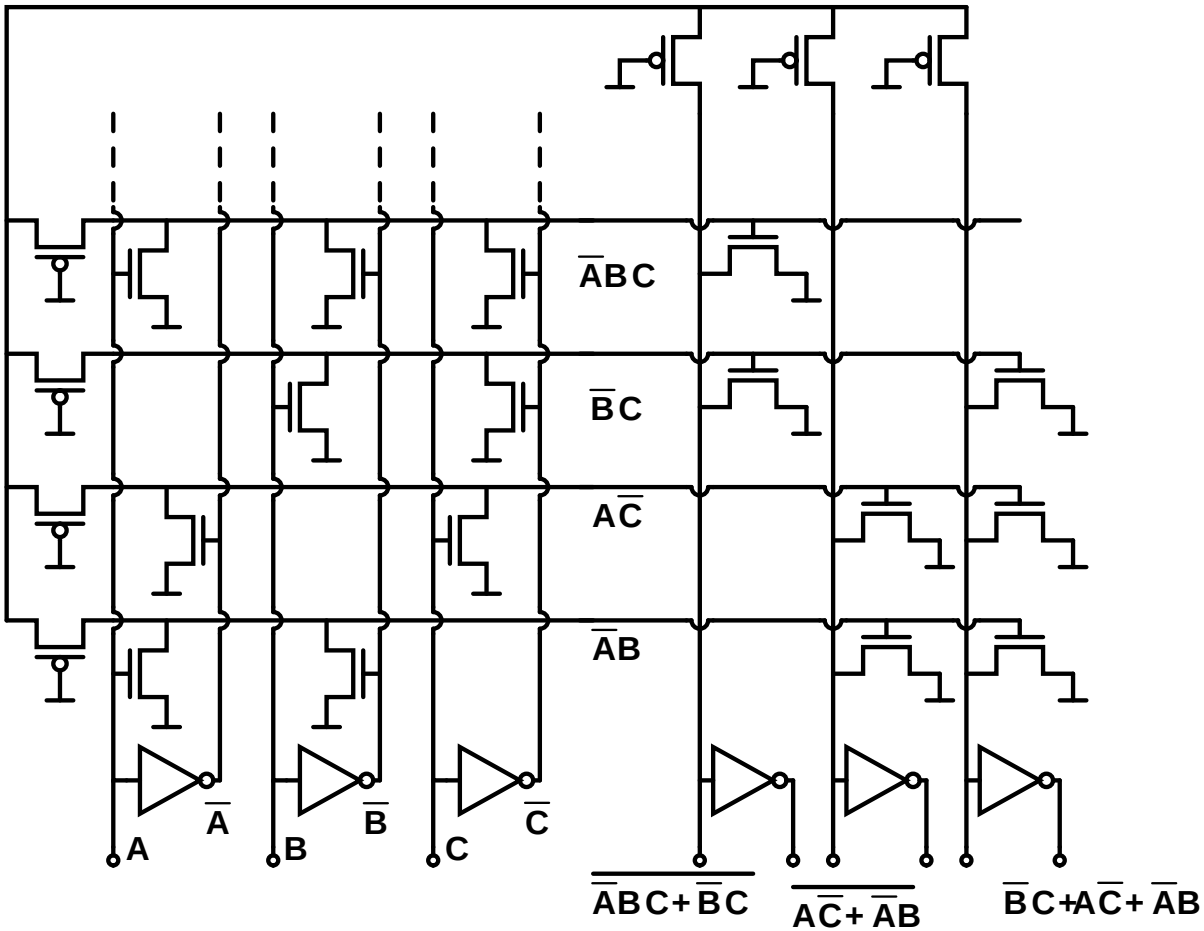


Programmable Logic: Sum Array

- The circuit on the right can produce sums of selected product terms.
- By connecting links at drains of nMOS transistors in a particular column, chosen products can be included in a sum output.
- Several columns can be used to generate different sums of products.
- Outputs are NORs of products. Inverters convert these to ORs of products.



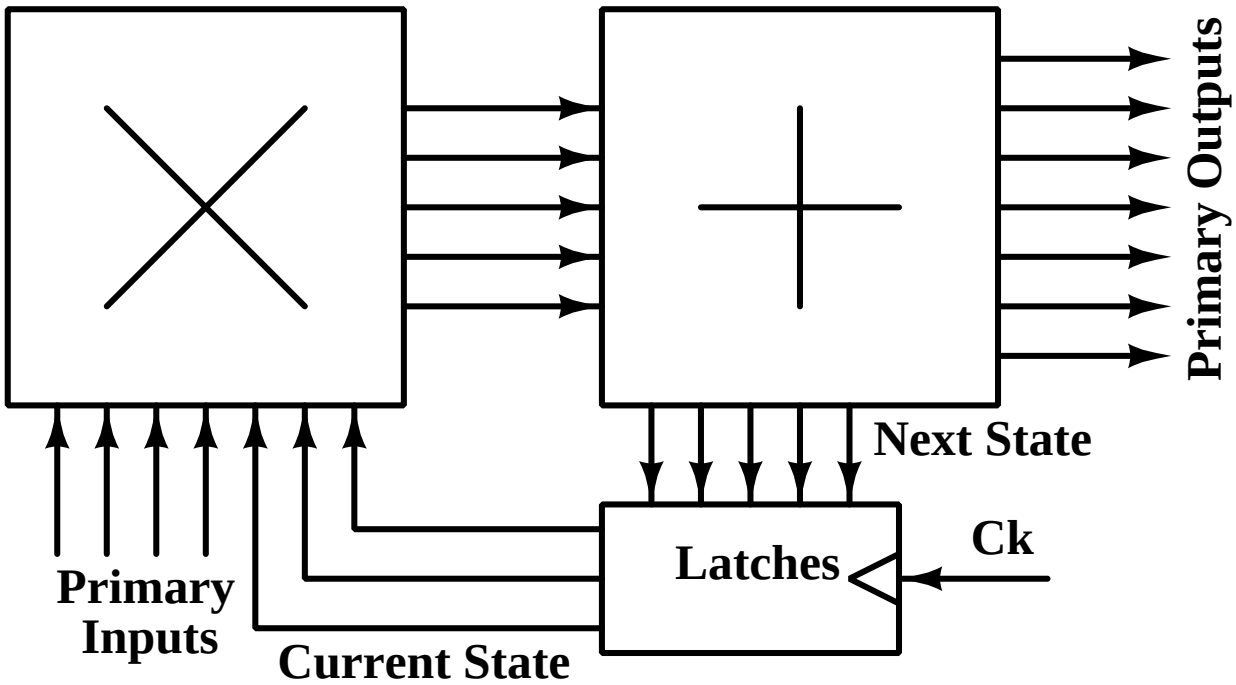
Programmable Logic Arrays



Implementation of Finite State Machines

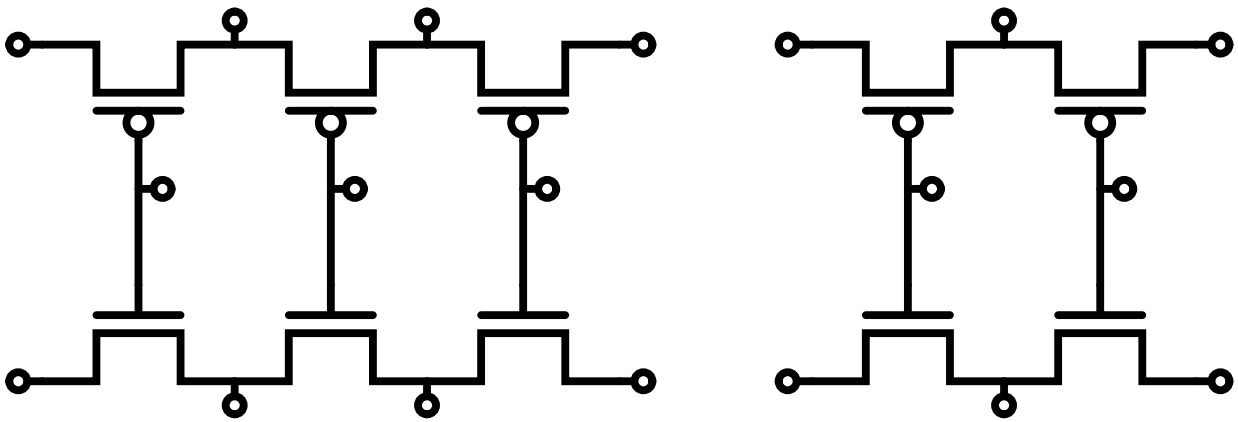
- A finite state machine has the following components:
  1. Storage elements to store the current state,
  2. Random logic to compute the next state as a function of current state and current inputs, and
  3. Random logic to compute the current output as a function of current state and optionally, the current inputs.
- By adding latches at the output of the PLA, we can provide for all of these.
- Output from latches is fed back to the product array.
- The PLA computes both the next state and current outputs.

Implementation of Finite State Machines



Sea of Gates

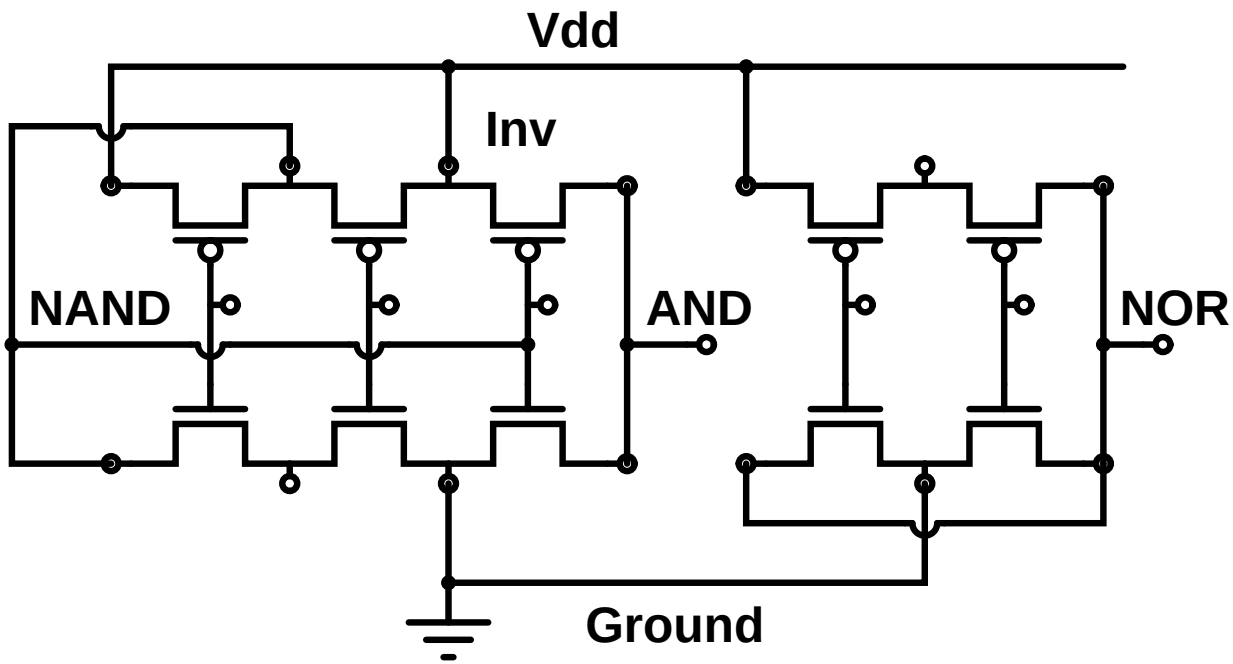
- This style uses CMOS logic as its base.
- In CMOS logic, each input goes to an nMOS as well as a pMOS.
- Transistors are pre-placed in a matrix of cells like the one shown below. Interconnects determine what kind of logic will be implemented.



Both n and p channel transistors are in series here!
How do we construct regular CMOS logic gates using these?

Logic from Sea of Gates

Actually, by wiring the apparently series connected devices, we can convert them to series or parallel as required.

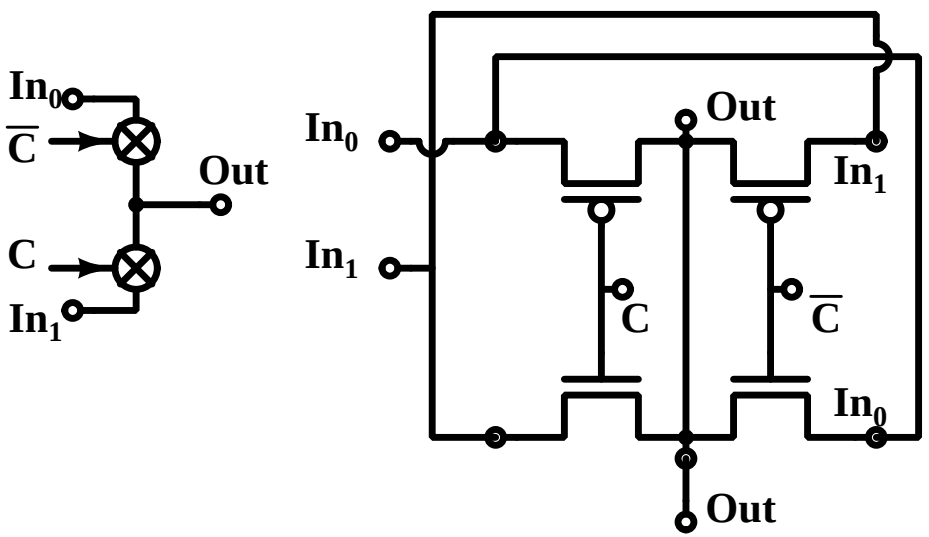


The example above shows a NAND gate, whose output is fed to an inverter to form the AND function. The structure on the right forms a NOR gate.

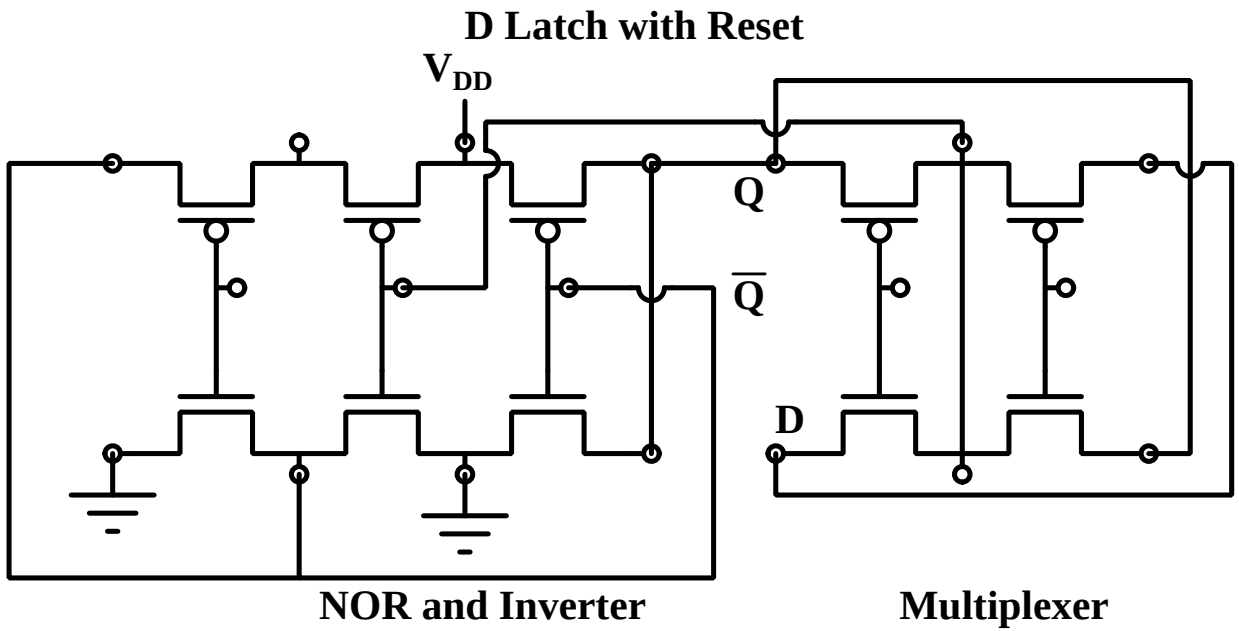
Sea of Gates Transmission Gate

- The sea of gates template makes use of the fact that all inputs go to an nMOS as well as to a pMOS in CMOS style gates.
- What about structures which don't follow this rule? For example, in a transmission gate, the nMOS and pMOS transistors are driven by complementary signals.

- Transmission gates are often used in pairs with one or the other being on. (Inputs should not be left floating in static CMOS design).
- The pair can be implemented as shown on the right, coupling diagonally opposite transistors in a 2-pair.



Sea of Gates implementation of D latch

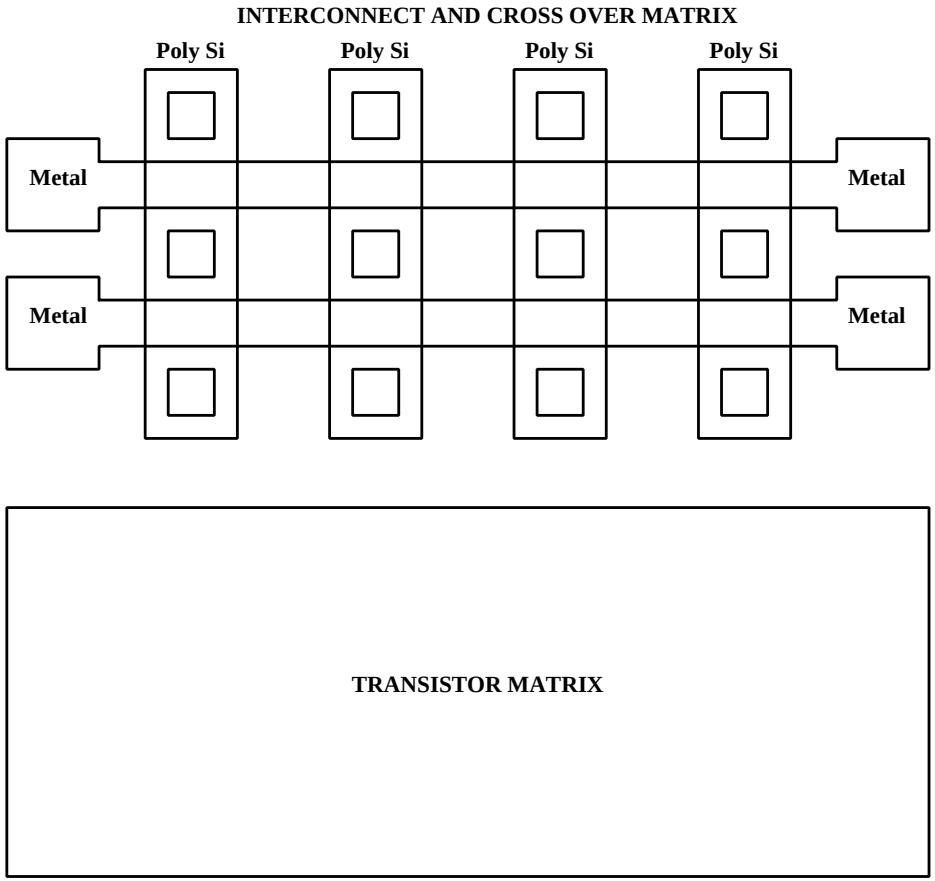


Two such latches can be connected in master-slave configuration to form an edge sensitive D flip-flop.

Interconnect Channels in Semi-Custom Logic

Pre-fabricated interconnect channels alternate with transistor matrices in the fabric of a semi-custom chip.

- The exact shape and composition of these interconnect channels varies from design to design.
- Typically, these will include facilities for local as well as global interconnects.
- These are optimized to provide a sufficient interconnect capacity without wasting too much area.



Using Memories as Logic

- A logic function can be represented by a truth table. This can be stored in memory. Inputs to the logic function act as address pins.
- Pre-computed value of the logic function are stored at the address represented by its input values.
- For any input combination, the address represented by these is looked up and the stored value presented as the output.
- For example, any logic function of 5 inputs can be implemented by a 32×1 bit memory.
- If the same 32 bit memory is organized as 16×2, it can store two logic functions of up to 4 inputs.

Evolution of FPGAs

Different types of semi-custom design chips have been introduced with various names. Here is a list:

PLAs: These were the first semi-custom devices to be introduced in the market by Philips in the early 1970's. These are used to implement generic sums of products and have a programmable AND plane as well as a programmable OR plane.
Because both AND and OR arrays are programmable in PLAs, these are rather slow.

PALs: A modification of PLAs in which the product array is programmable, but the OR plane is fixed were introduced as Programmable Array Logic or PAL. To cover for the lack of flexibility due to the OR array being fixed, these were introduced in different size combinations and multiple devices were used for different functions.

Evolution of FPGAs

CPLDs As technology progressed, it was possible to put multiple devices on the same chip. Combinations of PALs and PLAs were introduced with programmable interconnect as complex programmable logic devices or CPLDs. Altera was one of the first companies to introduce CPLDs commercially. CPLDs were a huge success and multiple companies introduced CPLDs of different architectures.

Sea of Gates In parallel with field programmable devices, mask programmable devices were manufactured. In these, one (or more) levels of metallization could be customized at the manufacturing site, over a “sea” of pre-fabricated devices. These provided circuits with better performance, but with less flexibility as programming could not be done at the user site.

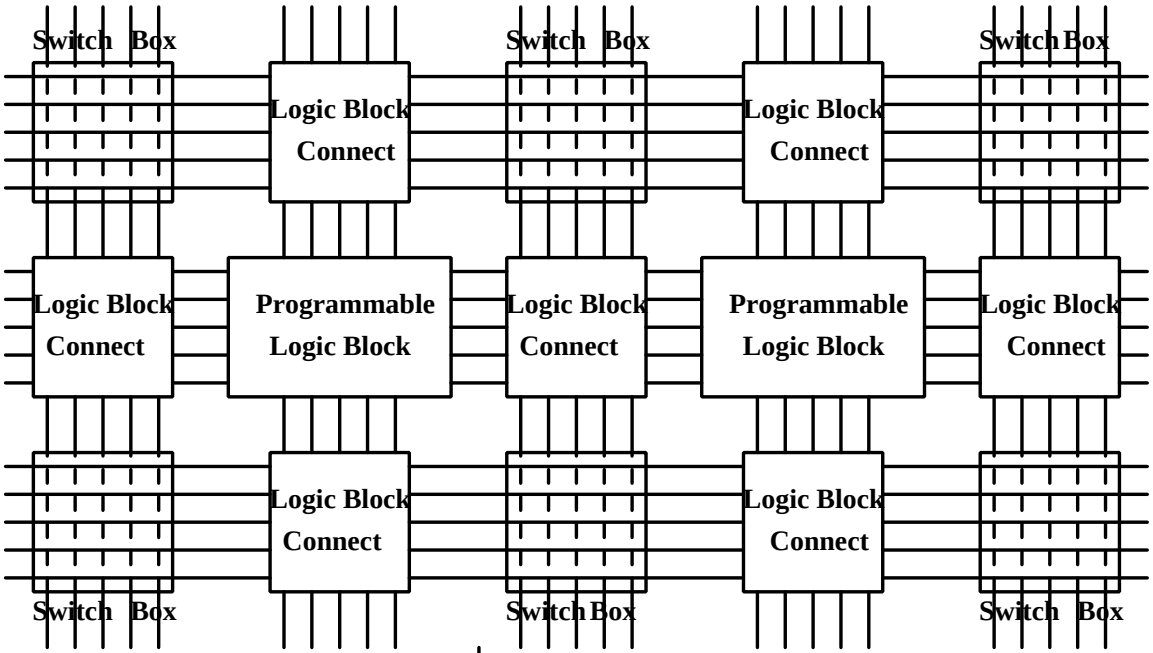
Evolution of FPGAs

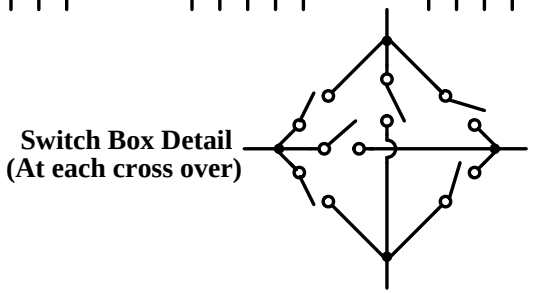
FPGAs Field Programmable Gate Arrays borrow features from sea of gates as well as CPLDs. Instead of a “sea” of transistors, these have a “sea” or a repetitive array of combinations of logic elements and memories.
In addition, these include a programming infrastructure for interconnects like CPLDs.

On the periphery of these chips, special Input Output devices are fabricated which help in establishing fast interconnection with the external world.

FPGA Architectures

A field Programmable array allows the logic functions as well as the interconnect to be programmed.





Transistors driven by SRAM can be used to program interconnects.

Current FPGA Architecture

Apart from logic blocks and interconnect fabric, modern FPGA's contain other useful structures, such as

- Block RAM,
- Processor cores (Power PC on Xilinx Vertex family),
- Fast multipliers
- I-O Blocks

Types of Reconfigurable Logic

- A typical board, used for implementing a variety of applications will have a micro-controller and a few programmable devices for providing dedicated functions and glue logic.
- This is typically configured once at power on, and then left alone to perform its functions. This is called “static re-configurability”.
- But we may want to re-configure a structure while it is in operation! For example, one can imagine the design of digital filter, which adapts itself during operation itself depending on inputs. This kind of re-configurability is called “dynamic re-configurability”.
- Obviously, dynamic configurability requires that the dead time during re-configuration should be minimized.

Fine and Coarse Grain Reconfigurability

- There is a trade-off between flexibility of re-configuration and the time taken to re-configure.
- In fine grain re-configuration, we can re-program every gate of the design. This is the case for current FPGA and CPLD devices. This gives very good design flexibility, but takes a long time to re-configure.
- In coarse grain re-configuration, relatively large functional blocks are re-configured. For example, by re-connecting a collection of shifters and adders, we can generate any combination of multipliers and adders. The effort to re-configure is smaller, because we do not alter the inner design of shifters and adders. This is compatible with dynamic re-configuration.

Use of Dynamic Re-configurability

Using coarse grain re-configurability, it becomes possible to dynamically change a circuit, *during* its operation.
This has obvious advantages in adaptive circuits.
We normally do not want to re-configure the whole circuit. Indeed the control signals for doing the re-configuration will be generated by a circuit which remains unchanged.
Therefore dynamic re-configurability requires circuits which can be partially re-programmed. Many FPGA are beginning to promise this feature.