SEQUENTIAL LOGIC/ SEQUENTIAL LOGIC CIRCUITS:

In sequential logic circuit, the output depends not only on the current input values but also on the preceding input values. In other words it can be said that a sequential circuit has memory as it has data from the past.

Examples: Latches, flip flops, shifter registers, counters, checkers etc.

CLASSIFICATION OF MEMORY:

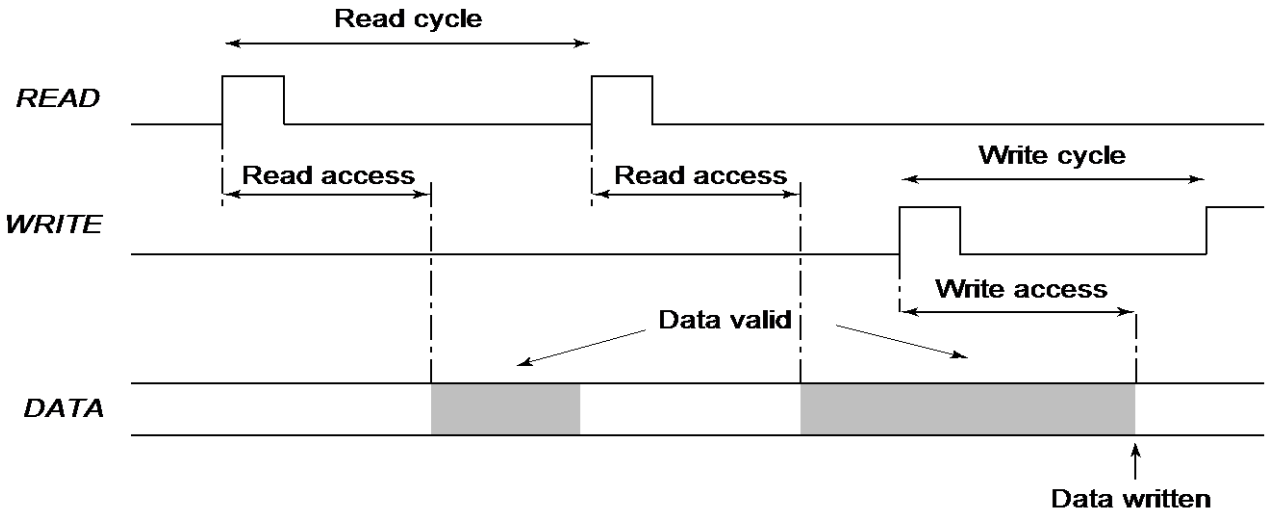
Memories are used for storing large data in digital system. The memory requirement is dependent on the application.

Basically memory is made of transistor cells, for large storage dense number of transistor integration is done and to reduce the number of transistors different MOS technologies are been used.

Memory is classified into three categories:

1. Read- Write Memory (RWM)
2. Non- volatile Read Write Memory (NVRWM)
3. Read- Only Memory (ROM)

RWM		NVRWM	ROM
Random Access	**Non- Random Access**	EPROM E^2PROM FLASH	Mask- programmed programmable (PROM)
SRAM DRAM	FIFO LIFO Shift register CAM		



MEMORY TIMING

READ- WRITE MEMORY:

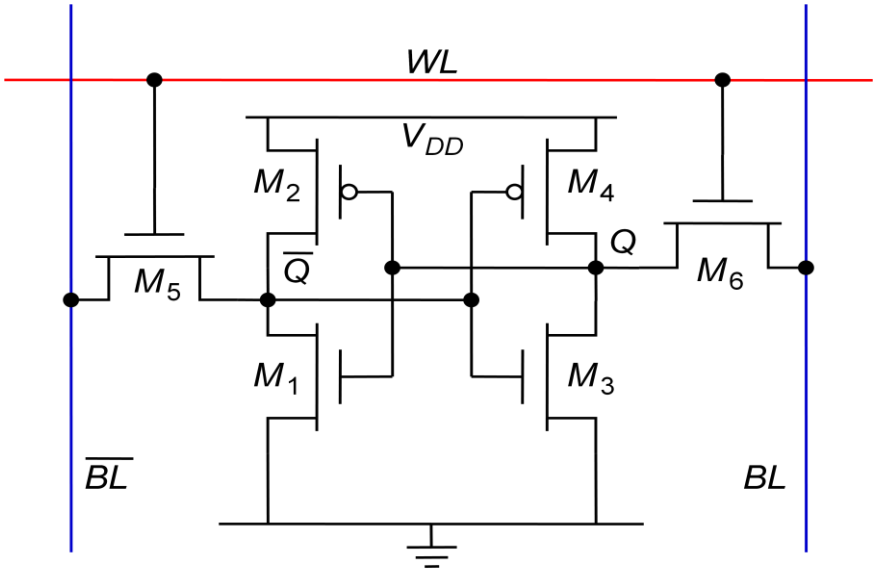
As prerequisite from the classification of memory RWM comprises of **Static Random Access Memory (SRAM)** and **Dynamic Random Access Memory (DRAM)**.

Static Random Access Memory (SRAM) (also called as 6T SRAM):

In SRAM the data is stored as long as the supply is applied. It is fast and differential.

The SRAM consists of 6 transistors (6T) i.e. a combination of two CMOS inverters comprising 4T (M_1, M_2, M_3 and M_4) and two nMOS pass transistors (M_5 and M_6), totally constituting 6T.

In SRAM access to each cell is enabled by the word line, which replaces the clock and controls the two pass transistors M_5 and M_6, shared between the read and write operation.



SRAM

Operation:

The SRAM should be sized as small as possible to achieve high memory densities.

Performing Write Operation:

To write the data into the cell, again the word line is enabled, the data to be written is made available in the bit lines and the data is stored in the latch.

Performing Read Operation:

To read the data from the cell, word line is enabled, this makes transistors M_5 and M_6 ON. Hence the stored data is available in both the true and complemented form in the bit line and complement bit line respectively.

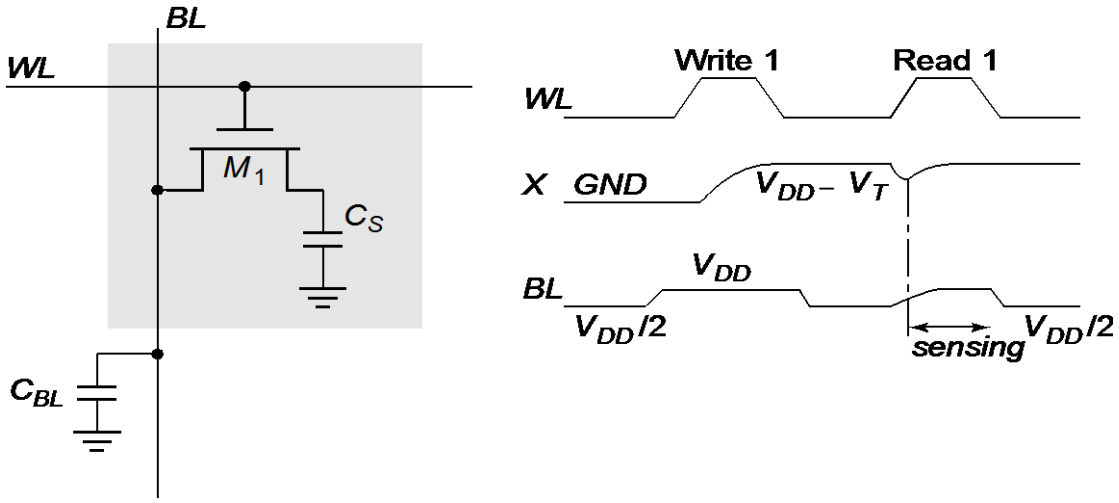
Dynamic Random Access Memory (DRAM):

In DRAM periodic refreshment is required, these are small consisting of IT, 3T or 4T cells. DRAMs are slower and single ended. In DRAM the data is stored in the parasitic capacitance which discharges with time.

The various types of DRAM are:

1. 1Transistor/ 1T DRAM
2. 3T DRAM
3. 4T DRAM

1T DRAM:



1T DRAM AND ITS SIGNAL WAVEFORMS DURING READ AND WRITE OPERATION

The 1T DRAM consists of one transistor (M_1) which is connected to the word line (WL) and bit line (BL) along with a storage capacitor.

Operation:

Performing Write Operation:

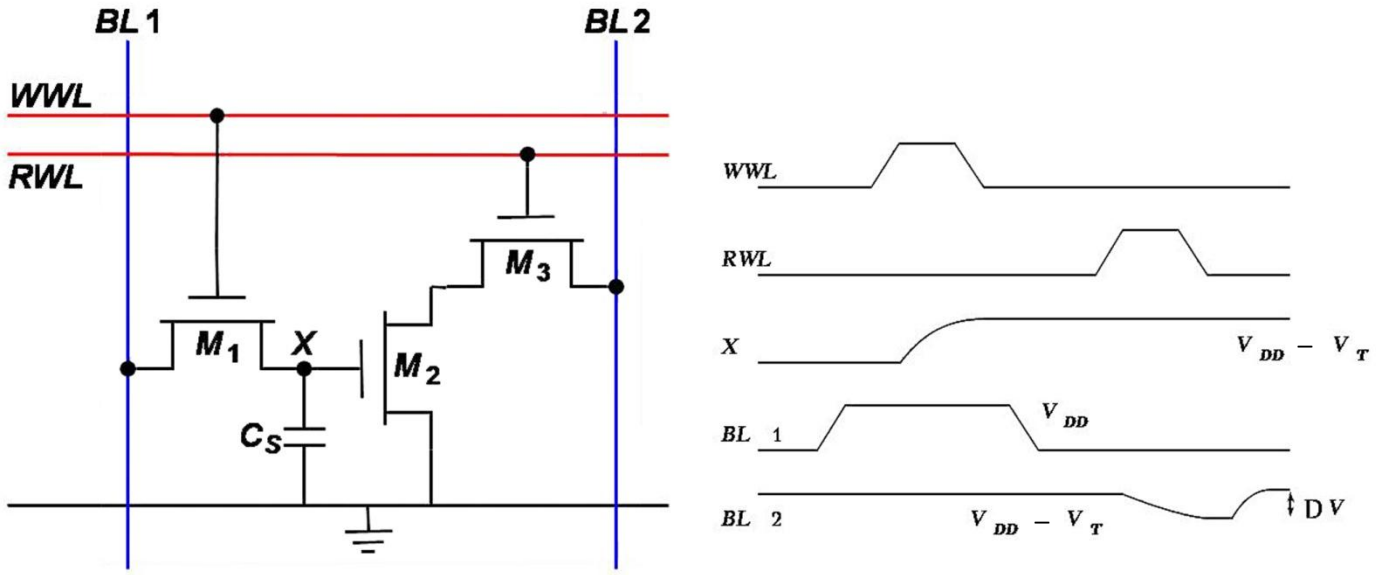
While performing the write operation the word line is enabled and the data from the bit line is stored in the capacitor C_S.

Performing Read Operation:

While performing the read operation the word line is enabled and stored data becomes available at the bit line. The charge distribution takes place between the bit line and storage capacitor.

- The 1T DRAM is mostly used in high density DRAM architecture
- The voltage swing is small, typically around 250 mV and it requires a sense amplifier for each bit line, due to charge distribution read out.

3T DRAM:



3T DRAM AND ITS SIGNAL WAVEFORMS DURING READ AND WRITE OPERATION

The three transistor DRAM consists of three transistors M_1, M_2 and M_3 respectively as shown. Transistor M_1 is connected to the write word line (WWL), the transistor M_3 is connected to the Read Word line (RWL) and the transistors M_2 is used for storing the binary data in association with its storage capacitance C_S.

Operation:

Performing Write Operation:

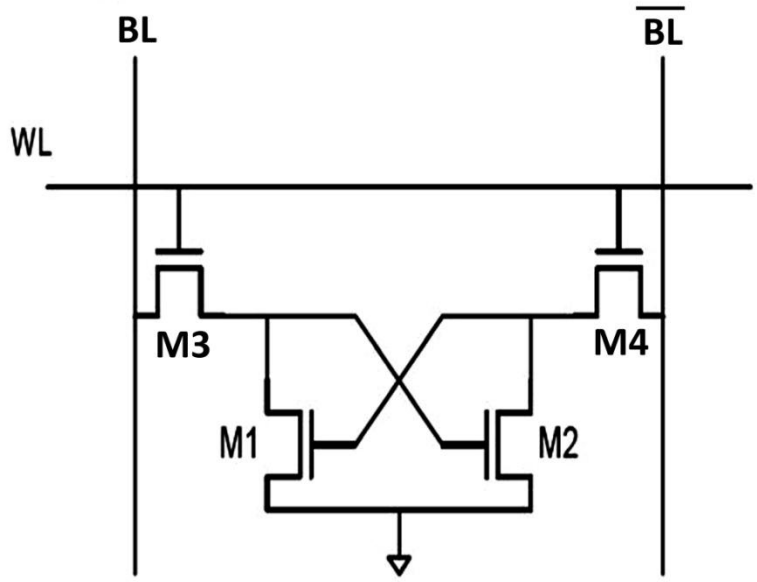
While performing the write operation, the writing word line (WWL) is enabled, the logic from the write bit line is passed to the parasitic storage capacitance C_S.

Performing Read Operation:

While performing the read operation, the read word line in enabled (RWL) and the complement of the stored data becomes available in the read bit line.

- In contrast to SRAM, in 3T DRAM no constraints exist on the device ratios.
- In contrast to other DRAM cells, reading the 3T cell contents is non-destructive i.e. the data value stored in the cell is not affected by a read.
- Here no special process steps are needed. The storage capacitance is nothing more than the gate capacitance of the readout device, this is in contrast to other DRAM cells.
- The 3T DRAM is attractively used in embedded memory applications.
- The value stored on the storage node X when writing a 1 equals to $V_{WWL} - V_{th}$. This threshold loss reduces the current flowing through M_2 during a read operation and increases the read access time. To prevent this, some designs bootstrap the word line voltage i.e. raise V_{WWL} to a value higher than V_{DD}.

4T DRAM:



4T DRAM

The 4T DRAM consists of four transistors M_1, M_2, M_3 and M_4 respectively with connections to word and bit lines as shown.

Here the binary data is stored in the parasitic capacitances present between the transistors M_1, M_3 and M_2, M_4 with respect to ground. The data is written in complemented form by enabling the word line. As there is no restoring path from V_{DD} to these capacitances, the stored charge is lost. So to retain the logic level, the capacitors must be refreshed periodically.

During read operation, the word line (WL) is enabled and the stored data becomes available at the bit lines (BLs) both in the true and complemented form respectively.

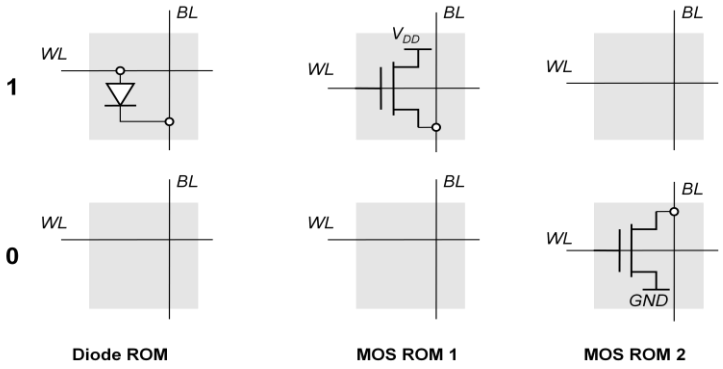
READ ONLY MEMORY (ROM):

Using ROM several large number of potential applications can be accomplished. Programs for processors with fixed applications such as washing machines, calculators and gaming machines- once developed and debugged need only reading.

Fixing the contents at manufacturing time leads to small and fast implementations.

With the fact that ROM cells are permanently fixed considerably simplifies the design. The cell should be so that a 0 or 1 is present to the bit line upon activation of its word line.

The different approaches for implementing 1 and 0 ROM cells are shown below.

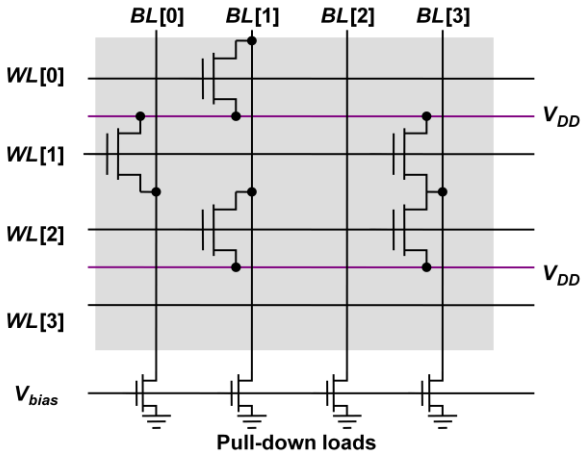


DIFFERENT APPROACHES FOR IMPLEMENTING 1 AND 0 ROM CELLS

Using the ROM cells, ROM arrays can be developed which can be designed as:

1. OR based ROM Memory Design
2. NOR based ROM Memory Design
3. NAND based ROM Memory Design

OR BASED ROM MEMORY DESIGN:

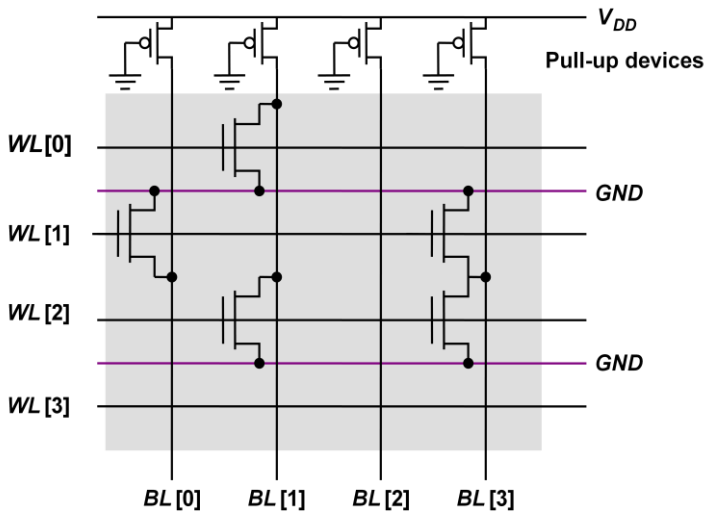


A 4 × 4 OR ROM CELL ARRAY

The figure shows a 4 × 4 OR RAM cell array. Here the values of the data stored at addresses 1, 2 and 3 as shown in figure are determined.

Here the absence of transistors between the word lines and bit lines means that logic 1 is stored and the 0 cell is realized by providing a MOS device between the bit lines and ground. Applying a high voltage on the word lines turns on the device, which in turn pulls down the bit line to ground.

NOR BASED ROM MEMORY DESIGN:



A 4 × 4 NOR ROM CELL ARRAY

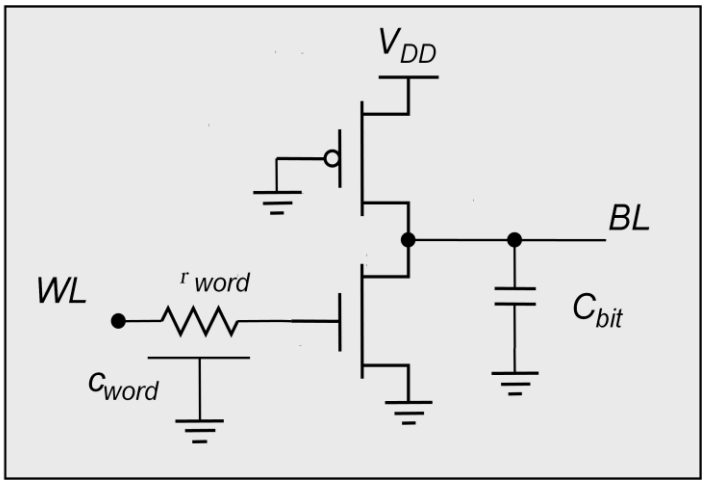
The above figure shows a 4 × 4 NOR ROM cell array. The values of the data stored at addresses 0, 1, 2 and 3 are determined. The combination of a bit line, PMOS pull- up and NMOS pull- downs constitutes a pseudo- NMOS NOR gate with the word line as inputs. Therefore, an N × M ROM memory can be considered as a combination of M NOR gates with at most N inputs (for a fully populated column) are called a NOR ROM.

Under normal operating conditions only one of the word line goes high, and at most one of the pull- down devices is turned on. This raises the issues regarding the sizing of both the cell and pull- up transistors.

- To keep the cell size and the bit line capacitance small, the pull- down device should be kept as close as possible to minimum size and
- The resistance of the pull- up device must be larger than the pull- down resistance to ensure an adequate low level.

Equivalent model of NOR based memory design:

The equivalent shown is appropriate for the analysis of the word and bit line delay of the NOR ROM. The word line is best modelled as a distributed RC line since it is implemented in polysilicon with a relatively high sheet resistance.



EQUIVALENT MODEL FOR NOR BASED ROM

The wire line parasitics consists of wire capacitance and gate capacitance, the wire resistance by the polysilicon and the bit line parasitic consists of resistance not dominant (metal) and drain, gate- drain capacitance.

The bit line is implemented in Aluminium and the resistance of the line only comes into play for very long lines. It is reasonable to assume that here a purely capacitive model is adequate and that all capacitive loads connected to the wire can be lumped into a single element.

- The word line parasitics are:

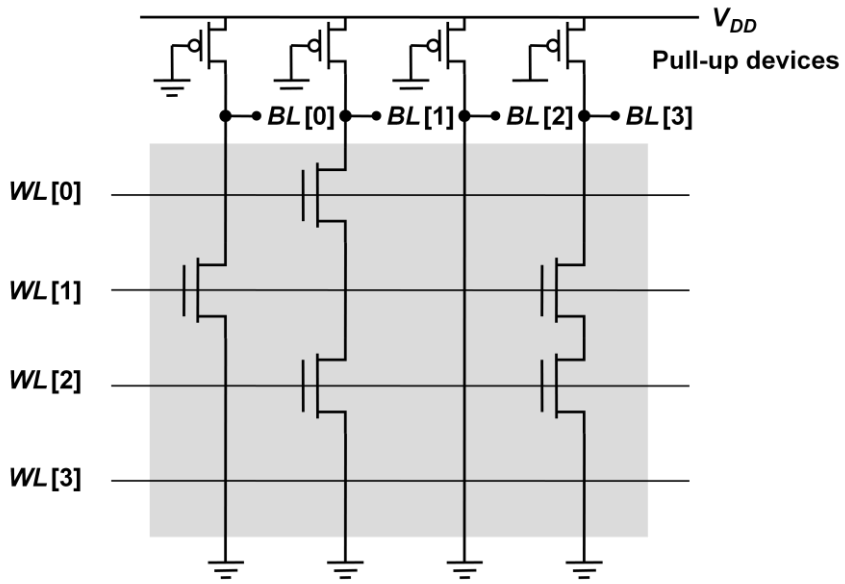
Resistance	Wire Capacitance	Gate Capacitance
$= {}^{7}/_{2} \times 5\,{}^{\Omega}/_{sq} = 17.5\Omega$	$= (3\lambda \times 2\lambda)(0.125)^2 \times 0.088 + 2 \times (3\lambda \times 0.125) \times 0.054 = 0.049fF$[1]	$= (4\lambda \times 2\lambda)(0.125)^2 \times 6 = 0.75fF$

- The bit line parasitics are:

Resistance	Wire Capacitance	Gate Capacitance
$= {}^{11}/_{4} \times 0.1\,{}^{\Omega}/_{sq} = 0.275\Omega$	$= (11\lambda \times 4\lambda)(0.125)^2 \times 0.041 + 2 \times (11\lambda \times 0.125) \times 0.047 = 0.09fF$	$= (5\lambda \times 4\lambda)(0.125)^2 \times 2 \times 0.56 + 14\lambda \times 0.125 \times 0.28 \times 0.6 + 4\lambda \times 0.125 \times 0.31 = 0.8fF$

[1] *"fF": Parasitic capacitance in high-performance integrated circuits can be measured in femtofarads (1 fF = 0.001 pF = 10^{-15} F).*

NAND BASED ROM MEMORY DESIGN:

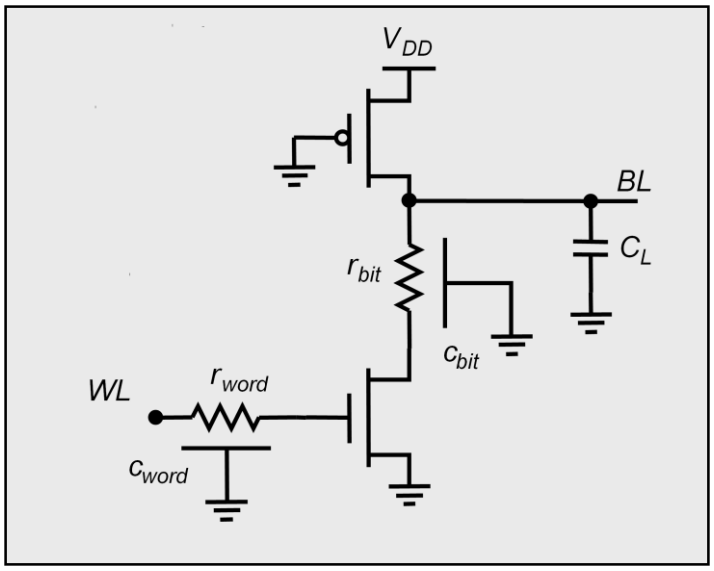


A 4 × 4 NAND ROM

The above fugue shows a 4 × 4 NAND ROM. The values of the data stored at addresses 0, 1, 2 and 3 are determined. The main advantage of using the NAND based structure is it that the basic cell consists of a transistor or no transistor and no connection to any of the supply voltages is needed, this reduces the cell size subsequently.

Here all word lines are high by default with the exception of selected row.

Equivalent model of NOR based memory design:



EQUIVALENT MODEL OF NAND BASED ROM

Using the above shown equivalent model for a NAND based ROM we can derive the delay of the NAND structure. It is identical to model the word line as the bit line behaviour is complex due to its long chain of connected transistors. The worst behaviour occurs when the transistor at the bottom of the chain is switched and the column is completely populated with transistors.

Each of the series transistors (which are normally in the ON mode) is modelled as a RC combination. The entire chain can be modelled as a distributed RC network for large memories.

- The word line parasitics are:

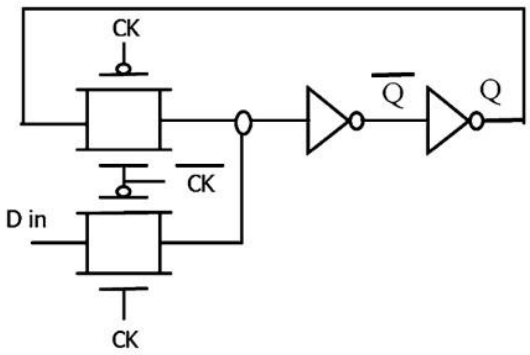
Resistance	Wire Capacitance	Gate Capacitance
$= {}^{6}/_{2} \times 5\,{}^{\Omega}/_{sq} = 15\Omega$	$= (3\lambda \times 2\lambda)(0.125)^2 \times 0.088 + 2 \times (3\lambda \times 0.125) \times 0.054 = 0.049fF$	$= (3\lambda \times 2\lambda)(0.125)^2 \times 6 = 0.56fF$

- The bit line parasitics are:

Resistance	Wire Capacitance	Gate Capacitance
$= {}^{13}/_{4} \times 1.5 = 8.7k\Omega$	*Included in diffusion capacitance*	$= (3\lambda \times 3\lambda)(0.125)^2 \times 2 \times 0.56 + 2 \times 3\lambda \times 0.125 \times 0.28 \times 0.6 + (3\lambda \times 4\lambda)(0.125)^2 \times 6 = 0.85fF$

In contrast to the NOR based ROM, here in NAND based ROM the source/ drain capacitance must include the gate- source and gate- drain capacitances, which means the complete gate capacitance whereas in NOR based ROM only the drain- source overlap capacitance is included.

D- FLIP PLOP USING TRANSMISSION GATES:



D- FLIP FLOP IMPLEMENTED USING TRANSMISSION GATES

The functioning of the flip flop incorporates storing a logical state in response of an input clock pulse. A transmission gate is similar to a relay which can conduct or block in both directions by a control signal almost any voltage potential.

As seen the D flip flop implemented using two transmission gates and a redundant feedback path for which,

- For CK = 1 causes Q_{n+1} = D, a bit is loaded.
- For CK = 0 causes $Q_{n+1} = Q_n$, thus a bit is stored through feedback.

For feedback along the path causes an abrupt increase of leakage current.

***COMBINATIONAL LOGIC*:**

The circuits in which the output/ outputs depends on the present input are referred as combinational logic circuits.

Examples of such circuits are adders, subtractors, multipliers, shifters etc.

In dense integration the major arithmetic logic circuits are adders, multipliers and shifters.

ADDERS:

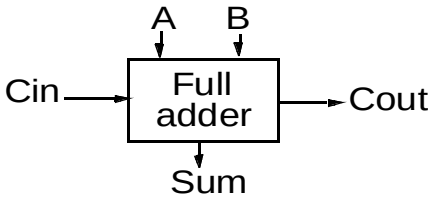
An adder is an input- output logical circuit which performs addition operation. Addition is the most commonly used arithmetic operation, often is the speed- limiting element as well. Therefore careful optimization of the adder is important, this optimization can proceed at the logic (using Boolean equations) or circuit level (using transistor sizing and circuit topology).

Some optimized adder circuits are:

1. Carry Select Adder
2. Carry Bypass Adder
3. Carry Look Ahead Adder
4. Manchester Carry Chain Adder etc

The Binary Adder:

It is a three input (*A, B and* C_{in}), two output (*Sum S and Carry Out* C_0) combination logic circuit which performs the arithmetic addition operation, usually referred as the binary adder or full adder.



A	B	C_i	S	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

$$S = A \oplus B \oplus C_i$$

$$= A\bar{B}\bar{C}_i + \bar{A}B\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i$$

$$C_o = AB + BC_i + AC_i$$

The sum and carry can be expressed as *Propagate (P), Generate (G) and Delete (D)* for all the input binary logical combinations depicted by the carry status as shown above. The three variables *P, G and D* depend on the inputs A and B only.

Generate (G) = AB

Propagate (P) = A ⊕ B

Delete = $\overline{A}\,\overline{B}$

Expressions for P, G and D

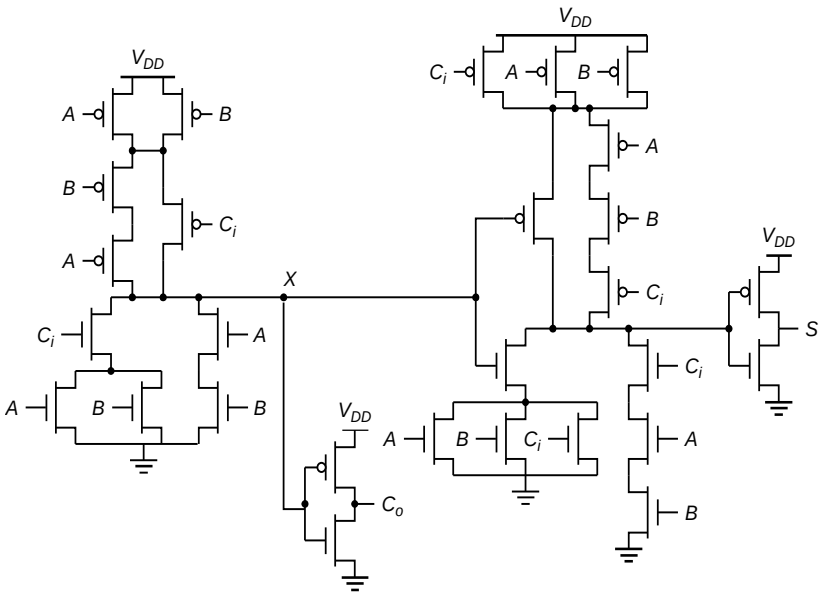
(Sometimes P = A+B is also used)

Expressing C_0 and S using *P* and *G*,

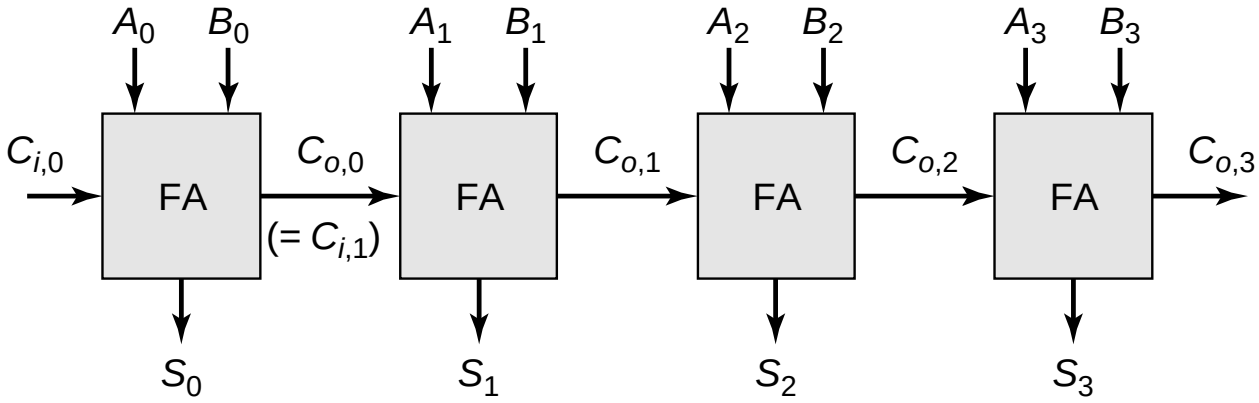
$$C_o(G,P) = G + PC_i$$

$$S(G,P) = P \oplus C_i$$

Expressions of C_0 and S can also be derived using *D* and *P* respectively.



COMPLIMENTARY STATIC CMOS FULL ADDER USING 28 TRANSISTORS



FOUR BIT RIPPLE CARRY ADDER TOPOLOGY

An N bit adder can be constructed by cascading N full adder (FA) circuits in series. Connecting $C_{o,\,k-1}$ to $C_{i,\,k}$ for k = 1 to *N-1*, and the first carry- in $C_{i,0}$ to 0 as shown in figure. This configuration is called **ripple- carry adder**, since the carry bit ripples from one stage to the other. The delay through the circuit depends upon the number of logic stages that must be traversed and is a function of the applied input signals.

For some input signals no rippling effect occurs at all. While for others the carry has to ripple all the way from LSB to the MSB. The propagation delay of such structure also called as critical path is defined as the worst case delay over all possible input paterns.

In ripple carry adder the worst case delay happens when a carry generated at least significant bit (LSB) position propagates all the way to the most significant bit (MSB) position. This carry is finally consumed in the last stage to produce the sum. The delay is then proportional to the number of bits in the input words N and is approximated by,

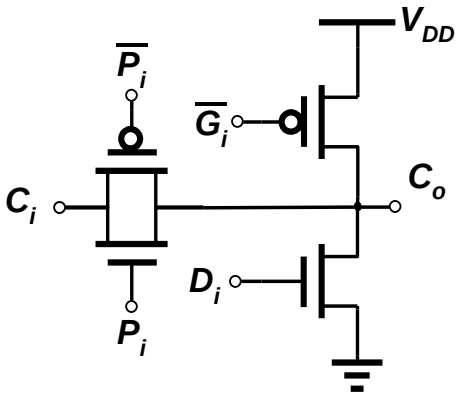
$$\boldsymbol{t_{adder} \approx (N-1)t_{carry} + t_{sum}}$$

Where:

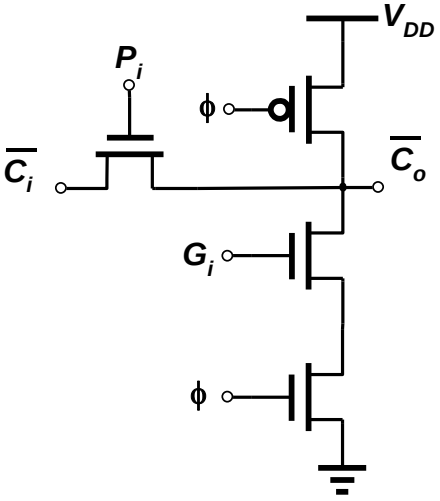
t_{carry} *and* t_{sum} *equal the propagation delays from* C_i *to* C_o *and S*

Manchester Carry Chain Adder:

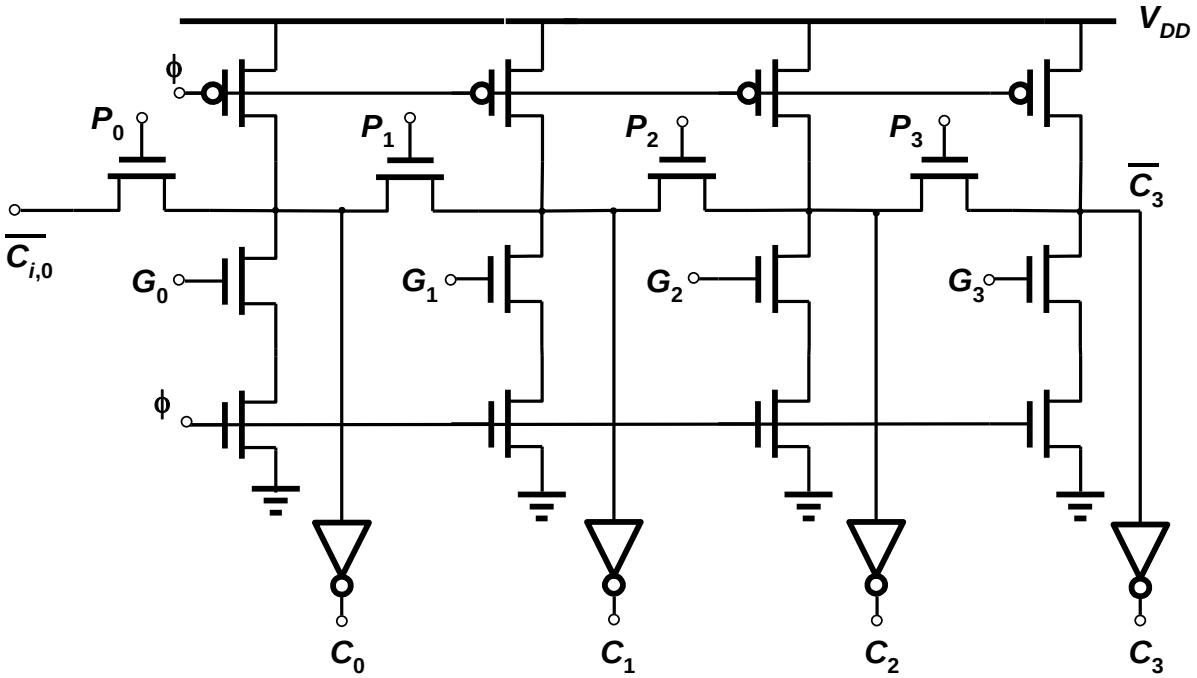
The Manchester carry chain adder uses cascading of pass transistors to implement the carry chain. It can be implemented operating as a static or dynamic. In static implementation it uses propagate, generate and kill/ delete signals and in dynamic implementation it uses only propagate and generate signals.



STATIC LOGIC IMPLEMENTATION



DYNAMIC LOGIC IMPLEMENTATION



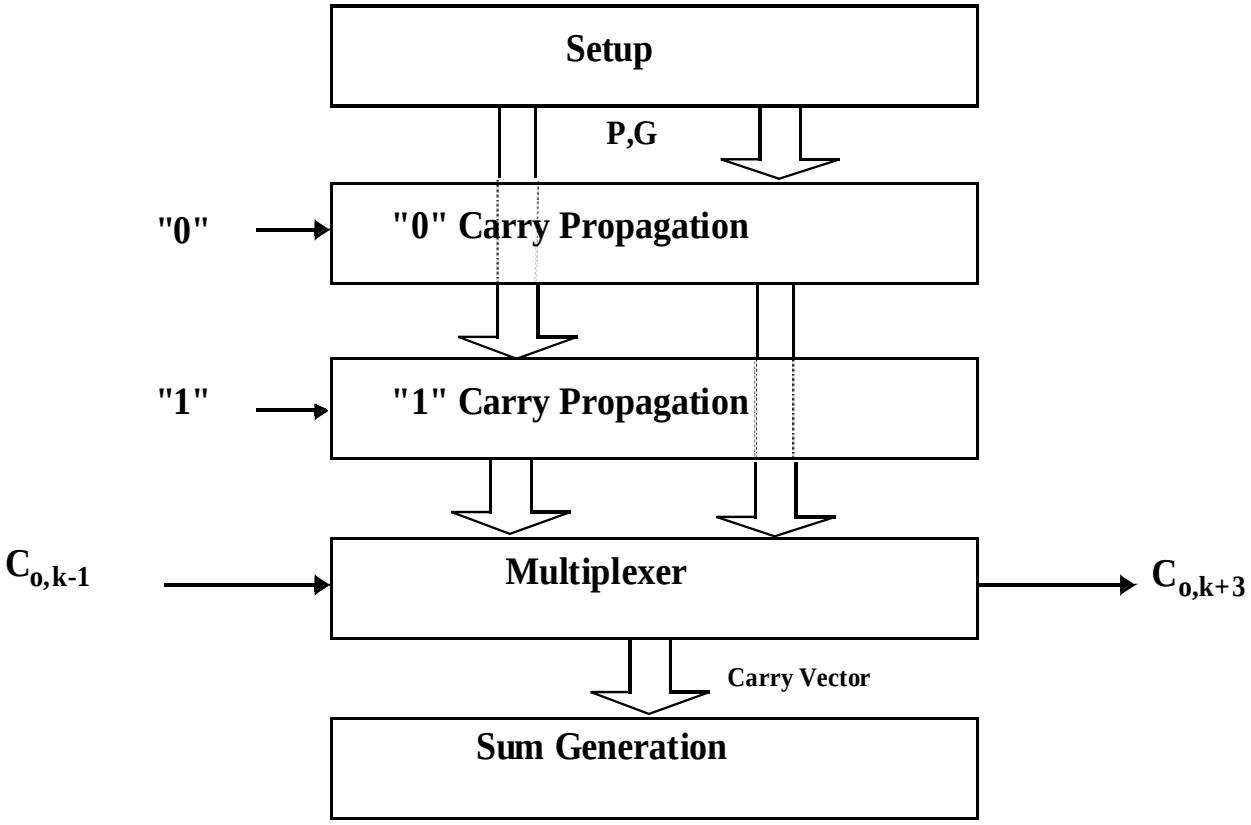
MANCHESTER CARRY CHAIN ADDER (IN DYNAMIC LOGIC) (FOUR- BIT)

The Manchester carry chain adder can be seen in the above figure in dynamic logic. During the pre charge phase ($\phi = 0$), all the intermediate nodes of the pass transistor carry chain are pre charged to V_{DD}. During

evaluation, the A_k node is discharged when there is a incoming carry and the propagate signal P_k is high, or when the generate signal for stage k (G_k) is high.

Carry Select Adder:

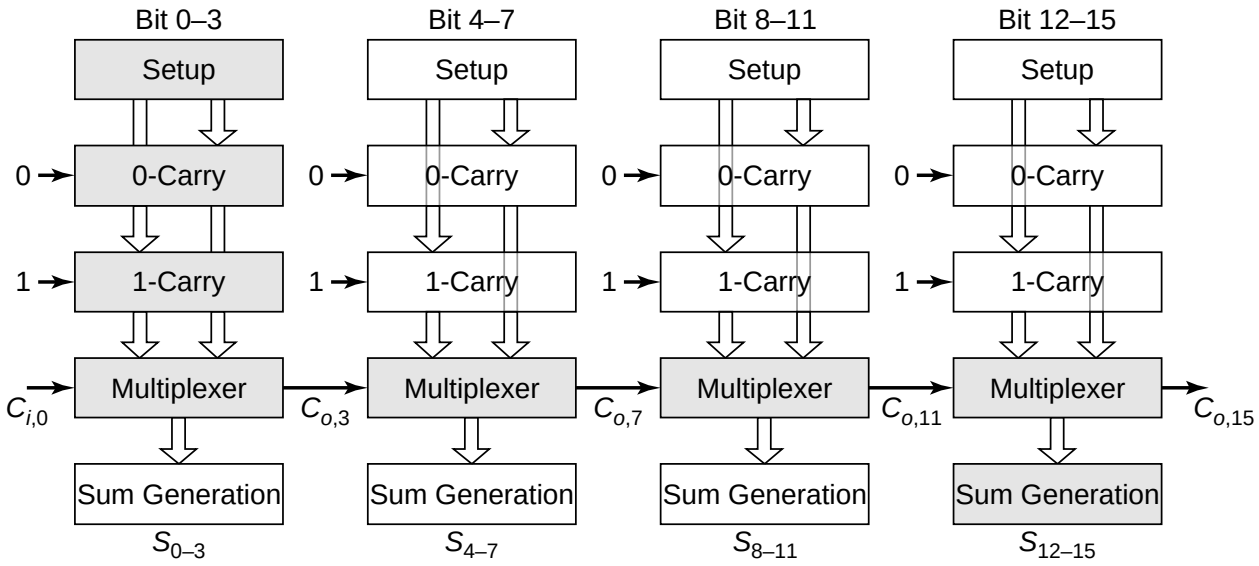
Carry select adder overcomes the problem of delay/ waiting for a carry in a ripple carry adder, where in the ripple carry adder- each adder cell has to wait for the incoming carry before an outgoing carry can be generated. This is done by anticipating both possible values of the carry input and evaluate the result for both possibilities in advance. Once the real value of the incoming carry is known, the correct result is easily selected with a simple multiplexer stage. This implementation is referred as **Carry- Select Adder**.



FOUR BIT CARRY SELECT ADDER MODULE TOPOLOGY

As seen from the above figure which illustrates a four bit Carry Select Adder. Consider the block of adders, which is adding bits *k* to *k+3,* instead of waiting on the arrival of the output carry of the bit *k-1,* both the 0 and 1 possibilities are analyzed. From a circuit point of view this means that two carry paths are implemented. When $C_{0,\,k-1}$ finally settles, either the result of the 0 or 1 path is selected by the multiplexer, which can be performed with a minimal delay. As it is evident from the figure, the hardware overhead of the carry select adder is restricted to an additional carry path and a multiplexer, and equals about 30% with respect to a ripple carry structure.

A full carry select adder can be constructed by chaining a number of equal length adder stages, as in the carry- bypass approach.



SIXTEEN BIT CARRY SELECT ADDER (CRITICAL PATH IS SHOWN BY SHADING)

From the inspection of the circuit we can derive a first order model of the worst case propagation delay of the module as,

$$t_{add} = t_{setup} + Mt_{carry} + \left(\frac{N}{M}\right) t_{mux} + t_{sum}$$

Where:

t_{setup}, t_{sum} *and* t_{mux} *are fixed delays*

N and M are the total number of bits and the number of bits per stage

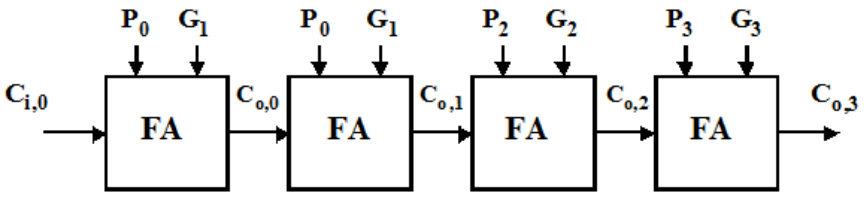
t_{carry} *is the delay of the carry through a single full adder cell*

The carry delay through a single block is proportional to the length of that stage or equals $M\ t_{carry}$.

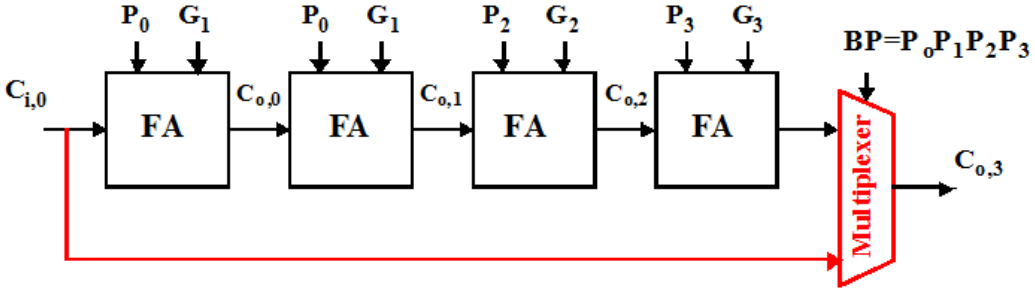
The propagation delay of the adder is again linearly proportional to *N*. The reason for this linear behaviour is that the block- select signal that selects between the 0 and 1 solutions still has to ripple through all stages in the worst case.

Carry Skip Adder:

A carry-skip adder (also known as a carry-bypass adder) is an adder implementation that improves on the delay of a ripple-carry adder with little effort compared to other adders. The improvement of the worst-case delay is achieved by using several carry-skip adders to form a block-carry-skip adder.



Carry Propagation



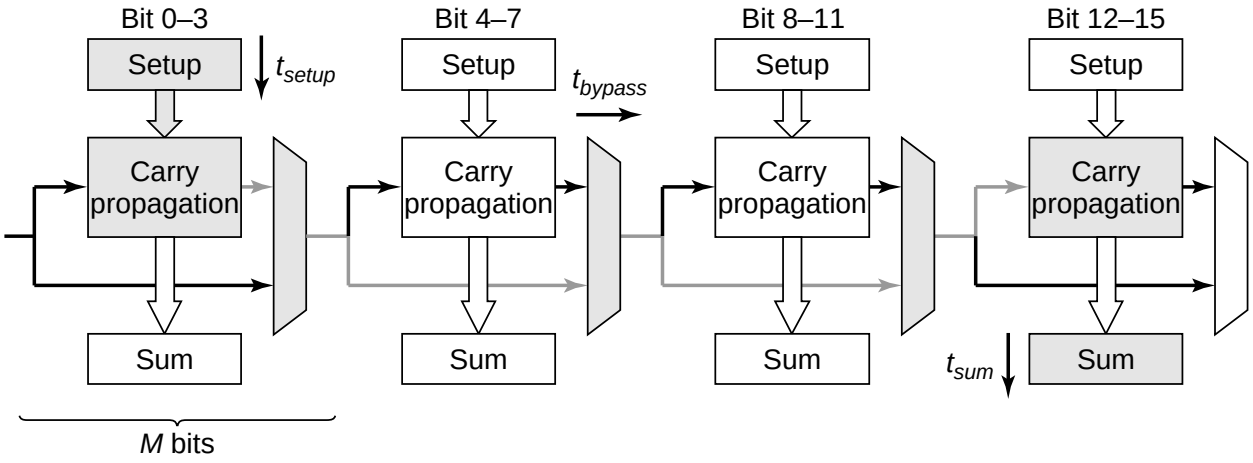
Adding a Bypass

BASIC CARRY BYPASS/ SKIP ADDER STRUCTURE

Consider the four bit adder block as shown in figure "Carry Propagation". Let the values of A_k and B_k ($k=0...3$) are such that all propagate signals P_k ($k=0...3$) are high. An incoming carry $C_{i,0}$ =1 propagates under those conditions through the complete adder chain and causes an outgoing carry $C_{o,3}$=1. i.e.

$$If\ (P_0P_1P_2P_3 = 1)\ then\ C_{o,3} = C_{i,3}\ else\ either\ \boldsymbol{Delete}\ or\ \boldsymbol{Generate}\ occured$$

This can be used to speed up the operation of the adder as shown in figure "Adding a Bypass". When BP = P_0 P_1 P_2 P_3 = 1, the incoming carry is forwarded immediately to the next block through the bypass transistor M_b, hence the name carry- bypass or carry- skip adder.



SIXTEEN BIT CARRY BYPASS ADDER (WORST CASE SHOWN SHADED)

The delay of a 16 bit carry bypass adder is done by assuming that the total adder is divided in (N/M) equal length bypass stages, each of which contains M bits. An approximate expression for the total propagation time can be derived from the above shown figure as,

$$t_p = t_{setup} + Mt_{carry} + \left(\frac{N}{M} - 1\right) t_{bypass} + (M-1)t_{carry} + t_{sum}$$

Where:

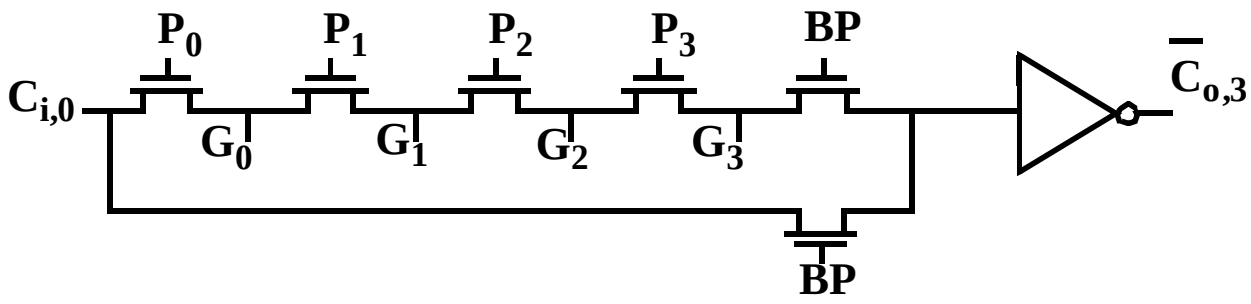
t_{setup} *is the fixed overhead time to create the generate and propagate signals*

t_{carry} *is the propagation delay through a single bit. The worst case carry- propagation delay through a single stage of M bits is approximately M times larger.*

t_{bypass} *is the propagation delay through the bypass multiplexer of a single stage*

t_{sum} *is the time to generate the sum of the final stage*

Carry Bypass in Manchester Carry- Chain Adder:



MANCHESTER CARRY CHAIN IMPLEMENTATION OF BYPASS ADDER

As seen from the figure, it shows the possible carry- propagation paths when the full adder circuit is implemented in Manchester Carry style. This figure shows how the bypass speeds up the addition. The carry propagates either through the bypass path or a carry is generated somewhere in the chain. In both cases the delay is smaller than the normal ripple configuration.

The area overhead incurred by adding the bypass path is small and typically ranges between 10 and 20%, however adding the bypass path breaks the regular bit- slice structure.

MULTIPLIERS:

A multiplier is an in effect complex adder arrays which performs multiplication operation. These multiplication operations are expensive and slow. The performance of many computational problems often is dominated by the speed at which a multiplication operation can be executed. This has promoted the integration of complete multiplication units in state of the art digital signal processors and microprocessors.

The analysis of the multiplier gives us an further insight into how to optimize the performance (or the area) of complex circuit topologies.

Consider two unsigned binary numbers A and B that are M and N bits wide respectively. To introduce the multiplication operation it is useful to express A and B in the binary form.

$$A = \sum_{i=0}^{M-1} A_i 2^i \ \ and \ \ B = \sum_{j=0}^{N-1} B_j 2^j$$

With A_i and $B_j \in \{0,1\}$. The multiplication operation is then defined as:

$$C = A \times B = \sum_{k=0}^{M+N-1} Z_k 2^k$$

$$= \left(\sum_{i=0}^{M-1} A_i 2^i\right)\left(\sum_{j=0}^{N-1} B_j 2^j\right) = \sum_{i=0}^{M-1}\left(\sum_{j=0}^{N-1} A_i B_j 2^{\,i+j}\right)$$

The simplest way to perform a multiplication is to use a single two input adder, for inputs M and N bit wide the multiplication takes M cycles using N bit adder. This *shift and add algorithm* for multiplication adds together M partial products. Each partial product is generated by multiplying the multiplicand with a bit of the multiplier which essentially is an AND operation, and by shifting the result on the basis of the multipliers bit's position.

A faster way of implementing multiplication is to resort to an approach similar to manually computing a multiplication. All the partial products are generated at the same time and organized in an array. A multioperand addition is applied to compute the final product as shown below.

```
                1 0 1 0 1 0     Multiplicand
x                 1 0 1 1       Multiplier
-------------------------------
                1 0 1 0 1 0
              1 0 1 0 1 0
            0 0 0 0 0 0         Partial products
+         1 0 1 0 1 0
-------------------------------
        1 1 1 0 0 1 1 1 0       Result
```

AN EXAMPLE OF BINARY MULTIPLICATION

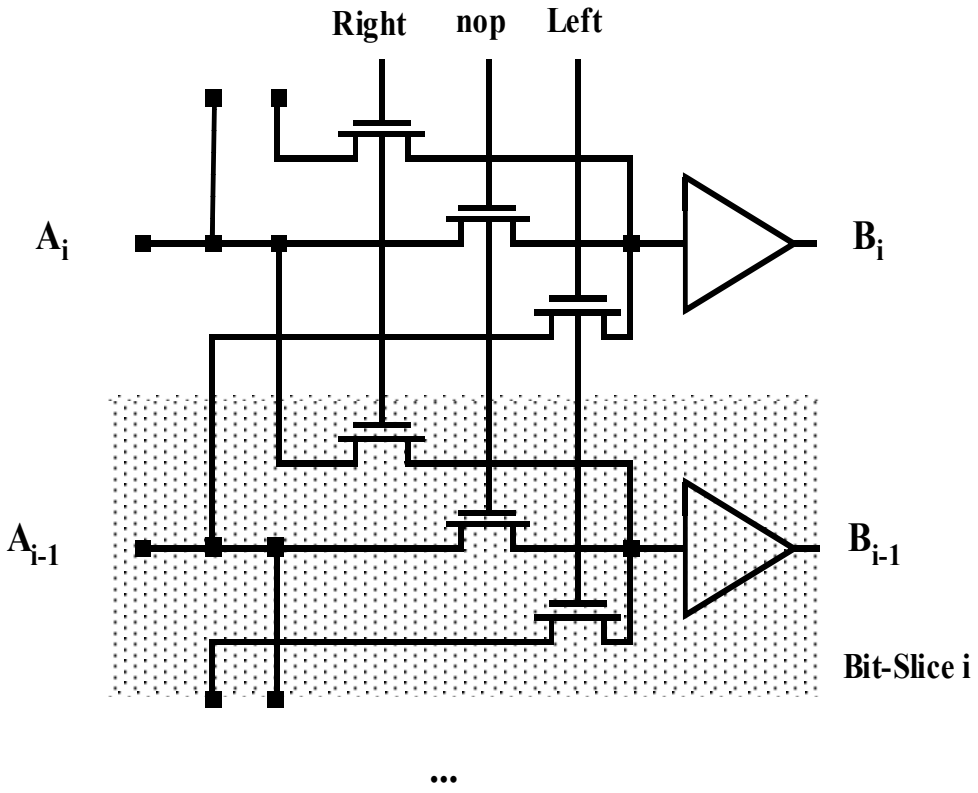
As shown above. This set of operations can be mapped directly into hardware. The resulting structure is called an ***array multiplier*** and combines the *partial- product generation, partial- product accumulation and final addition* functions.

SHIFTERS:

The shifter performs the shifting operation which is an essential arithmetic operation that requires adequate hardware. A shifter shifts a data word left or right over a constant amount and is implemented by an appropriate signal wiring. Latter can be implemented as a combination of add and shift operations.

Shifters are extensively used in floating point units, scalars and multiplications by constant numbers.

Programmable shifters are more complex and require active circuitry, these are nothing less than an intricate multiplexer circuit.



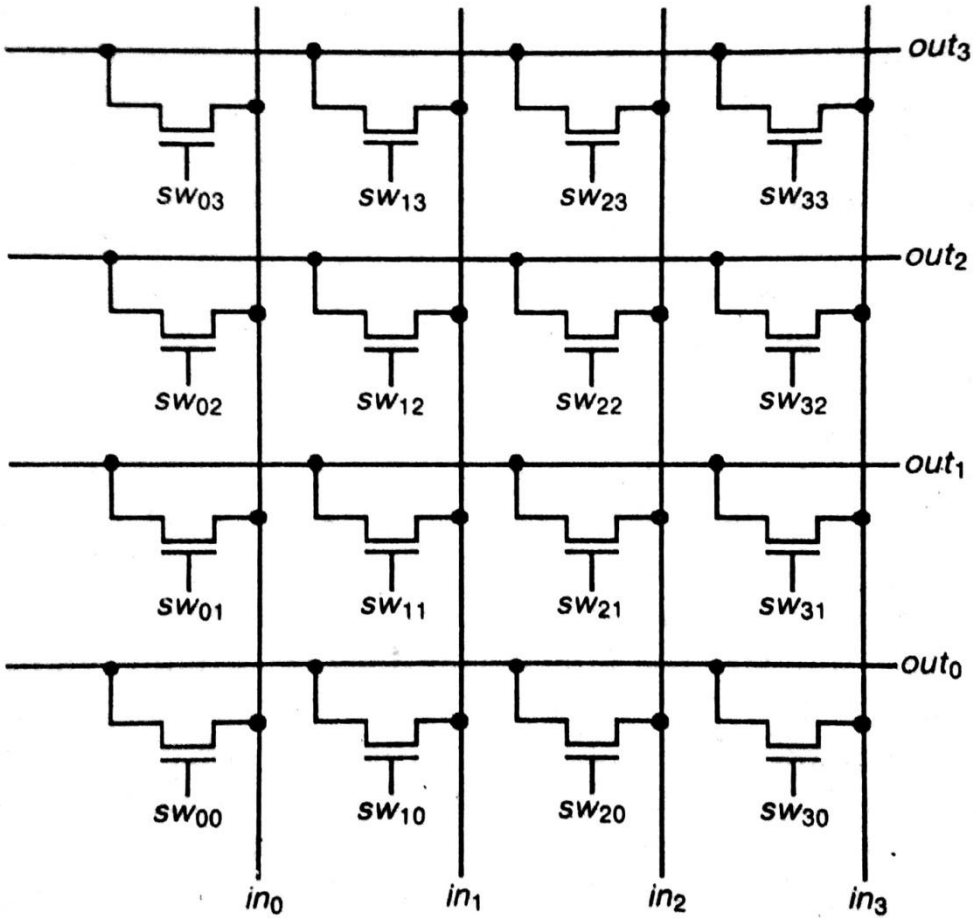
ONE BIT PROGRAMMABLE SHIFTER (LEFT- RIGHT SHIFTING)

The figure shows a simple one bit shifter. Depending on the control signals, the input word line is either shifted left or right, or else remains unchanged. Multi bit shifters can be built by cascading a number of these units. Multi bit shifters are complex and slow for larger shift values.

There are two types of shifters:

- Crossbar Shifter
- Barrel Shifter

Crossbar Shifter:

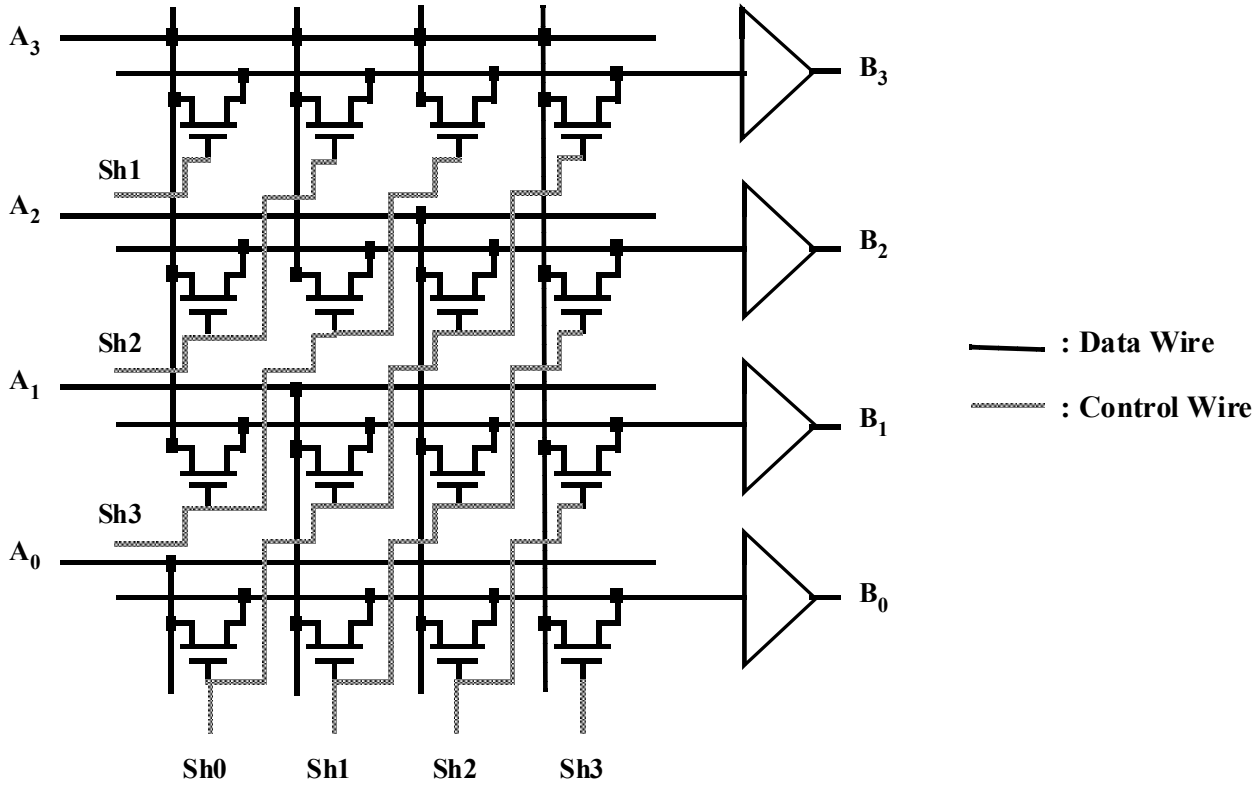


A 4 × 4 CROSSBAR SHIFTER

The structure of the four bit shifter known as crossbar shifter is shown, which consists of 16 transistors connected in lattice like structure with 16 control signals, one for each transistor must be provided to drive the crossbar shifter. To carry out the shift operation four control signals have to be activated, then the corresponding transistor switches is turned ON and the data gets shifted to right by one place. This shifter requires 16 control signals as the number of bits are increased, control signals are also increased. If the control signal due to fault or error gets missed, then all the transistor switches might get turn ON, all inputs connected to all outputs, resulting in short circuit.

Therefore to avoid the problem happening in crossbar shifter an adaptation of this arrangement that we can couple the switch gates together in groups of four as its a four bit shifter and also form four separate groups corresponding to shift of zero, one, two and three bits. This arrangement is readily adapted so that the in- lines also run horizontally, this arrangement is known as *Barrel Shifting or the shifter is known as Barrel Shifter*.

Barrel Shifter:



A 4 × 4 BARREL SHIFTER

The structure of the barrel shifter is shown in figure. It consists of an array of transistors, in which the number of rows equals the word length of the data, and the number of columns equals the maximum shift width. Here in the structured considered both are equal set to four. The control wires are routed diagonally through the array. The barrel shifter is an overcome of the drawbacks of a crossbar shifter.

The main advantage of this shifter is that the signal has to pass through at most one transmission gate, in other words the propagation delay is theoretically constant and independent of the shift value or shifter size. This is not practical as the capacitance at the input of the buffers rises linearly with the maximum shift width.

Here the layout size of the circuit is not dominated by the active transistors as in the case of all other arithmetic circuits but by the number of wires running through the cell. The size of the cell is bounded by the pitch of the metal wires. Barrel shifter needs control signals to shift, and the number of control signals depends on the number of bits i.e. here for a four bit barrel shifter, four control signals are required and the barrel shifter requires an extra decoder to decode the signal while shifting into the former when required.

Assignment:

1. Explain Partial- product generation, partial- product accumulation and final addition in multipliers?
2. What is Pseudo- NMOS?
3. Differentiate between static and dynamic memories?
4. Draw the gate level logic diagram of a half adder, full adder and D- flip flop along with their truth tables.