

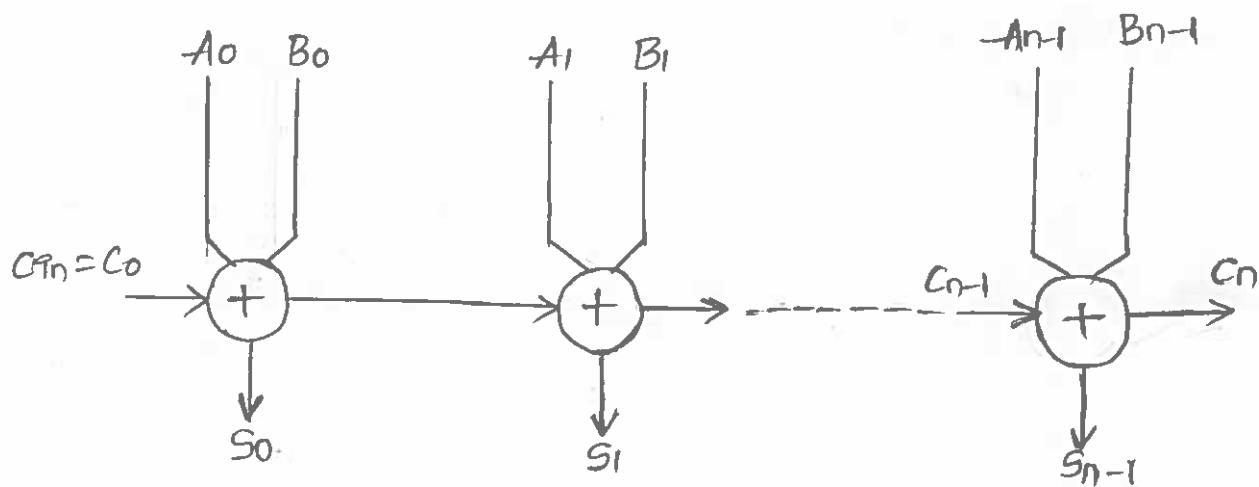
## VLSI CMOS SUBSYSTEM DESIGN

## 7.1 Parallel ADDERS:-

Parallel adders are the most important elements used in arithmetic operations of microprocessors, DSPs etc. As in any logic design they are constrained by parameters such as speed, area, and power dissipation. The adder cell is also an element of multipliers, dividers, multiplier-accumulators (MACs), etc. Among the various adders implementations used in many designs, we can cite the following classes.

- Ripple Carry Adders (RCA);
- Carry Look-Ahead Adders (CLA);
- Carry Select Adders (CS); and
- Conditional Sum Adders (CSA).

## → Ripple Carry Adders :-



In a n-bit adder, a propagation of the carry always occurs. This propagation limits the speed of the adder. The simplest way to construct an n-bit adder is to cascade  $n-1$  bit adders shown in fig: this adder is called Ripple carry Adder (RCA).

Because the carry ripples through the n stages, the sum of the nth bit can't be performed until the carry

$C_{n-1}$  is evaluated. The delay of  $n$ -bit addition is given by.

$$t_{\text{ca}} = (n-1) t_c + t_s$$

where,  
 $t_c$  is the carry delay  
 $t_s$  is the sum delay.

Since the carry propagation path is a critical stage for the delay, the full-adder cell should be optimized. The sum and carry out are given by

$$S = A \oplus B \oplus C$$

$$C_{\text{out}} = A \cdot B + (A+B) \cdot C_{\text{in}}$$

Compared to the conventional CMOS full-adder implementation, there is no inverter stage. Therefore, the carry delay is reduced. To optimize the cell, the transistors in the carry path  $W_{\text{op}}$  and  $W_{\text{in}}$ , can be sized up. It is symmetrical and leads to better layout and small area. Since the outputs are complemented, and in order to implement an RCA circuit, the configuration of Fig: 7.3 can be used. In this case, many cells are use inverted ~~outputs~~ inputs.

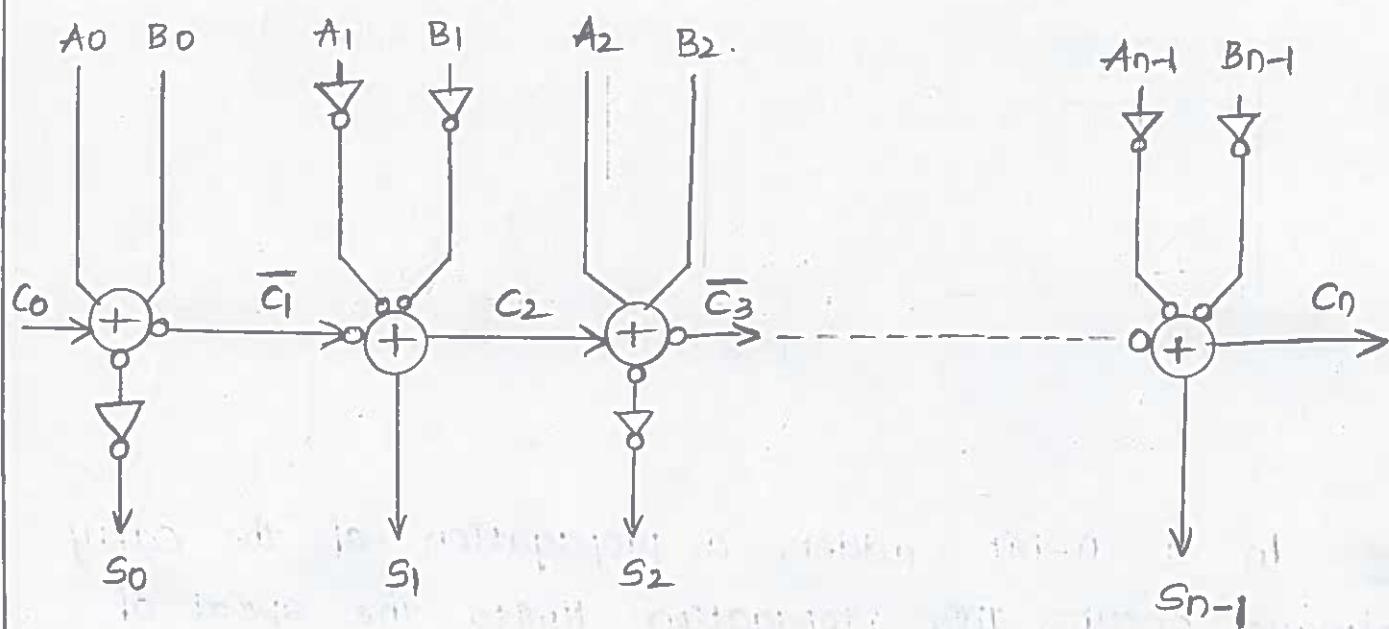


Fig: 7.3 Parallel Ripple carry adder for the Optimized cell.

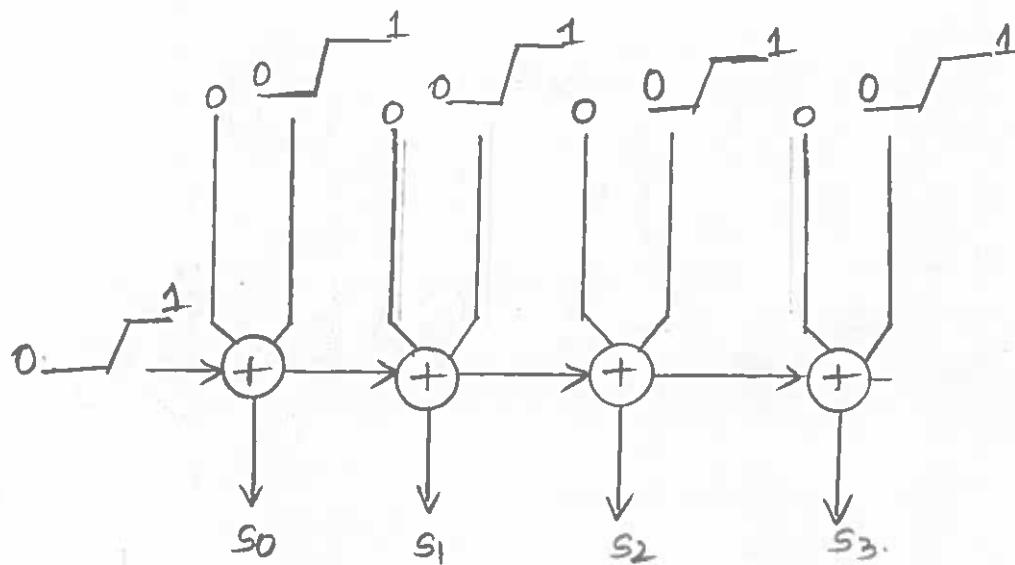
Note that an n-bit RCA circuit is subject to the glitching problem. It shows a static simulation of a 4-bit adder, with the inputs A<sub>1</sub> set to zero (0), and the inputs B<sub>1</sub> and C<sub>in</sub> rising from 0 to 1. The outputs S<sub>1</sub> should stay at 0, however due to the delay of the carry signal, through chain of full adders, the outputs exhibit spurious transitions (glitching).

### → Carry Look-Ahead Adders:

To avoid the linear growth of the carry delay, we use a carry Lookahead Adder (CLA) in which the carries can be generated in parallel. The carry of each bit is generated from the propagate and the generate signals ( $P_i, G_i$ ) as well as the input carry ( $C_0$ ). The propagate and the generate signals ( $P_i, G_i$ ) are derived from the operands A<sub>1</sub> and B<sub>1</sub> by

$$G_P = A_1 \cdot B_1$$

$$P_P = A_1 + B_1.$$



sum voltage(V)

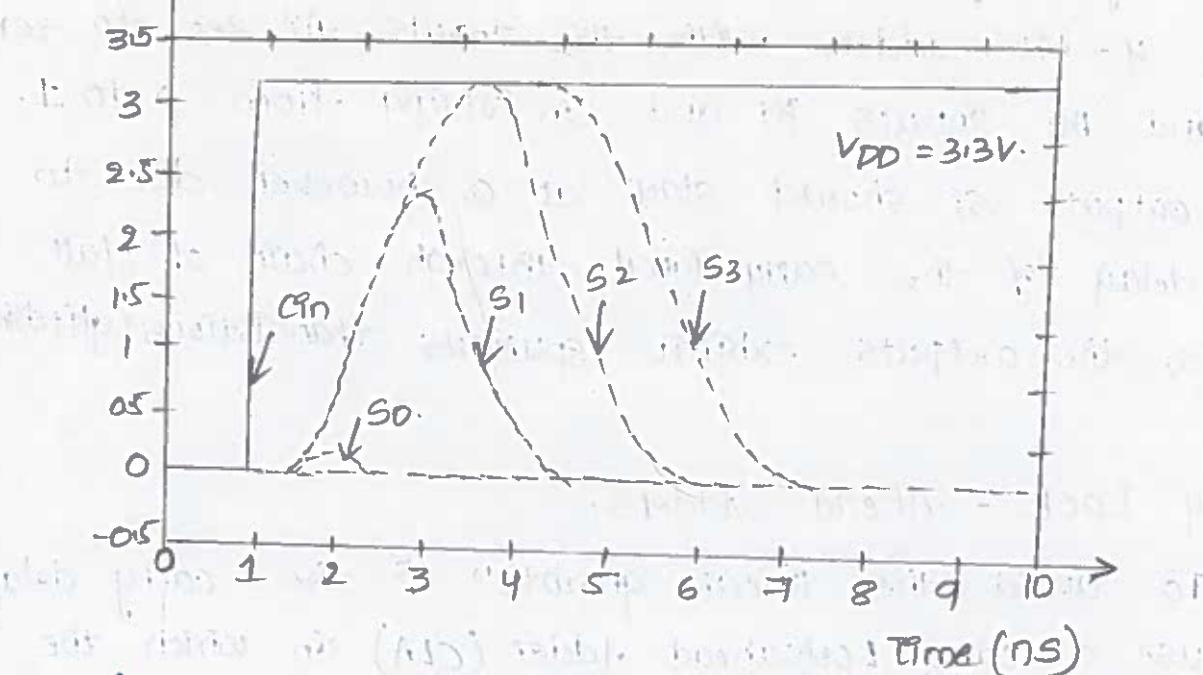


fig:7.4 Sum voltage waveforms with Q-flicking phenomenon.

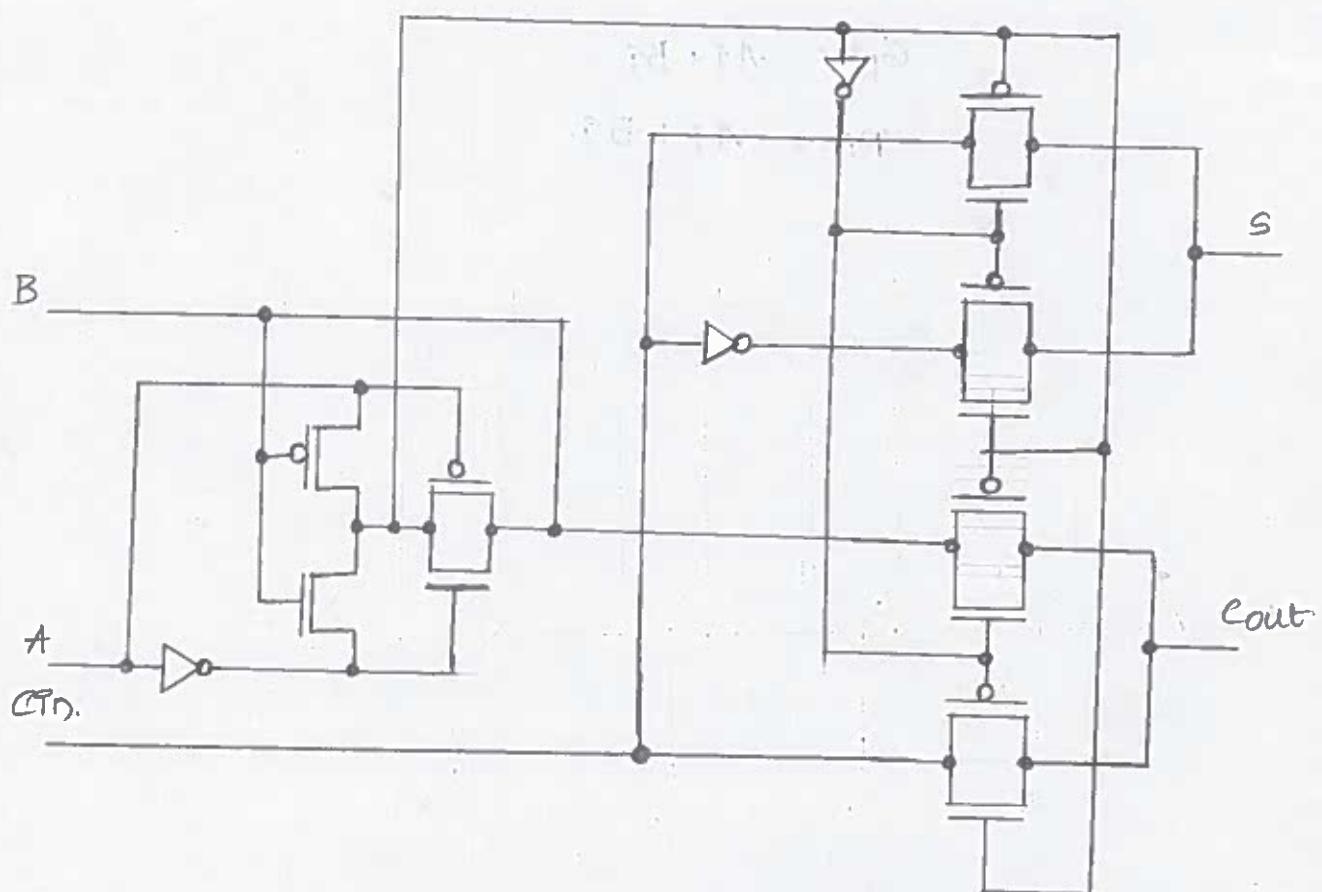


Fig: 7.5 Optimized TG full-adder cell.

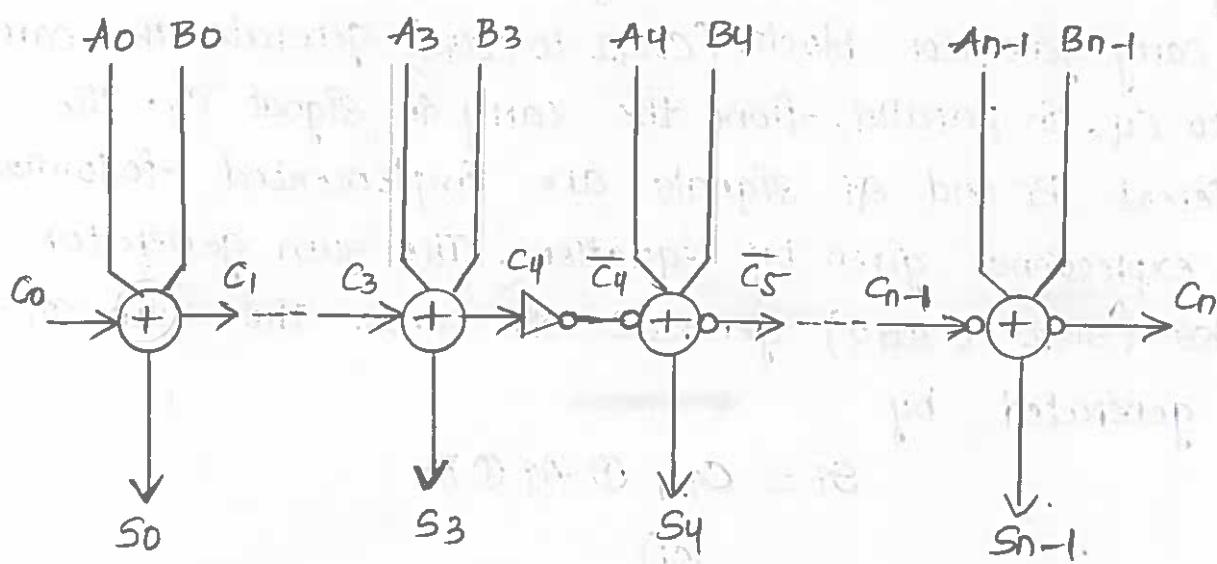


fig: Ripple carry adder for the optimized TG<sub>1</sub> cell.

$$C_{in} = C_0$$

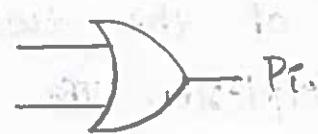
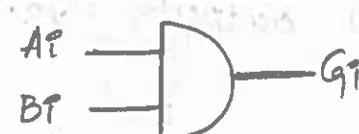
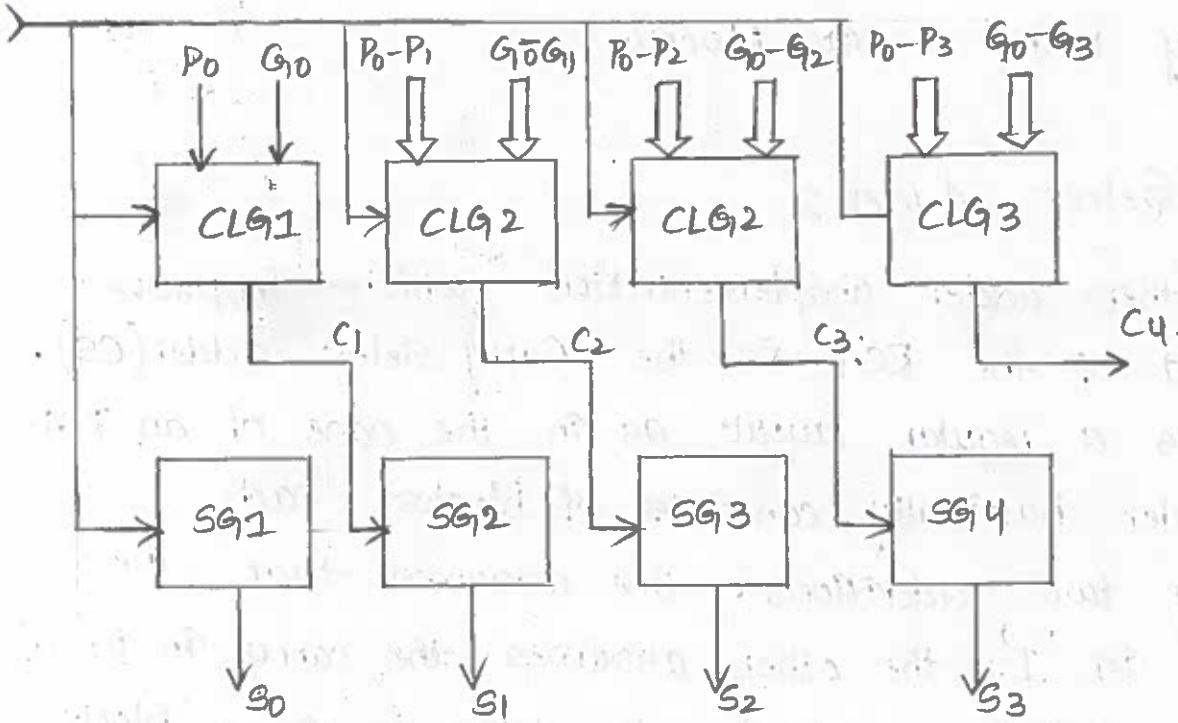


fig:7.7 Block diagram of 4-bit carry lookahead adder (CLA)

The carries of the four stages are given by.

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

Fig:7.7 shows the block diagram of a 4-bit CLA adder. The carry generator blocks (CG1 to CG4) generate the carries  $C_1$  to  $C_4$ , in parallel, from the carry in signal  $C_0$ . The different  $P_F$  and  $G_F$  signals are implemented following the expressions given by equations. The sum generator blocks (SG1 to SG4) generate the sums. The sum  $S_i$  is generated by.

$$S_i = C_{i-1} \oplus A_i \oplus B_i \\ (\text{or})$$

$$S_i = C_{i-1} \oplus P_i.$$

If the propagate signal  $P_i$  is given by

$$P_i = A_i \oplus B_i.$$

In general, an  $n$ -bit CLA adder can be implemented efficiently using 4-bit blocks.

#### → Carry-Select Adder:-

Another adder implementation which improves the speed of the RCA is the Carry Select adder (CS). It provides a regular layout, as in the case of an RCA. A 'es' adder basically consists of blocks; each executing two additions. One assumes that the carry in is '1'; the other assumes the carry in is '0'. The real carry is computed from the previous block and selects one of the two sum outputs with a simple 2:1 multiplexer.

The carry signal,  $C_4$ , selects the next four sums and carry  $C_5$ . The 4-bit adder blocks usually use RCA with transmission gate implementation. For a 32-bit adder, the use of normal staggering 4-4-4-4-4-4, doesn't lead to an optimum delay. This is due to the multiplexing delay of the next carry.

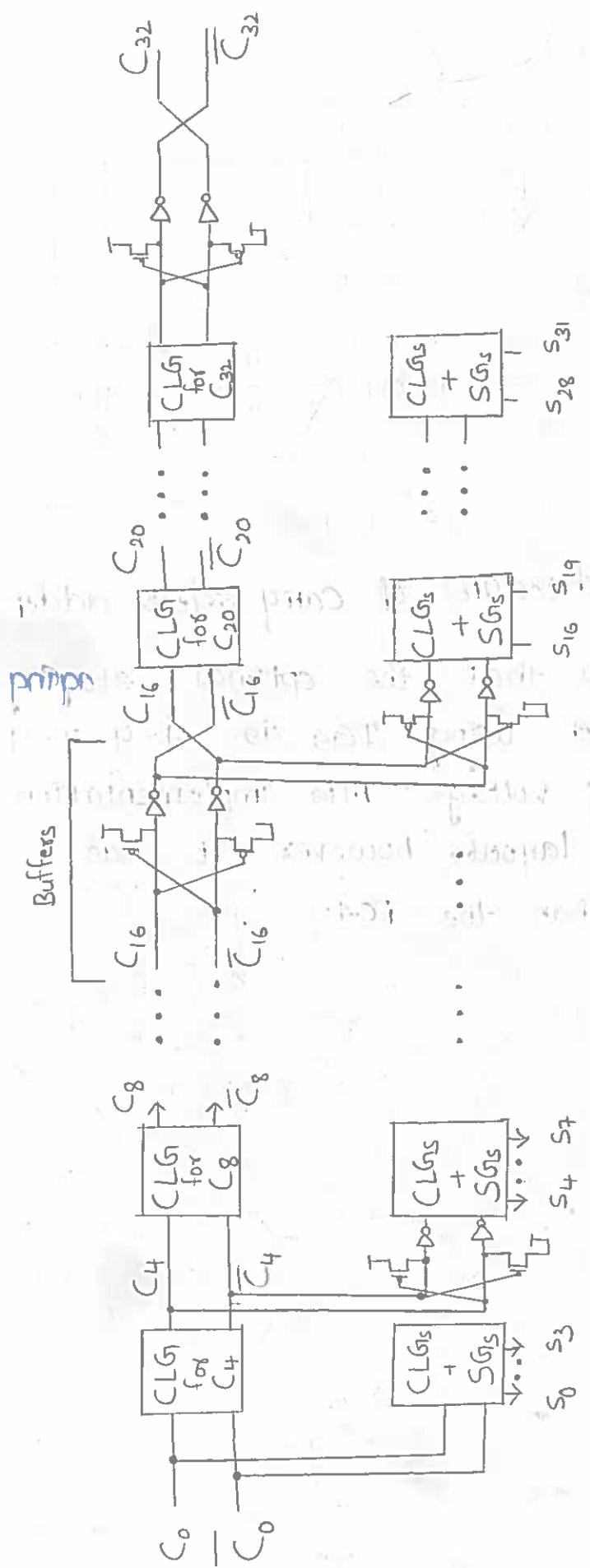
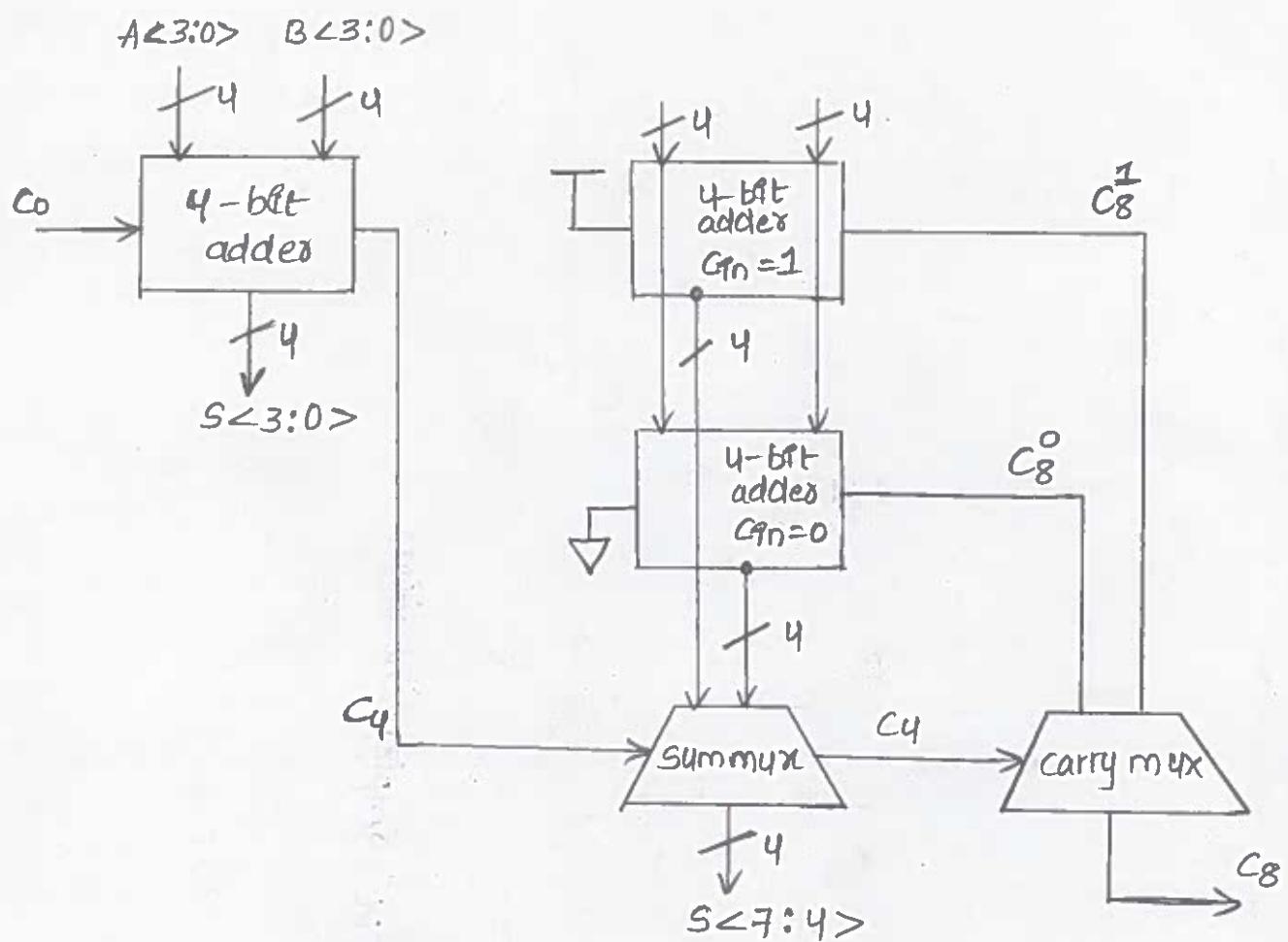


Fig: 7.11 Block diagram of CPL implementation



Figs:- An 8 bit architecture of carry Select adder.

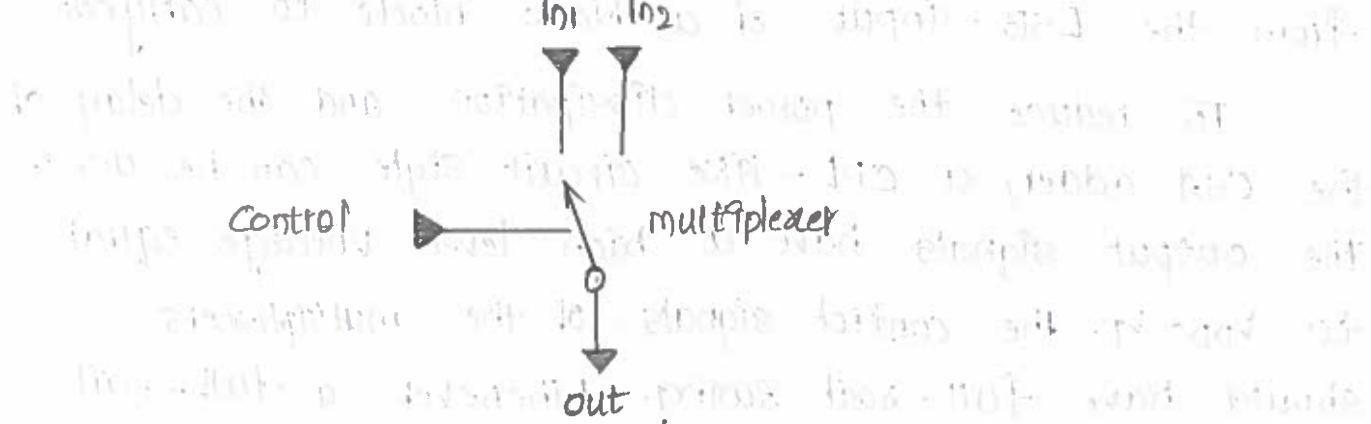
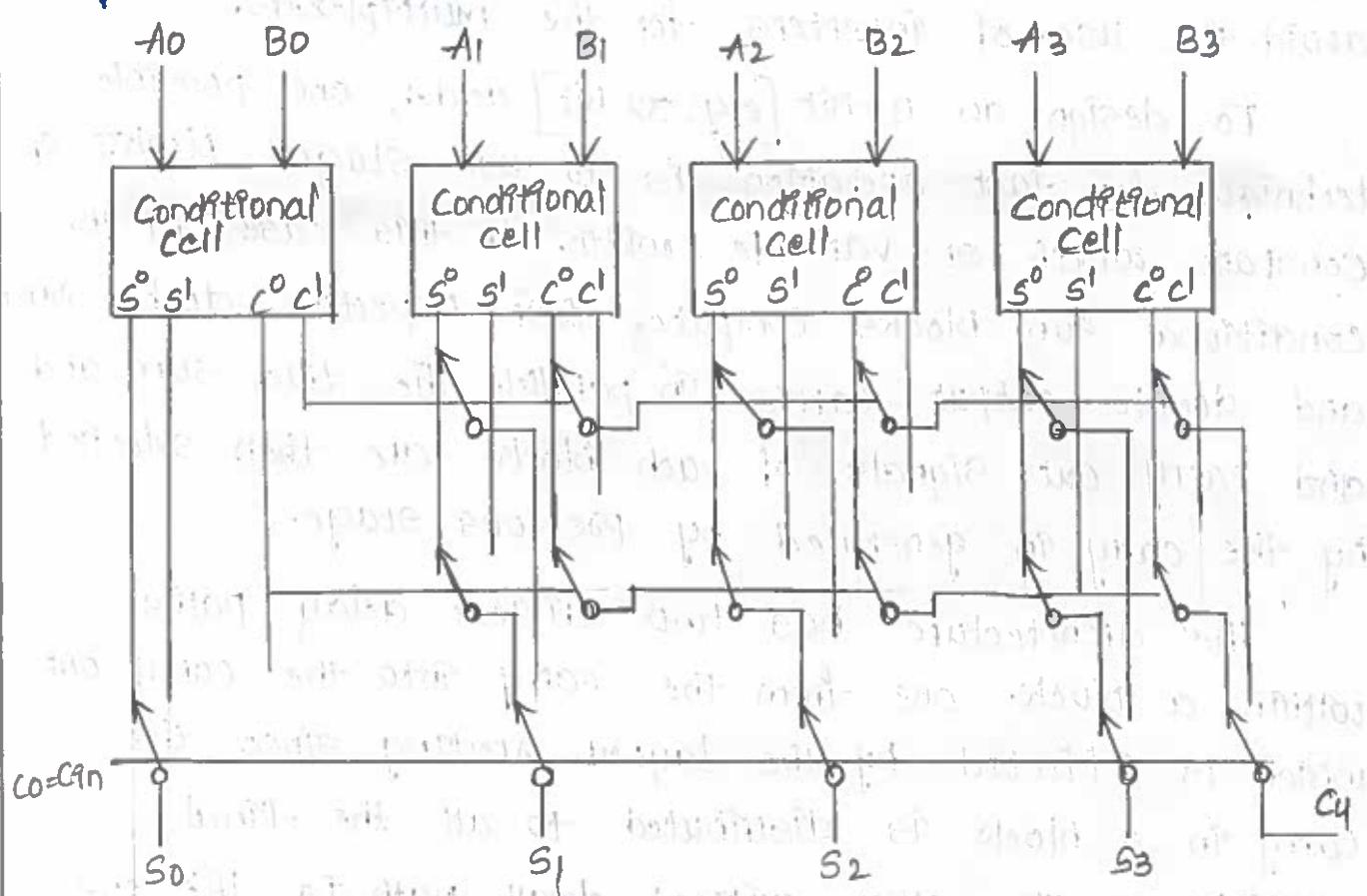
Simulations show that the optimal staging of a 32-bit CS adder using TGs is 4-4-7-9-8 at 3.3V power supply voltage. This implementation is regular and easy to layout, however it has a higher occupied area than the RCA.

## → Conditional Sum adders :-

Conditional Sum Adders (CSA) as the fastest one, from a theoretical point of view. The concept behind this architecture is explained using the basic circuit. It uses two types of cells. → The conditional cell  
→ The multiplexer.

For each bit there is one conditional cell circuit. It computes two sums and two carries.  $s^0$  and  $c^0$  are calculated for a carry in zero, and  $s^1$  and  $c^1$  are calculated for a carry in one. The selection of the true sum is done with the first carry in and previous carries. The true final carry out.  $c_4$  is also selected.

Fig:- 7.16 A simplified schematic of a 4 bit CSA.



A possible implementation of the conditional sum adder is shown in fig: 7.17 for the case of a 4-bit adder [4]. The conditional cell can be implemented with the compact logic elements. the different signals of the conditional cell are constructed using the following relations.

$$\overline{S_0^0} = A_0 \cdot B_0 + \overline{A_0} \cdot \overline{B_0}$$

$$\overline{C_0^0} = \overline{A_0} \cdot B_0$$

$$\overline{S_1^0} = \overline{A_0} \cdot B_0 + A_0 \cdot \overline{B_0}$$

$$\overline{C_1^0} = \overline{A_0 + B_0}$$

The adder uses mainly for the multiplexers transmission gates. Note that the architecture uses the signals and their components [dual-rail architecture] to avoid the use of inverters for the multiplexers.

To design an  $n$ -bit [e.g; 32 bit] adder, one possible technique for fast operation is to use staged blocks of constant width or variable width. In this case, all the conditional sum blocks compute their respective double sum and double output carries in parallel. The true sum and first carry out signals of each blocks are then selected by the carry in generated by previous stage.

The architecture has two critical delay paths within a block. one from the carry into the carry out which is affected by the layout routing since the carry in a block is distributed to all the final multiplexers. The other critical delay path is the one from the LSB input of a block block to carry out.

To reduce the power dissipation and the delay of the CSA adder, a CPL-like circuit style can be used. the output signals have a high level voltage equal to  $V_{DD} - V_T$ . the control signals of the multiplexers should have full-rail swing. Whenever a full-rail

$R_S$  needed if can be generated with the double-raf circuit of Fig:7.18. The feedback PMOS transistor  $R_S$  needed to restore the high level when only a single-raf exists. The layout of such an adder is regular. Only three cells of the 1st, 2nd, 3rd BFs have to be drawn. Fig:7.19 illustrates the layout of a 4-bit block in  $0.8 \mu m$  design rules.





## Low-voltage, Low-power Design Techniques :-

Low-power design is a necessity today in all integrated circuits. As companies started packing more and more features and applications on the battery-operated devices, battery backup time became very important. Power consumption slowly became an increasingly important criterion from the customers. There are efforts to reduce dynamic as well as static power consumption. So a lot of low power design techniques started to get employed during the chip design process to reduce both static and dynamic power consumption. There are five types of low power design techniques:

- Clock Gating.
- Power Gating.
- Dynamic voltage and frequency Scaling.
- RPS
- Save and Restore power Gating.

### i) Clock Gating:-

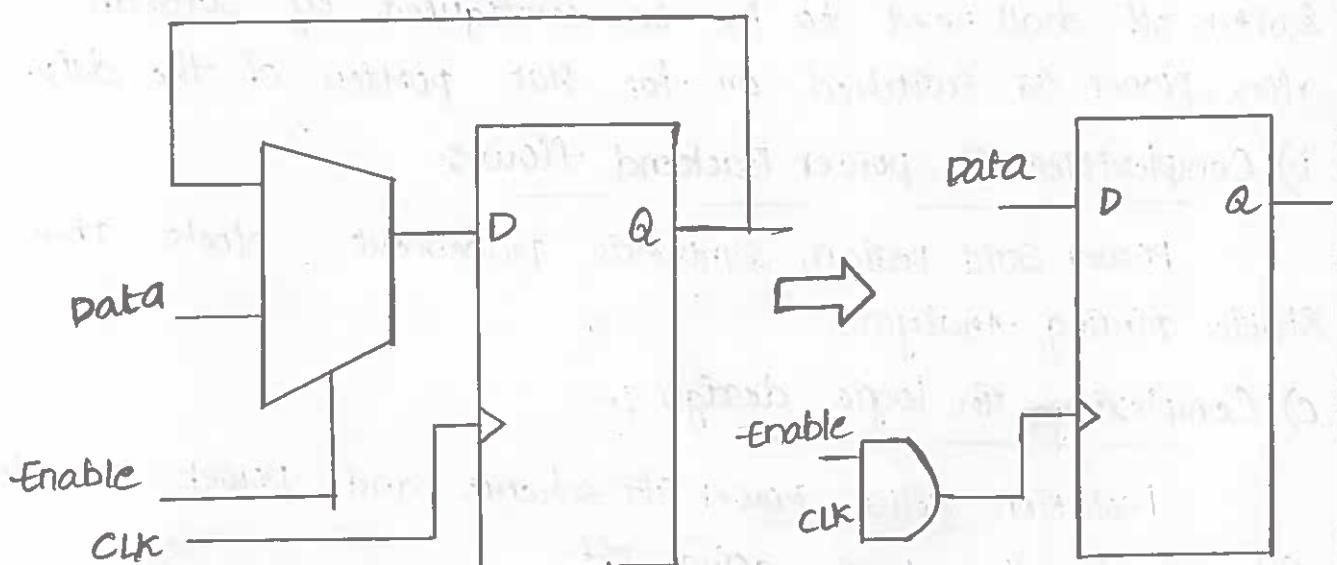


fig:1 above, shows a simplistic clock gating scheme. Instead of using a simple AND gate companies use more complex cells to avoid glitches.

This was by far one of the first techniques to save power [and it also results in saving some area due to sharing, however, it makes the design slightly difficult for timing and DFT]. The thought here is that if a common clock signal is going to hundreds of flops, a lot of them are retaining their old value, then we can gate off the clock to such ff's & they still retain their old value. This gating off results in lesser toggling in the clock path cells and thus saves dynamic power.

## a) Power Gating:-

In a device/application, a large portion of ICs is not in use for a reasonable amount of time. If the power to such a portion of IC can be switched off, it can save a lot of power. This saves static (leakage) power and can save some dynamic power as well where the clock was not gated off. There are some considerations for this technique.

### a) Complexity for Application Development:-

One of the penalties here is that portion being switch off shall need to be re-configured by software after power is switched on for that portion of the chip.

### b) Complexities in power Backend flows:-

power grid design, synthesis, placement, clock tree, static timing analysis.

### c) Complexity in logic design :-

Isolation cells, power off scheme and power on the scheme for the logic going off.

### Logic Design:-

We need to control the signal values, which are off from the gated off portion. As power is gated off, o/p's from the power gated domain shall start to float

to unknown values. However, there may be cells, in the continuous clock domain, sampling the signals from power Gated domain. Also, normally a static value is sufficient to continue the normal functioning of the continuous clock domain cells. So we need to isolate such floating values and give logic 0 (or) logic 1 to signals from the power Gated domain.

### Physical Design:-

#### i) Floorplanning

The size of both the power domains needs to be ascertained and floorplan to divide the area accordingly.

#### ii) Power Grids

Two power domains are created on the chip, namely the gated power domain and the continuous clock domain. Separate power grids are created for the two domains. The grids are shorted at a no. of points to ensure that current flows between the two domains & voltage differences are minimal. A power switch is used to switch off the power to power Gated domain. The enable for the switch is provided by the Lao power controller.

#### iii) Synthesis and placement

Normally the synthesis engines also support the automatic insertion of isolation cells, so they are required to be placed at the boundary of both the power domains. Synthesis has to take care that it does not do any cross border optimization between the two power domains.

#### iv) Clock tree insertion

has to ensure that all clock tree cells for flip-flops, in the continuous clock domain are placed in the same region and not in power Gated region.

IV) Any new o/p ports added to the power gated domain, as part of clock tree insertion or reset tree insertion should have isolation cells added.

### Isolation cells:-

These cells are like a buffer with enable. The enable signal when asserted makes the cell act like a buffer and when the enable signal is de-asserted, the cell gives a constant value of logic 0 (or) logic 1. The additional complexity is that such cells have two power domains, the power gated domain (input domain) & continuous clock domain (o/p domain).

### Low power controller:-

A low power design techniques controller module controls the sequence of events during power on and power off. To move the Gated power domain to a power-off state, the following is the sequence of events:

- The low power controller gets an indication to start the power off sequence. Normally such an indication is given just before the processor moves to its low power mode.
- Low power controller initiates the gating off of the clocks in the Gated power domain. The low power and clock controllers may switch the clocks of continuous power domain to slower frequencies to save further power.
- Enable signal is de-asserted, so that isolation cells give constant value output.
- The power switch cells, which short both the power grids, are opened so that the voltages in the two power domains are no longer shorted.
- Lastly, the power of the gated domain is switched off by the power switch.

### (3) Dynamic voltage and frequency islands:-

To save power another technique is to use multiple voltages and frequency domains. The portion of circuit requiring higher freq can be taken to a higher voltage for the time the high performance mode is required.

Now-a-days, almost all laptops use similar techniques to increase performance when connected with the mains by default. Normally the default power saving options, also reduce the performance level to save battery life. largely all such devices also provide programmable options so that the user can choose as per his or her preferences.

### 4) Retention Power Gating:-

This technique has an advantage over power Gating that, one can retain the value of state machine in the Gated power domain. However, this a more complex techniques and requires higher overhead in terms of area and implementation. Also, they have logic internally to store the state of flip-flop when the power is switched off.

Due to area and gating overheads, this is used in very special scenarios, where one needs a fast wake-up of the Gated domain and wants to avoid reconfiguring the Gated domain. Compare to other techniques, this method has the fastest wake-up time with state machine data being retained.

### 5) Save and Restore power Gating:-

This technique also has an advantage over power Gating that, one can retain the value of state machine in the Gated power domain. This low power design techniques also has an additional area overhead. In this technique, a RAM is added to the continuous power domain. Before moving into power off state, the Gated power domain shall save the state machine in RAM using an additional state machine.

This low power design technique has an overhead of area, the complexity of design takes time to go into power off state and also takes time while coming back from power off state.

### Low power Design Trends:-

Several techniques have been implemented over the past 30 years to solve these problems. Initially, scaling provided the benefit of lower power consumption and higher feature density. But eventually, clock scaling increased power density to the point where new techniques were needed. The major low power design techniques used in ICs include:

#### \* Dynamic Voltage Scaling:-

The voltage of logic levels can be scaled up or down as needed to control power consumption. Reducing the logic level ensures lower power consumption during switching.

#### \* Dynamic Frequency Scaling:-

The clock frequency and edge rate of system clock can be ramped up or down as needed.

#### \* Clock gating:-

This is used to cut off the system clock from certain logic blocks and prevent switching in logic circuits that are not manipulating data.

#### \* Substrate bias control:-

This is used alongside voltage scaling to control the threshold for entering either linear or saturation regions in MOSFETs that make up logic circuits. This technique is sometimes called back biasing, where a voltage is applied to the substrate regions in CMOS buffers to increase or decrease the logic state threshold voltages and reduce leakage current.