Module 3

Scaling of MOS Circuits

- High density chips in VLSI technology requires packing density of MOSFETS used is high and also the size of the transistors to be as small as possible. This reduction of size i.e. the dimension of MOSFET is referred as 'scaling'.
- It is expected that characteristics of the transistors will change with scaling and physical limitations will restrict the extent of scaling.
- Microelectronic technology is characterized in terms of indicators or figures of merit. The common figure of merits are
    - Minimum feature size
    - Number of gates on one chip
    - Power dissipation
    - Maximum operational frequency
    - Die size
    - Production cost
- The figure of merits can be improved by shrinking the dimensions of transistors, interconnections and the separation between features, adjusting doping levels and supply voltages.
- There are 2 types of size reduction/scaling strategies/ scaling models
    1. Full scaling or constant-field scaling.
    2. Constant-voltage scaling.
- Recently combined voltage and dimension model is presented. It is also called as 'Lateral scaling'
- Two scaling factors $1/\alpha$ and $1/\beta$ are used.
- $1/\beta$ is chosen as the scaling factor for supply voltage V_{DD} and gate oxide thickness D
- $1/\alpha$ is used as scaling factor for other linear dimensions
- Scaling theory indicates that the characteristics of MOS devices can be maintained and the basic operational characteristics have to be preserved if the parameters of a device are scaled in accordance to a given criteria.

    - Constant field scaling: scaled device is obtained by applying dimension less factor to
        - All dimensions
        - Device voltages
        - Concentration densities

Scaling factors for device parameters:

The device dimensions are shown in Fig. 3.1

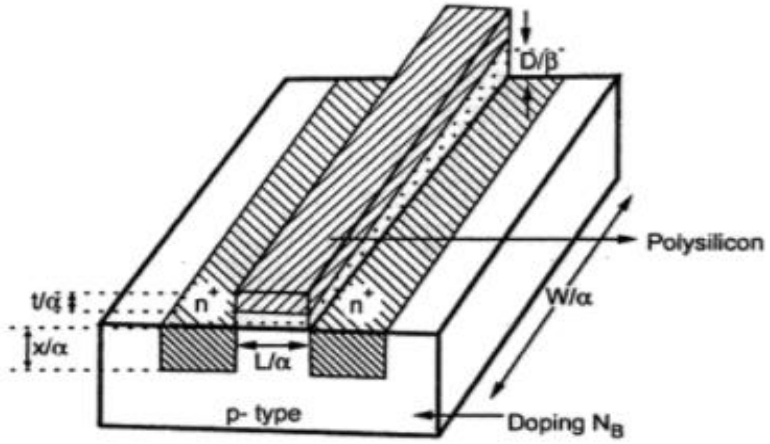


Fig. 3.1 scaled nMOS transistor

1. Gate Area A_g

$$A_g = L\ W$$

L and W defines channel length and width respectively. They are scaled by factor $1/\alpha$.

Thus $A_g$is scaled by $1/\alpha^2$.

2. Gate Capacitance per Unit Area C_o or C_{ox}

$$C_o = \frac{\varepsilon ox}{D}$$

$\varepsilon_{ox}$is the permittivity of gate oxide and D is the thickness of gate oxide (thinox).
Thus $C_o$is scaled by $\frac{1}{1/\beta} = \beta$

3. Gate Capacitance C_g

Gate capacitance is given by, $C_g = C_o L\ W$
Thus $C_g$is scaled by $\beta * 1/\alpha^2 = \beta/\alpha^2$

4. Parasitic Capacitance C_x

$C_x$is proportional to A_x/d
Where d depletion width around source or drain, it is scaled by $1/\alpha$. A_x is area of depletion region around source and drain, it is scaled by $1/\alpha^2$
Thus C_x is scaled by $(1/\alpha^2)/(1/\alpha) = 1/\alpha$

5. Carrier Density in channel Q_{on}

$Q_{on}$is the average charge per unit area in the channel in the 'on' state.
$Q_{on} = C_o V_{gs}$ [C_o is scaled by β and $V_{gs}$is scaled by $1/\beta$]
Thus $Q_{on}$is scaled by $\beta * 1/\beta = 1$

6. Channel Resistance R_{on}

$$R_{on} = \frac{L}{W} * 1/Q_{on}\mu$$

L is scaled by 1/α, W is scaled by 1/α, $Q_{on}$is scaled as 1, μ is carrier mobility in the channel and is a constant.
Thus R_{on} is scaled by (1/α)/(1/α)*1/1=1

7. Gate Delay T_d

$T_d$is proportional to R_{on}. C_g
R_{on} is scaled to 1 and C_g is scaled by β/α^2
Thus T_d is scaled to $1*\beta/\alpha^2 = \beta/\alpha^2$

8. Maximum Operating Frequency f_o

$f_o = W/L *(\mu C_o V_{DD})/C_g$
or$f_o$is inversely proportional to T_d
Thus $f_o$is scaled as $1/(\beta/\alpha^2) = \alpha^2/\beta$

9. Saturation Current I_{dss}

$$I_{dss} = \frac{C_o \mu}{2} * \frac{W}{L} * (V_{gs} - V_t)^2$$

C_o is scaled by β, μ and 2 are constants, W and L is scaled by 1/α, V_{gs} and $V_t$both voltages are scaled by 1/β.
Thus $I_{dss}$is scaled by $\beta*(1/\beta)^2 = 1/\beta$

10. Current Density J

$$J = \frac{Idss}{A}$$

A is the cross sectional area of channel in 'on' state and is scaled by $1/\alpha^2$ and I_{dss} is scaled by 1/β.
Thus J is scaled by $(1/\beta)/(1/\alpha^2) = \alpha^2/\beta$

11. Switching Energy per gate E_g

$$E_g = \frac{1}{2} C_g V_{DD}^2$$

C_g is scaled by β/α^2 and V_{DD} voltage is scaled by 1/β.
ThusE_g is scaled by $(\beta/\alpha^2)*(1/\beta)^2 = 1/\alpha^2\beta$

12. Power Dissipation Per Gate P_g

P_g comprises 2 components, i.e., $P_g = P_{gs} + P_{gd}$
$P_{gs}$is the static component, given by $P_{gs} = (V_{DD})^2/R_{on}$
$P_{gd}$is the dynamic component given by $P_{gd} = E_g * f_o$
$V_{DD}$is scaled by 1/β, $R_{on}$is scaled by 1
E_g is scaled by $1/\alpha^2\beta$, $f_o$is scaled by α^2/β
Thus $P_{gs}$and P_{gd} both are scaled by $(1/\beta)^2$

Thus P_g is scaled by $1/\beta^2$

13. Power Dissipation Per Unit Area P_a

$P_a$is defined as scaled by

$$P_a = \frac{P_g}{A_g} = \frac{\left(\frac{1}{\beta^2}\right)}{\left(\frac{1}{\alpha^2}\right)} = \frac{\alpha^2}{\beta^2}$$

14. Power-Speed Product P_T

$P_T = P_g * T_d$

P_g is scaled by $1/\beta^2$ and T_d is scaled by β/α^2

Thus $P_T$is scaled by $(1/\beta^2)*(\beta/\alpha^2) = 1/\alpha^2\beta$

Summary of Scaling effects of all the 3 models is given in the table below

Parameters		*Combined V and D*	*Constant E*	*Constant V*
V_{DD}	Supply voltage	$1/\beta$	$1/\alpha$	1
L	Channel length	$1/\alpha$	$1/\alpha$	$1/\alpha$
W	Channel width	$1/\alpha$	$1/\alpha$	$1/\alpha$
D	Gate oxide thickness	$1/\beta$	$1/\alpha$	1
A_g	Gate area	$1/\alpha^2$	$1/\alpha^2$	$1/\alpha^2$
C_0(or C_{ox})	Gate C per unit area	β	α	1
C_g	Gate capacitance	β/α^2	$1/\alpha$	$1/\alpha^2$
C_x	Parasitic capacitance	$1/\alpha$	$1/\alpha$	$1/\alpha$
Q_{on}	Carrier density	1	1	1
R_{on}	Channel resistance	1	1	1
I_{dss}	Saturation current	$1/\beta$	$1/\alpha$	1
A_c	Conductor X-section area	$1/\alpha^2$	$1/\alpha^2$	$1/\alpha^2$
I	Current density	α^2/β	α	α^2
V_g	Logic 1 level	$1/\beta$	$1/\alpha$	1
E_g	Switching energy	$1/\alpha^2.\beta$	$1/\alpha^3$	$1/\alpha^2$
P_g	Power dispn per gate	$1/\beta^2$	$1/\alpha^2$	1
N	Gates per unit area	α^2	α^2	α^2
P_a	Power dispn per unit area	α^2/β^2	1	α^2
T_d	Gate delay	β/α^2	$1/\alpha$	$1/\alpha^2$
f_0	Max. operating frequency	α^2/β	α	α^2
P_T	Power-speed product	$1/\alpha^2.\beta$	$1/\alpha^3$	$1/\alpha^2$

Constant E:$\beta = \alpha$; Constant V:$\beta = 1$

Subsystem Design Processes

Here the chapter deals with design of a digital system using a top down approach. The system considered is a 4 bit microprocessor.

The microprocessor includes ALU, control unit, I/O unit and memory. Here only ALU or data path is considered. The data path by itself is further divided into subsystem and in this 'shifter' unit is considered.

General Considerations:

- ✓ The considerations provide ways of handling problems, provides way of designing and realizing systems which are too complex
- ✓ Also help in understanding and appreciating technologies. The considerations are as follows:
    - Lower unit cost : with different approaches available for same requirement lower unit cost is appreciable
    - Higher reliability: high levels of system integrations greatly reduces interconnections and this in turn provides good reliability.
    - Lower power dissipation, lower weight and lower volume: in comparison with other approaches.
    - Better performance: particularly in terms of speed power product.
    - Enhanced repeatability: if there are fewer process to be controlled in the whole system or very large part to be realized on a single chip, this reduces the repeatability which is appreciable
    - Possibility of reduced design/development periods: for more complex systems reduced development time is appreciable.

- ✓ Some Problems related with VLSI design are

1. How to design complex systems in a reasonable time and reasonable effort.
2. The nature of architectures best suited to take full advantage of VLSI and the technology
3. The testability of large/complex systems once implemented on silicon

For the problem seen the solution is as follows:

Problem 1 & 3 are greatly reduced if two aspects are followed.

- a) Top-down design approach with adequate CAD tools
- b) Partitioning the system sensibly
- c) Aiming for simple interconnections
- d) High regularity within subsystem
- e) Generate and then verify each section of the design
- Devote significant portion of total chip area to test and diagnostic facility

Problem 2 can be solved by

- Select architectures that allow design objectives and high regularity in realization

Illustration of Design Processes:

- Structured design begins with the concept of hierarchy. This involves dividing any complex function into less complex sub-functions. This can be done until bottom level referred to leaf cells is reached.
- This process is known as top-down design
- As a systems complexity increases, its organization changes as different factors become relevant to its creation
- Coupling can be used as a measure the sub-modules interconnection. Clever system aims at minimum sub-module interconnection resulting in independent design.
- Concurrency should be exploited so that all gates on the chip do useful work most of the time
- As technology is changing fast, the adaptation to a new process must occur in a short time.

General Arrangement of a 4-bit Arithmetic Processor:

The basic architecture of microprocessor is shown below.

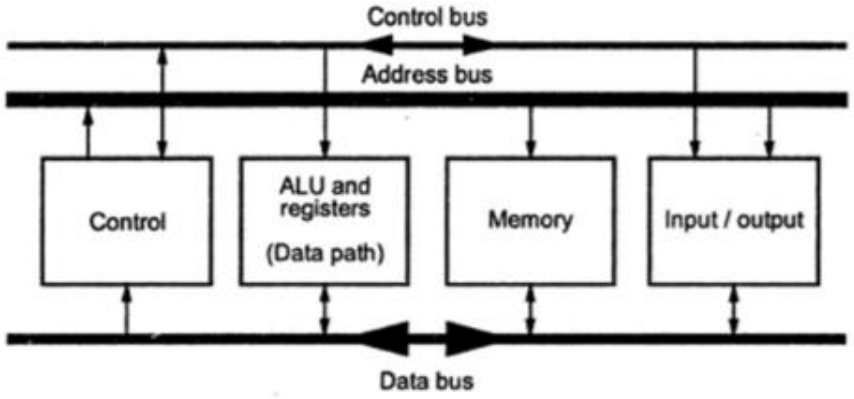


Fig. Basic digital processor structure

- It has a unit which processes data when applied at one port and gives output at second port.
- It is also possible that both data ports can be combined to form a single bidirectional port if storage facility is available in the data path.

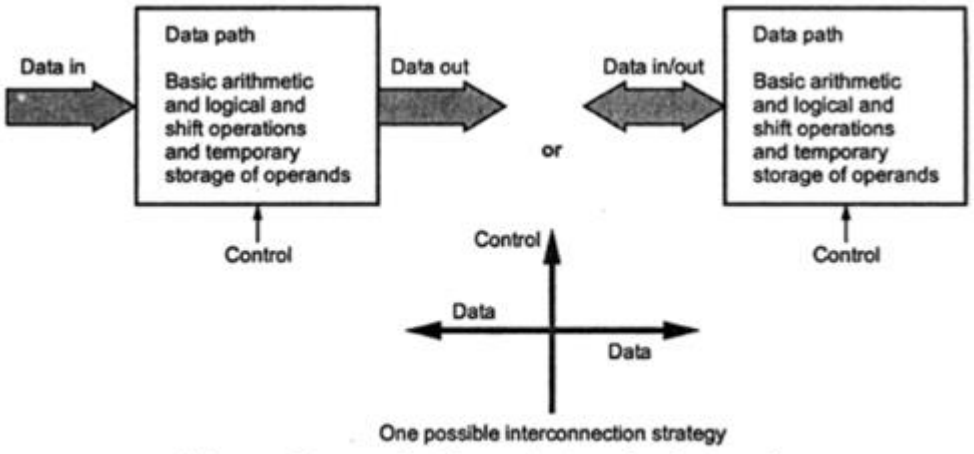
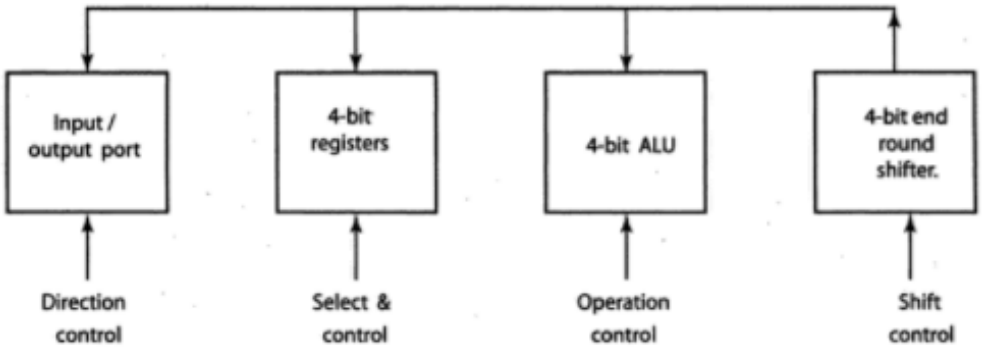


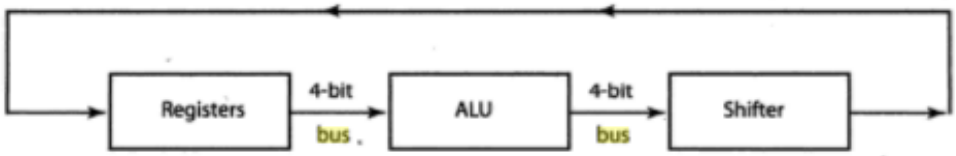
Fig. Communications strategy for data path

- The data path can be decomposed into having main subunits as it will be helpful in deciding the possible floor plan.



- The sub units can be linked with different bus architecture. It can be either one-bus, two-bus or three-bus architecture.

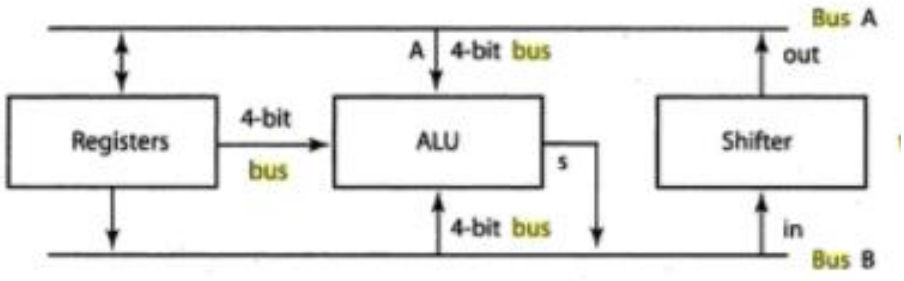
One bus architecture:



The sequence is:

1. First operand is moved from register to ALU and stored there.
2. Second operand is moved from register to ALU where operands are added (subtraction or any other arithmetic operation) and result is stored in ALU
3. The result is then passed through shifter and stored in register.
4. The process takes 3 clock cycles and this be fastened by using two bus architecture.

Two bus architecture:

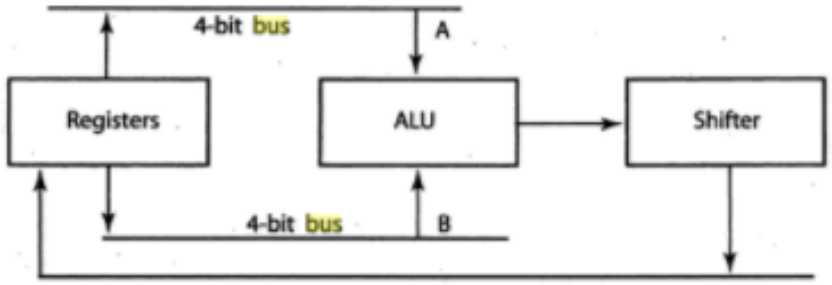


The sequence is:

1. Both operands (A & B) are sent from register(s) to ALU & are operated upon, result S in ALU.
2. Result is passed through the shifter & stored in registers.

3. This requires 2 clock cycle.

Three bus architecture:



1. Both operands (A & B) are sent from registers, operated in the ALU and result which is shifted is returned to another register. All these happen in same clock period.

❖ During the design process, care must be taken for allocating layers to various data path and guidelines. The guidelines are as follows:
- Metal can cross poly or diffusion
- Poly crossing diffusion form a transistor.
- Whenever lines touch on the same level an interconnection is formed
- Simple contacts can be used to join diffusion or poly to metal.
- Buried contacts or a butting contacts can be used to join diffusion and poly
- If 2nd metal layer is available, this can cross over any layer and can be used for power rails.
- Two metal layers may be joined using a via
- Each layer has particular electrical properties which must be taken into account
- For CMOS layouts, p-and n-diffusion wires must not directly join each other nor may they cross either a p-well or an n-well boundary

Design of 4 bit shifter

- Ageneral purpose n-bit shifter should be capable of shifting n incoming data up to n-1 place either in a right or leftdirection.
- Shifting should take place in ‘end-around’ basis i.e., any bit shifted out at one end of a data word will be shifted in at the other end of the word. Thus the problem of right shift or left shift can be easily eased.
- It can be analyzed that for a 4-bit word, that a 1-bit shift right is equivalent to a 3-bit shift left and a 2-bit shift right is equivalent to a 2-bit left etc. Hence, the design of either shift right or left can be done. Here the design is of shift right by 0, 1, 2, or 3 places.
- The shifter must have:
  - input from a four line parallel data bus
  - Four output lines for the shifted data
  - The input data can be transferred to output lines using shift from 0, 1, 2 to 3 bits
- While designing the strategy should be decided. The chosen strategy here is data flow direction is horizontal and control signal flow direction is vertical.

- To meet the criteria a 4 × 4 crossbar switch is used. The MOS switch implementation of 4 × 4 crossbar switch is shown in the Fig below.

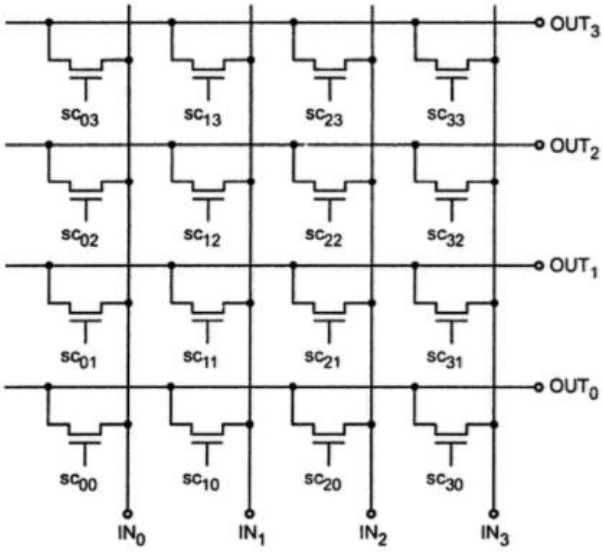


Fig. 4×4 crossbar switch

- To drive each cross bar switch - 16 control signals SC_{00} to SC_{15} are provided to each transistor switch.
- Arrangement is general and can be expanded to accommodate n-bit inputs/outputs.
- In this arrangement any input can be connected to any or all the outputs.
- If all switches are closed all inputs are connected to all outputs and it forms a short circuit.
- As it needs 16 control signal it increases the complexity and to reduce the complexity, the switch gates are coupled in groups (in this case it is grouped into 4)
- Here 4 groups of 4 is formed which corresponds to shift 0, shift 1, shift 2 and shift 3 bits.This arrangement is called 'barrel shifter'. (In this only 4 control signals is needed)

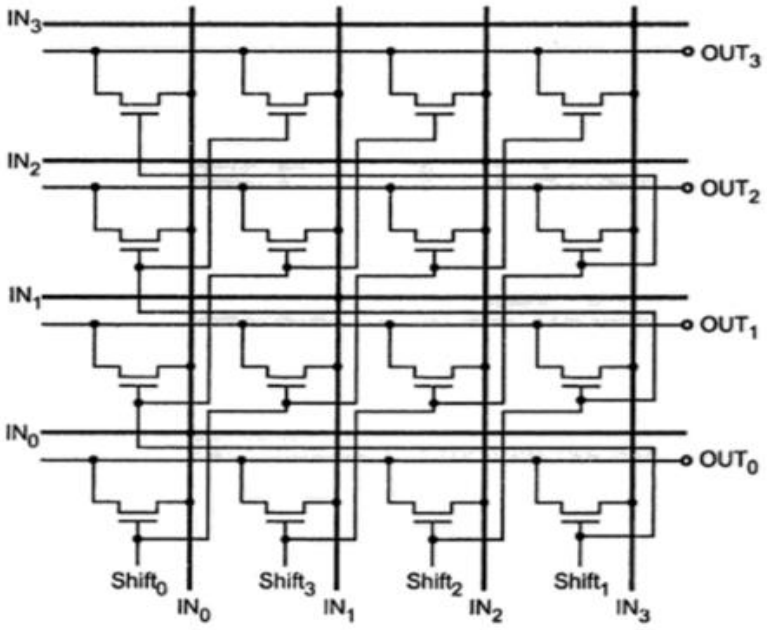
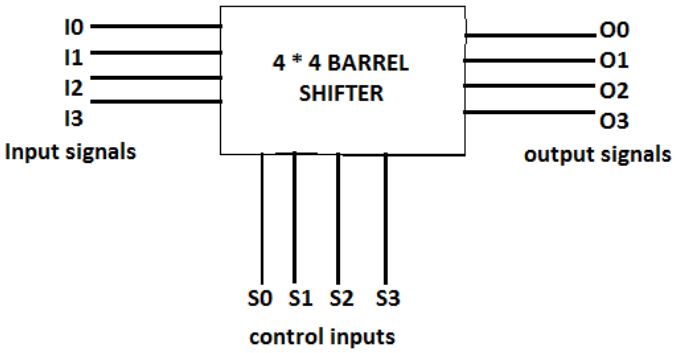


Fig. 4×4 barrel shifter

- The interbus switches have their gate inputs connected in a staircase fashion in groups of four and there are now four shift control inputs which must be mutually exclusive in the active state.
- CMOS transmission gates may be used in place of the simple pass transistor switches if appropriate. Barrel shifter connects the input lines representing a word to a group of output lines with the required shift determined by its control inputs (sh0, sh1, sh2, sh3). Control inputs also determine the direction of the shift. If input word has n – bits and shifts from 0 to n-1 bit positions are to be implemented.



SH0	SH1	SH2	SH3	SH4
O0	O1	O2	O3	O0
O1	O2	O3	O2	O1
O2	O3	O0	O1	O2
O3	O0	O1	O0	O3

Block diagram of barrel shifter and shifting of data is shown in the table.

Illustration of the Design Process

Regularity: It is an essentiality of any design. It reduces design efforts required for a system. Regularity of any particular design can be gauged as

$$Regularity = \frac{Total\ number\ of\ transistors\ on\ the\ chip}{Number\ of\ transistor\ circuits\ that\ must\ be\ designed\ in\ detail}$$

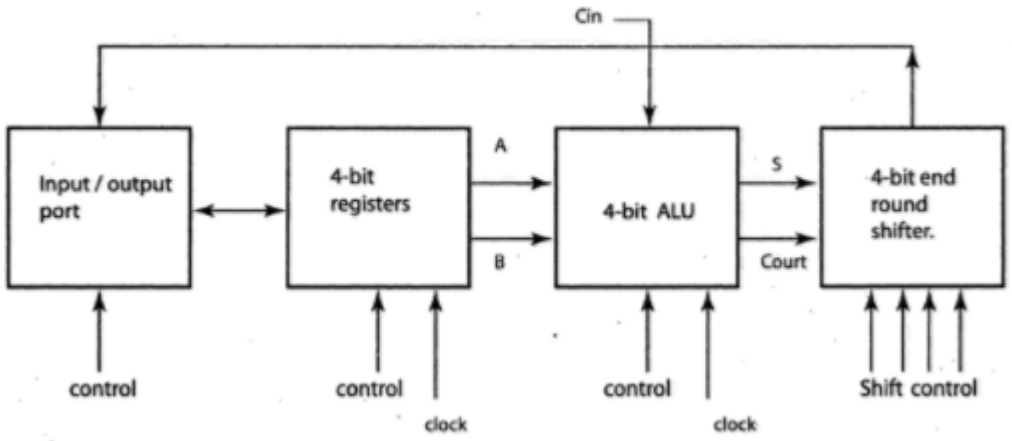
- For 4×4 bit barrel shifter, regularity factor is given by

  $$Regularity = \frac{16}{1}$$

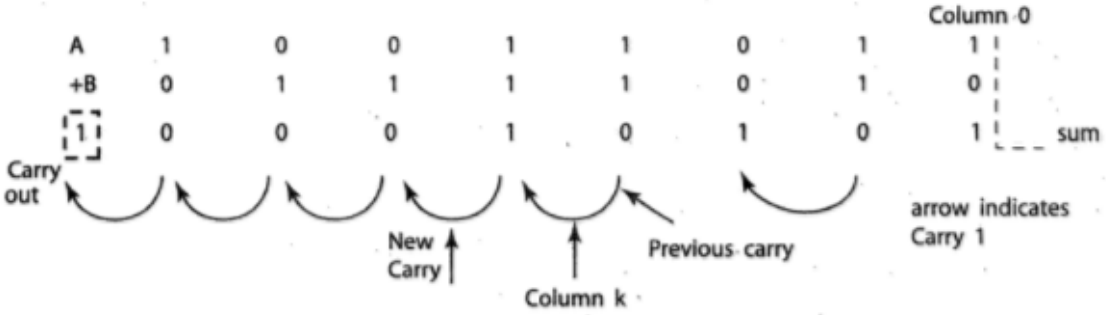
  In case of barrel shifter it needs only one transistor designing and the same can be applied to all the 16 transistors.
- Higher the regularity lesser will be the design effort. Good system design achieve regularity factor of 50 or 100.

Design of ALU subsystem

- Heart of the ALU is the adder circuit
- 4 bit adder will perform the sum of 2 4-bit number and here it is assumed that parallel operation form is done.
- The input to the adder needs two 4-bit buses, a single 4 bit is needed to move data from adder to shifter and other another 4-bit bus for shifting output back to register array. It also provide ‘carry out’ and possible ‘carry in’ signal
- The block diagram of 4 bit carry adder is shown below



- Considering an example of binary arithmetic operation.



- In any column we observe, there are 3 input. This number represents 3 inputs i.e., the corresponding input bits and previous carry/carry in
- If we consider kth column, it includes A_k, B_k representing input bits. C_{k-1} represents previous carry
- There are 2 outputs, sum and new carry. i.e., in Kth column S_k and C_k respectively.

Inputs			Outputs	
A_k	B_k	C_{k-1}	S_k	C_k
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- The sum carry equation we get is as follows:
- Sum $S_k = H_k C'_{k-1} + H_k' C_{k-1}$

  New carry $C_k = A_k B_k + H_k C_{k-1}$

  Where H_k is the half sum given by $H_k = A'_k B_k + A_k B'_k$

- The above sum and new carry equation can be implemented using 'AND-OR' or using 'EX-OR' gates. But in VLSI this implementation becomes complicated and alternate method is to find a standard element through which the above equations can be implemented.
- Analyzing the table of full adder again, we can see that

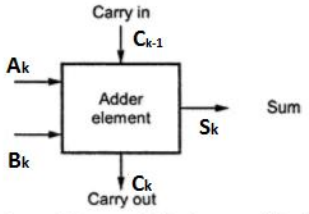
  For S_k, if $A_k = B_k$ then $S_k = C_{k-1}$

  else $S_k = C'_{k-1}$

  For C_k if $A_k = B_k$ then $C_k = A_k = B_k$

  else $C_k = C_{k-1}$

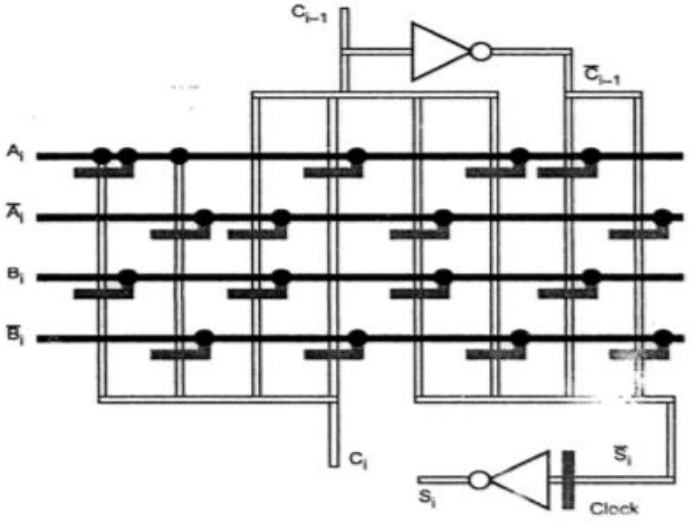
- A standard adder element (1 bit) can be implemented as shown below



n such elements would be cascaded to form an *n*-bit adder.

Adder element

- To form n-bit adder, n adder elements must be cascaded with carry-out of 1 element with carry-in of next most significant bit.
- The adder element can be implemented using multiplexer either in nMOS or CMOS technology. This method of implementation is easy to follow and results obtained are in the required forms.
- Implementation of adder using MUX (pass transistor) is shown in the last page. Below figure shows the stick diagram of the same.

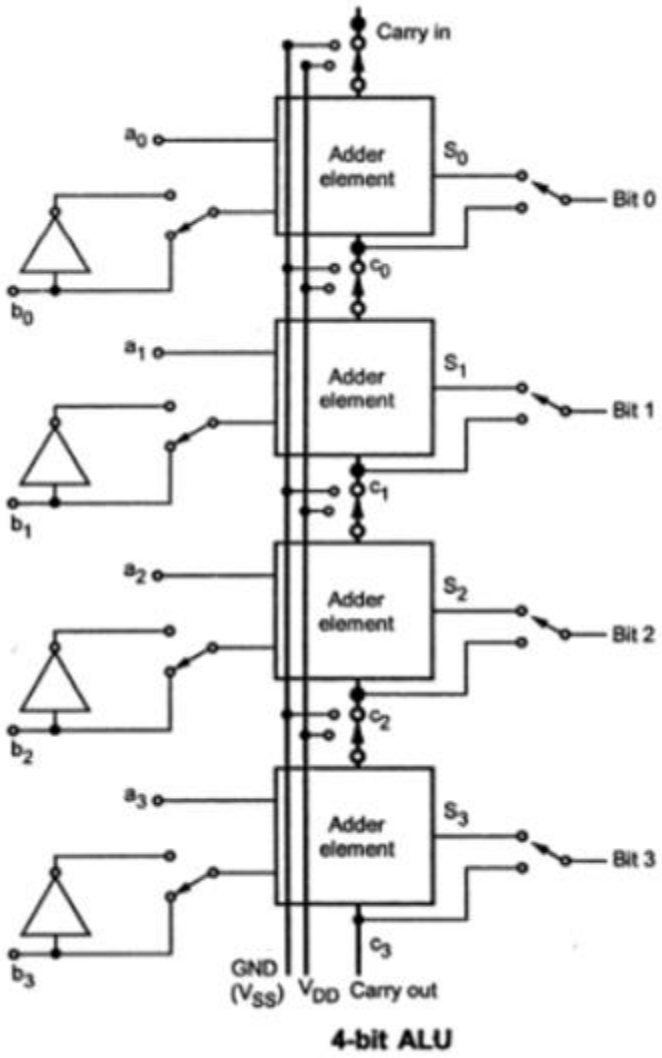


Multiplexer (n-switches)-based adder logic with stored and buffered sum output

Implementing ALU function with adder element:

- An ALU must be able to add and subtract two binary numbers, perform logical operations such as AND, OR and Equality (EX-OR) functions.
- Subtraction can be easily implemented using the adder element. Suppose we need find A – B, then
  - Compliment B and add 1 to it to get B'
  - Add this to A i.e., A+B'
  - This gives subtraction using adder
- Here compliment for subtraction can be served from 'logical compliment' (negate)
- In order to keep architecture as simple as possible it would be better if all the logical operations can be performed on the adder.
- Considering the possibility
  We have the sum and new carry equations as
  Sum $S_k= H_kC'_{k-1}+ H_k' C_{k-1}$
  New carry $C_k=A_k B_k + H_k C_{k-1}$
  Where $H_k =A'_k B_k+ A_kB'_k$

- If in S_k, if C_{k-1} is held at logic 0 then the expression would be
  - $S_k= H_k .1+ H_k' 0$
  - $S_k= H_k = A'_k B_k+ A_kB'_k$
  - This represents the EX-OR function/operation
- If in S_k, if C_{k-1} is held at logic 1 then the expression would be
  - $S_k= H_k .0+ H_k'. 1$
  - $S_k= H'_k = A'_kB'_k+ A_k B_k$
  - This represents the EX-NOR function/operation
- If in C_k, if C_{k-1} is held at logic 0 then the expression would be
  - $C_k=A_k B_k + H_k0$
  - $C_k=A_k B_k$
  - This represents the AND function/operation
- If in C_k, if C_{k-1} is held at logic 1 then the expression would be
  - $C_k=A_k B_k + H_k. 1=A_k B_k + H_k$
  - $C_k=A_k B_k + A'_k B_k+ A_kB'_k$
  - $C_k= B_k+ A_kB'_k$
  - $C_k= B_k+ A_k$
  - This represents the OR function/operation
- Thus with suitable switching of carry line between adder element will give ALU logical functions.
- One such arrangement for both arithmetic and logical function is shown in figure..



4-bit ALU

Further considerations of Adders:

- Simple and direct implementation of adder circuit was observed now alternate form of adder equations will be observed.
- In the adder carry will be propagated when either $A_k$or B_k is high. Thus this equation can be written as

  $P_k = A'_k B_k + A_k B'_k = A_k XOR B_k$
- Also carry will be generated when both A_k and B_k are high. This equation can be written as

  $G_k = A_k B_k$
- With this the expression for sum $S_k$and carry C_k the expression can be written as

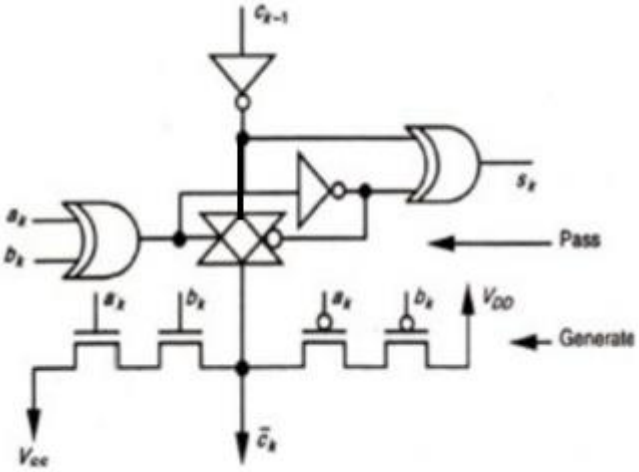
  $S_k = A'_k B_k C'_{k-1} + A_k B'_k C'_{k-1} + A'_k B'_k C_{k-1} + A_k B_k C_{k-1}$

  $\boxed{S_k = A_k XOR B_k XOR C_{k-1}}$

  And $C_k = A_k B_k + A'_k B_k C_{k-1} + A_k B'_k C_{k-1}$

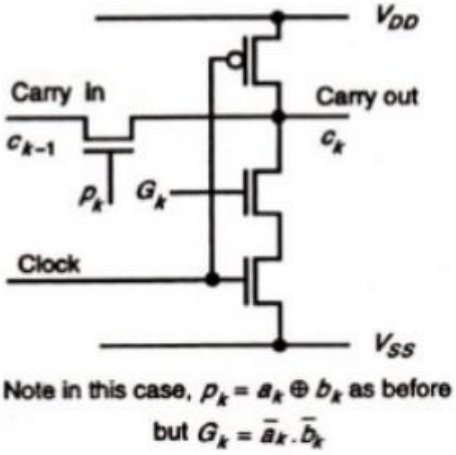
  $\boxed{C_k = G_k + P_k C_{k-1}}$

- Thus adder based on pass/generate carry concept is shown in the Fig
- Using this circuit carry can either be propagated to next stage or generated.



- Here propagate signal from ex-or gate is used to drive the transmission gate which acts like a controlling signal to propagate the carry signal. Only when P signal is high TG will be enabled and propagates C_{k-1}
- When P signal is 0 hen depending on the input A and B carry will be generated.
- The nMOS and pMOS pass transistor will generate the carry signal. (When A=B=1, carry is generated and when A=B=0 carry is not generated.)
- This is a direct realization which leads to the concept of carry chain and leads to popular arrangement known as Manchester carry chain.

Manchester Carry Chain:



Note in this case, $p_k = a_k \oplus b_k$ as before
but $G_k = \bar{a}_k . \bar{b}_k$

- It is a fast adder circuit, it is a carry propagate adder in which the delay taken by the carry to reach the last stage of output is reduced.
- As seen in the previous concept of passing the carry the transmission gate, the path can be precharged by the clock
- Then the path can be gated by the n-type pass transistor.
- When clock is 0, output will be charged to logic high due to pMOS. When clock is 1 pMOS will be off. If $P_k$is high, then the carry will be propagate.
- If P_k is 0, then C_{k-1} will not be propagated.

- Depending on inputs at G_k,(A_k= B_k =1) carry will be generated or (A_k= B_k =0) no carry generation.
- Even though Manchester carry cells are faster while cascading delay is observed. Cascading is done by connecting pass transistor in series.
- As n pass transistor is cascaded the delay also increases as square of n. thus in order to reduce delay buffers are included at after every 4 chain as shown in the block diagram

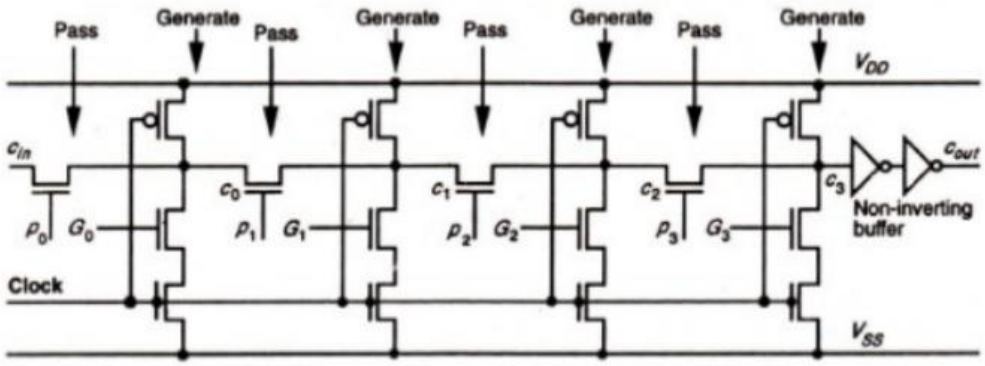


Fig. Cascading of Manchester carry adder

Adder Enhancement Techniques:

- In any adder element the previous carry bit is necessary to compute its own sum and carry out bits.
- For smaller adder (n<8) the delay observed for carry to propagate in ripple carry adder is small but at the n increases the delay is more and hence the time to find sum also increases.
- Thus some special techniques are used to improve the time taken for addition.
- This includes 3 methods
    - Carry Select Adder
    - Carry Skip Adder
    - Carry Look Ahead Adder

Note: All the above 3 techniques is a modification of ripple carry adder.

Carry Select Adder:

- In carry select adder the adder will be divided into group.
- It requires two identical parallel adders in each group except the least significant group as seen in the block diagram.
- Each group generates a group carry. Here, two sums are generated simultaneously. One sum assumes that the carry in is equal to one as the other assumes that the carry in is equal to zero.
- A mux is used to select valid result.
- It can be observed that the group carries logic increases rapidly when more high-order groups are added to the total adder length.
- This complexity can be decreased, with a subsequent increase in the delay, by partitioning a long adder into sections, with four groups per section, similar to the CLA adder.

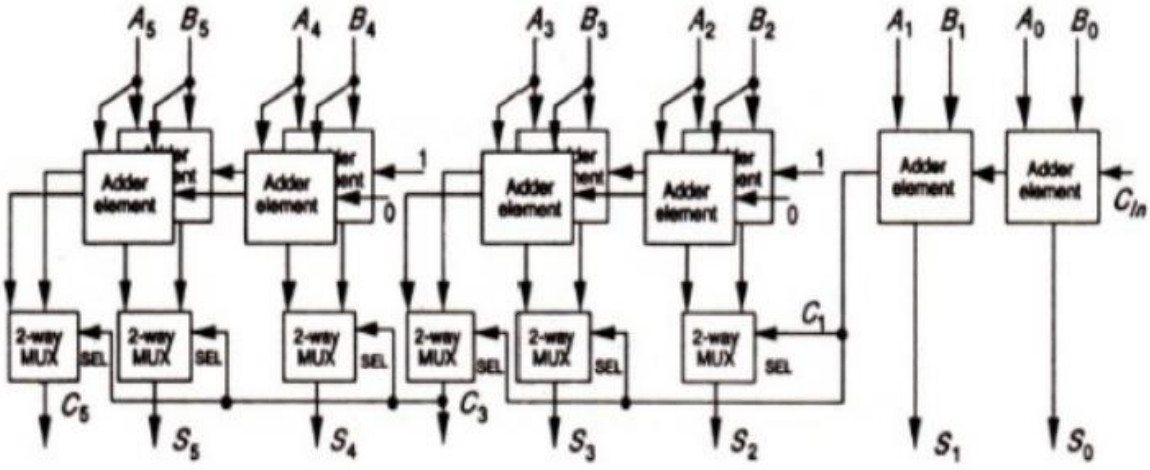


Fig. Block diagram of Carry Skip Adder

Optimization of Carry Skip Adder:

- For n bit ripple carry adder, the computational time T is given by
  T = K1n , where K1is the delay through 1 adder cell
- As carry select adder is a modification of carry skip which is having 2 adder in each block i.e., each block has 2 parallel paths. Thus computation time T becomes
  T = K1n/2 + K2 where K2 is the time needed by the mux to select the actual carry output.
- If there are many multiplexer then ripple through effect of carry is observed in mux rather than in the carry chain. Thus optimum value of mux should be selected for the size of each block.
- Suppose if there is an n-bit adder divided into M-blocks and each block contains P adder cells (n = M.P). The computation time for overall carry has 2 components
  - Propagation delay through the first block
  - Propagation delay through mux
    Thus, T = PK1 + (M – 1) K2
    Minimum value for T is seen when M=$\sqrt{nK1/K2}$

Note: except the first group all other groups have mux. If M is the number of groups the number of mux for n bit adder will be (M-1) and delay observed is (M -1)K2.

Carry Skip Adder:

- In a Ripple Carry Adder, if the input bits Ai and Bi are different for all position i, then the carry signal is propagated at all positions (thus never generated), and the addition is completed when the carry signal has propagated through the whole adder.
- In this case, the Ripple Carry Adder is as slow as it is large. Actually, Ripple Carry Adders are fast only for some configurations of the input words, where carry signals are generated at some positions
- Carry Skip Adders take advantage both of the generation or the propagation of the carry signal.
- They are divided into blocks, where a special circuit detects quickly if all the bits to be added are different. This signal is called ‘block propagation signal’

- If in the block, if A & B bits differ then the output i.e., block propagation signal = 1. If it is 1 then carry entering the block can bypass and transmit the carry to the block through multiplexer.
- Fig below shows 24 bit adder with 4 blocks and each block has 6 adder elements.

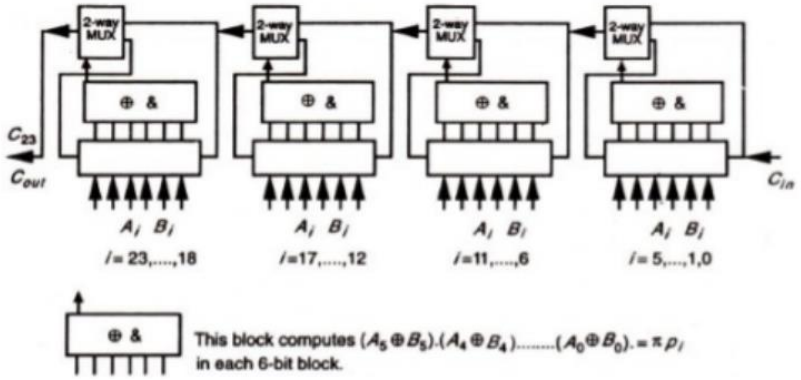


Fig. Block diagram of 24 bit carry skip adder

Optimization of carry skip adder:

- Let K1 represents the time needed by the carry signal to propagate through an adder cell, and K2 the time it needs to skip over one block. Suppose the N-bit Carry Skip Adder is divided into M blocks, and each block contains P adder cells. The actual addition time of a Ripple Carry Adder depends on the configuration of the input words. The completion time may be small but it also may reach the worst case, when all adder cells propagate the carry signal.
- In the same way, we must evaluate the worst carry propagation time for the Carry Skip Adder. One of the worst case of carry propagation is depicted in Figure.

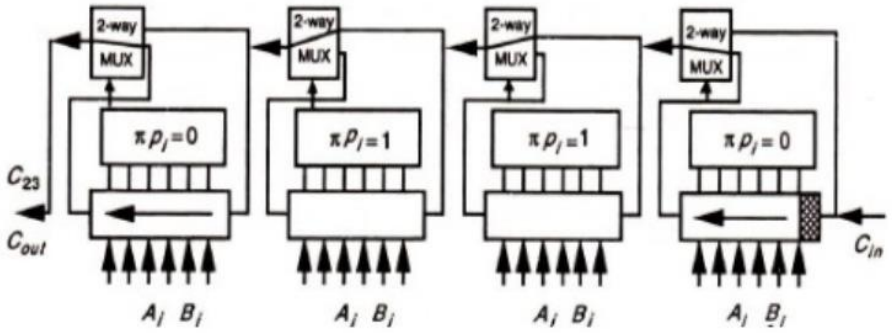


Fig. Worst case carry propagation for Carry Skip adder

- The configuration of the input words (for the worst case) is such that a carry signal is generated at the beginning of the first block. Then this carry signal is propagated by all the succeeding adder cells but the last which generates another carry signal.
- In the first and the last block the block propagation signal is equal to 0, so the entering carry signal is not transmitted to the next block.
- Consequently, in the first block, the last adder cells must wait for the carry signal, which comes from the first cell of the first block. When going out of the firstblock, the carry signal is distributed to the 2nd, 3rd and last block, where it propagates.

- In these blocks, the carry signals propagate almost simultaneously (we must account for the multiplexer delays).
- For the above condition the time to compute addition is given by

$$T = 2 (P – 1) K1 + (M – 2) K2$$

Note: in the first block except the first bit all other bits have for carry propagation hence it (P – 1) and as the same is observed for last block it is 2(P-1) and delay is K1. Also the input carry signal does not skip over the first and last block hence K2 is multiplied with (M-2).

Carry Look Ahead Adder:

- This is another method to improve the throughput time of adder.
- Here the prediction of carry is done and based on this designing is done.
- With the carry generate $G_k = A_kB_k$ and carry propagate $P_k = (A_k XOR B_k)C_{in}$ the carry propagation can be avoided and carry outputs can be calculated.
- If n (number of adder) is large, the carry output bits increases and expression larger and complex. Thus 'carry look ahead adder' and 'ripple carry adder' are clubbed together. This can be seen in the block diagram.

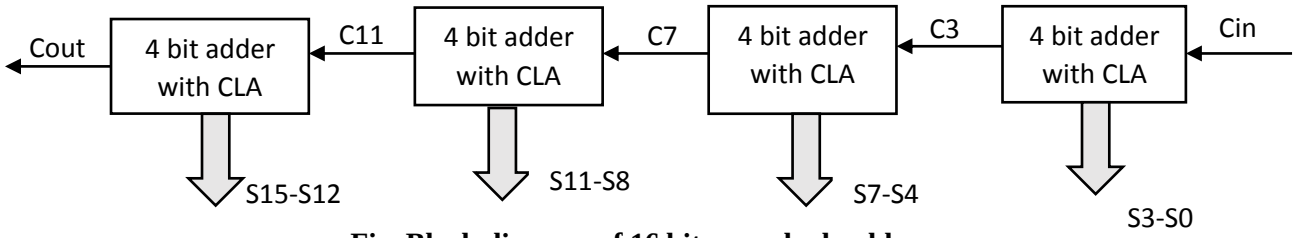


Fig. Block diagram of 16 bit carry look adder

- Considering the first block, carry out C3 can be calculated as follows
  $C_{out} = G_k + P_kC_{in}$
  Thus for the first bit in first adder block is given by

$$C_0 = G_0 + P_0C_{in}$$

Similarly for 2nd bit it is given by

$$C_1 = G_1 + P_1C_0 \quad \text{using } C_0 \text{ in the equation for } C_1$$
$$C_1 = G_1 + P_1G_0 + P_1P_0C_{in}$$

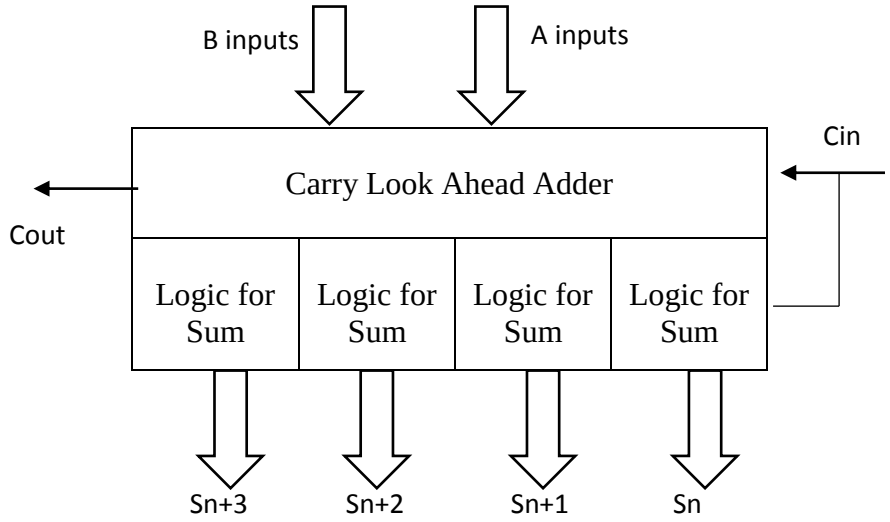
For 3rd bit it is given by

$$C_2 = G_2 + P_2 C_1 \quad \text{using } C_1 \text{ equation for } C_2$$
$$C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{in}$$

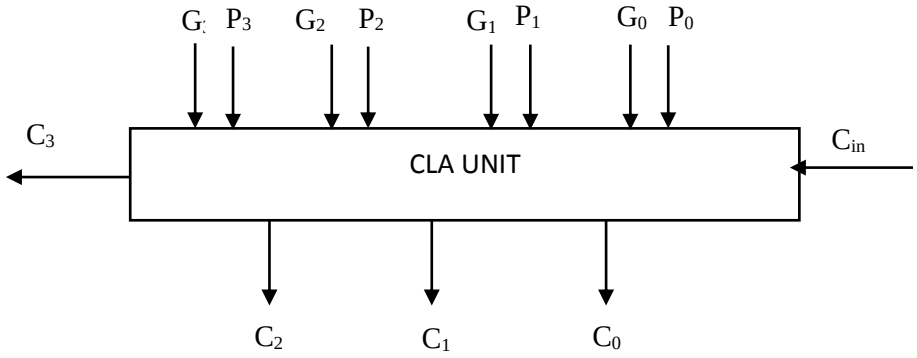
For the 4th bit it is given by

$$C_3 = G_3 + P_3 C_2 \quad \text{using } C_2 \text{ equation for } C_3$$
$$C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_{in}$$

- From the above equation we see that carry out C_3 can be calculated in prior without the need of need of carry being propagated through the adder cells.



- Each adder block has a 4 bit CLA(Carry Look Ahead) unit and is shown below.



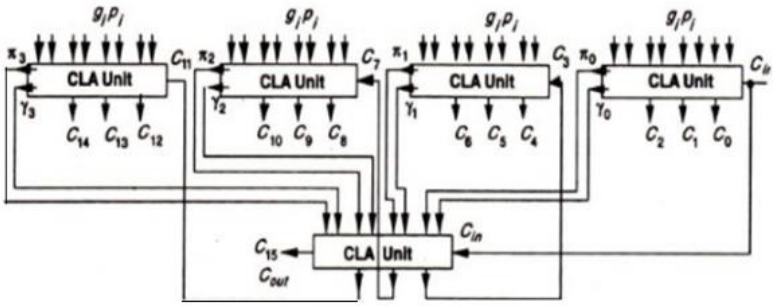
- From the CLA unit carry outputs C_0, C_1, C_2 and C_3 can be simultaneously determined. In this C_0, C_1, C_2 are required to determine the sum and will not ripple through. However C_3 will propagate to the next block.
- By observing the expression for C_0, C_1, C_2 and C_3 the last term in the equation become zero when $C_{in} = 0$.
- Thus to increase the speed, the expression for C_3 can be written as

$$C_3 = \gamma + \pi$$

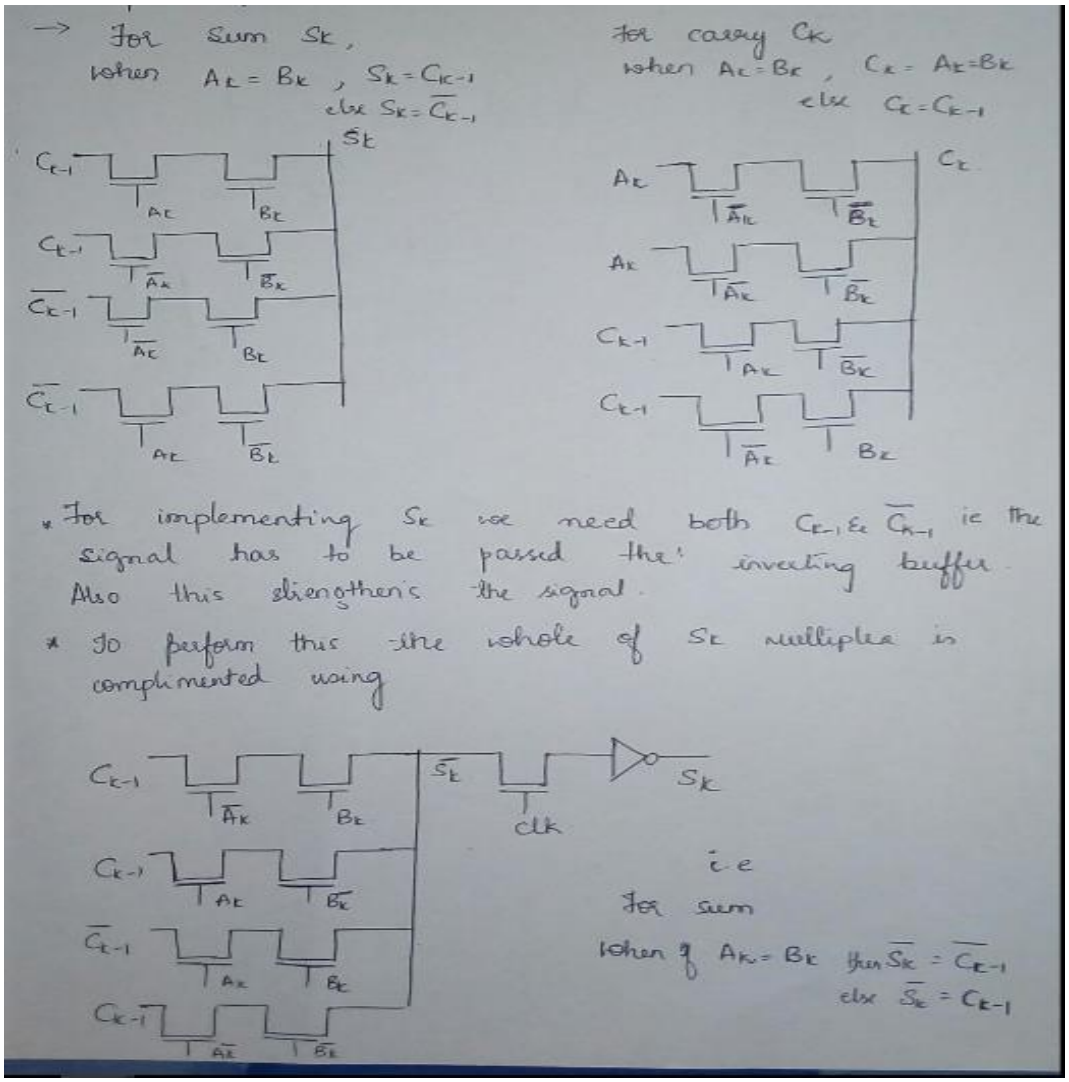
Where $\pi = P_3P_2P_1P_0C_{in}$ and

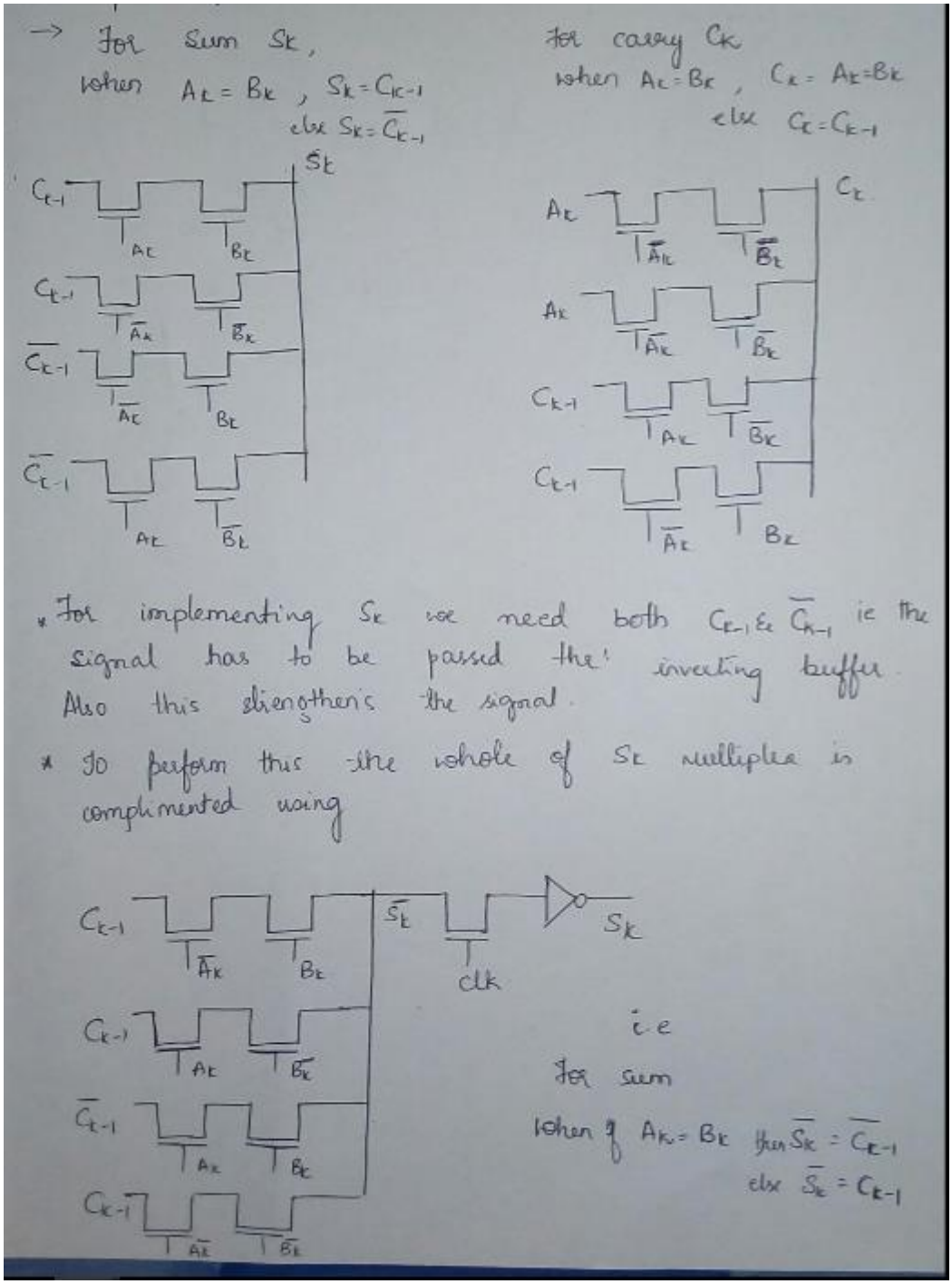
$$\gamma = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

- Here the term γ is independent of C_{in} and π becomes 0 when $C_{in} = 0$. Hence the speed of adder can be minimized by using these 2 defined functions γ and π.
- The implementation is shown in the block diagram. The carry outputs C3, C7 and C11 are propagated sequentially. This propagation delay can be reduced by using γ and π outputs, where C3, C7 and C11 are computed seperately.



16-bit, 4X4 block Carry look-ahead adder unit





Implementation of Adder using MUX.