Systematic Study of High Speed Adders and their MCML Realizations

Dissertation submitted in

partial fulfilment of the requirement

for the award of the degree of

Master of Technology

in

VLSI and Embedded System Design

by

Radhika

University Roll No. 2K12/VLS/15

Under the Guidance of

Dr. Neeta Pandey

Associate Professor,

Electronics and Communication Engineering Department, DTU

2012-2014

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

DELHI – 110042, INDIA

Electronics and Communication Engineering Department

Delhi Technological University

Delhi-110042

www.dce.edu

CERTIFICATE

This is to certify that the dissertation titled "**Systematic Study of High Speed Adders and their MCML Realization**" is a bonafide record of work done by **Radhika, Roll No. 2K12/VLS/15** at **Delhi Technological University** for partial fulfilment of the requirements for the degree of Master of Technology in VLSI and Embedded System Design. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Date: _________

(Dr. Neeta Pandey)

Associate Professor

Department of Electronics and Communication Engineering

Delhi Technological University

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor **Dr. Neeta Pandey** for providing the opportunity of carrying out this project and being the guiding force behind it. I am deeply indebted to her for the support, advice and encouragement she gave without which the project could not have been a success.

Secondly, I am grateful to **Prof. Rajeev Kapoor**, HOD, Electronics and Communication Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University for providing the right academic resources and environment for this work to be carried out.

At last I would like to express sincere gratitude to my parents for constantly encouraging me during the completion of work.

(Radhika)

University Roll no: 2K12/VLS/15
M.Tech. (VLSI and Embedded System Design)
Department of Electronics & Communication Engineering
Delhi Technological University
Delhi – 110042

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

There is shift in trend to current mode circuits due to their inherent advantage of reduced power consumption and high speed performance. Recently MOS Current Mode Logic (MCML) has gained interest and is dealt in this thesis. This topology is suitable for high speed design in mixed signal environment. One of the most important and frequent used structure in numerous processors and digital filters are adders. The overall performance of the block is affected by the delay incurred in addition operation. Hence use of high speed adder structures is in need.

This thesis presents study of various adder structures and their implementation using MCML style. When the number of bits in the input words is large, the MCML square root carry select adder is a viable alternative to MCML ripple carry adder and therefore it is designed. Significant improvement in delay was obtained for 16-bit MCML square root adder. When the input word is small, parallel prefix adders are considered to be the fastest one and therefore MCML approach is utilized to design parallel-prefix adders. The computation speed can further be enhanced by using Ling's scheme, hence MCML Ling adders which can be used as a viable choice for MCML parallel-prefix adders has been designed. All these adders inherit the advantages of MCML circuit over CMOS style as internal part of the adder is constructed using MCML basic gates. The Ling adders are designed with Brent Kung, Kogge Stone, Sklansky, Han Carlson, Ladner Fischer and Knowles tree structures. Their performances have been compared with 16-bit parallel-prefix adders employing the same tree structures. The simulation results prove the high speed nature of MCML Ling Adders. The reduction in delay in MCML ling adders implemented with six different tree structures varied from 35.69% to 4.62% with maximum reduction observed in MCML Han Carlson Ling adder and minimum delay seen in MCML Kogge Stone Ling adder. The workability of all proposed adders is confirmed through PSPICE simulations using TSMC 180 nm CMOS technology parameters.

CHAPTER 1

INTRODUCTION

1.1 Motivation

The recent advancement in VLSI technology has supported brisk growth in the field of high speed portable electronic devices. Cellular phones, Laptops and personal desktop assistants are some common electronic devices that have become not only a part but also a need of modern era. The increase in demand for portable electronic devices has forced the evolution of low power building blocks that facilitate long battery life for the system. In contrast to it, increase in the complexity of the circuit along with its operating frequency to fulfil the needs of today's high performance applications demands very high speed circuits.

The data path of every microprocessor, data processing ASIC or digital signal processors is a crucial circuit component in terms of area, power dissipation and the operating speed of the processor. The base of addressing units and data path in the processor is arithmetic and logic unit which performs arithmetic operations and is made up of digital components like comparators, shifters, multipliers and adders. The fundamental operation seen in most of the arithmetic component is addition of binary numbers. Even operations like increment, decrement, magnitude comparison etc. are based on binary addition of operands. Hence, this signifies the importance of binary addition which is performed by binary adders. The hardware implementation of adders is also critical due to dependency of carry propagation and evaluation time on its length. The efficient hardware implementation of high speed adders in an IC is a key challenge and motivation for VLSI designers.

There are abundant adders available. But our motive is to design adders for high frequency applications as is the need of modern world. The conventional complimentary CMOS topology is not suitable for such high speed applications as the operating speed of this topology depends on its input frequency and also the circuit dissipates dynamic power due to switching. There are many alternative logic styles proposed in literature but recent trend of shifting to CM is gaining interest due to their high performance over conventional logic styles. MCML or MOS current mode logic is a promising logic style in mixed signal environment. MCML circuits do not dissipate dynamic power and also power dissipated by them is independent of their operating speed. They have reduced input and output voltage swing and therefore give fast response. These interesting features of MCML have captured the interest of VLSI designers to implement high performance digital circuits using it.

1.2 Review of Relevant Literature

There are numerous research papers available on varied adder design structures and also their implementation with many topologies. The progress of the same through years is elaborated in this section.

O.J Bedrij in 1962 [1] first proposed carry select adder in which sum was computed individually with two carry inputs and finally the sum selection was performed.

A VLSI implementation scheme to design an adder using CMOS technology was proposed in 1988 by V. G. Oklobdzija [2]. They simulated adders employing regular structure with carry look-ahead scheme.

In same year R. W. Doran [3] introduced Ling's approach to design adder with reduced CLA equation scheme and elaborated it to explore the extract of his scheme. From Ling’s manipulations he observed that various other variants of CLA were possible and that was explored by him.

A. Tyagi in 1990 proposed reduced area carry select adder scheme [4] where one of the carry chain of RCA is replaced by an OR gate per bit. Also, select prefix adder was introduced by replacing RCA of CSA with parallel prefix block.

In 1990, adaptively biased pseudo-NMOS logic was investigated by F. Lu and H. Samueli [5] and they implemented full adder using it. This technique was proved to be faster than conventional cascode voltage switch logic.

In 2011, hybrid adder scheme to enhance the speed of carry select adder was proposed by H. G. Tamar, A. G. Tamar, K. Hadidi and A. Khoei [6]. The hybrid adder replaced the internal RCA of carry select adder with carry look-ahead adder. Hence, the linear dependence on carry in RCA was removed thereby reducing the delay. Also, the designed adder consumes low power and occupies less area.

Subodh Wairya, Rajendra Kumar Nagaria and Sudarshan Tiwari in 2012 [7] carried out analysis of low voltage hybrid adders. They used MOS capacitor in the circuit to reduce power consumption and designed high performance adder structure using XOR-XNOR style.

In the same year, Deepa Yagain, Vijaya Krishna A and Akansha Baliga designed high speed adder structures [8]. They implemented improved logarithmic adders i.e. Ling adder using complimentary CMOS and transmission gates. Comparative study of these adder structures with CLAs were done along with IIR filter implementation using it.

Another hybrid carry select adder was proposed by S. Parmar and K. P. Singh in 2013 [9]. They designed modified carry select adder wherein one RCA with carry input one was replaced with binary excess one convertor (BEC). Hence, there was increase in speed and also reduction in area as BEC require less area as compared to RCA.

1.3 Research Objective

The base of taking up this work is the shift in trend to current mode circuits and the modern world need for high speed and low power structures. MCML being CM offers certain advantages over other traditional logic styles, therefore being an appropriate topology for high speed and low power digital circuit design. Hence this thesis proposes new design of various high speed adders using MCML topology thereby capturing the extract of above discussion. The objectives of carrying out this research work are;

- To study and analyse MCML parameters.
- To simulate basic MCML structures.
- In depth study of various parallel adders.
- To introduce new high speed adders implementation using a promising current mode technique i.e. MCML.
- Verification of proposed circuitry by performing PSPICE simulations.

The literature survey performed in previous section brings out the extensive use of high speed adders and their analysis using many topologies. MCML topology is recently proposed technique, hence not explored much. So exploring this topology for high speed digital circuit design is chief objective of this work. High speed adders find their extensive usage in high speed processors where accelerated performance of adders is essential. Hence, a new implementation of square root carry select adder using MCML is proposed. Carry look-ahead adders are known to be fastest adders that give logarithmic delay and MCML implementation of which is given in [10]. The delay of CLA's can further be reduced by using scheme proposed by ling [11]. Therefore, MCML parallel prefix adder using ling scheme is freshly proposed and implemented here. All proposed designs are verified by performing PSPICE simulations to check their functionality.

1.4 Structure of Thesis

The thesis is framed in five chapters. Here, in chapter 1 we have reviewed the basic literature related to adders and also discussed the motivation behind this work along with the objectives of this thesis. The chapter 2 describes basic MCML structure and the parameters to be considered while designing its circuits. It also presents a separate section where the functions of basic MCML gates are elaborated and their functional simulation is performed. The chapter 3 delivers background information on existing adder architectures. The simulated results of those adder structures using MCML gates are presented concurrently. A novel high speed MCML square root carry select adder is also proposed and verified in this chapter. Also the chapter introduces Carry look-ahead adders called logarithmic adders and ling adders but they are elaborated and designed in next chapter. Therefore, chapter 4 is all about parallel prefix adders where CLA's and Ling adders are implemented using parallel prefix tree. Six different tree structures namely Kogge Stone, Sklansky, Ladner Fischer, Han Carlson, Brent Kung and Knowles are described and used to form prefix structure. Hence, MCML parallel prefix Ling adders are proposed here and they are functionally verified through simulation. The resultant plots showing the high speed nature along with power dissipation is furnished. Finally, the conclusions from the varied study and simulation, in accordance with the thesis objectives are drawn in chapter 5.

CHAPTER 2

MCML AND ITS PARAMETERS

The high speed circuits design with conventional CMOS technology suffers from delay issue which affects the switching speed of the circuit. The propagation delay issue can be resolved by sizing the transistors i.e. Increasing the W/L ratio to get the fast switching response but this will result in higher power consumption. There are many techniques proposed in literature like CPL, DCVSL etc. that utilize differential signals. This provides the enhanced noise immunity and a compact structure which motivates its usage for high speed operation. MOS Current Mode Logic is also differential logic style and therefore it shares the entire advantages of differential logic operation. It has only static power dissipation i.e. current drawn from the supply is constant and independent of switching activity taking place in the circuit. These features of MCML make it an appropriate logic style in mixed signal environment [12].

This chapter describes the basics of MCML circuits and the parameters that are used to design them. The basics gates implementation is also presented and the results are displayed.

2.1 MCML Basics

The MCML is current mode logic and therefore shares all the advantages of current mode circuits. MCML is entirely differential logic; therefore it needs all signals along with their complements. It is based on the source coupled pair of NMOS transistors which forms pull down network and the circuit is biased by constant current source. MCML is built up of three blocks, as shown in Fig. 1. These blocks include the pull-up resistive load, the pull-down network (PDN) and a constant biasing current source (I_{Bias}) [13]. The PDN block implements the logic function and depending on the internal logic, the current is steered through one of the branch. The output voltage of the branch with no current flowing in it reaches VDD as it is pulled up through the load resistors. Whereas some voltage drop occurs in the other branch with current flowing through it, the output voltage drops to $V_{DD} - I_{Bias} . R_L$.

The core part of an MCML circuit is constructed through NMOS differential pairs in PDN. The use of only NMOS transistors for internal logic computation makes this topology inherently faster than other logic families. Also, MCML being CM logic style operates the internal logic in current domain i.e. analysis is done in the form of current, hence works at an accelerated speed. The differential nature of MCML makes it highly immune to common mode noise and also the use of differential signals in the circuitry for inputs and outputs lessen the switching noise. MCML does not offer full rail-to-rail swing and this reduced swing lessens the dynamic power dissipation of the circuit [14]. The power curve of MCML is almost flat over a wide frequency spectrum against the other logic styles where there is linear increase in power consumption with respect to frequency [15]. Therefore at high frequencies, the power consumption by MCML is comparatively lower than other topologies.

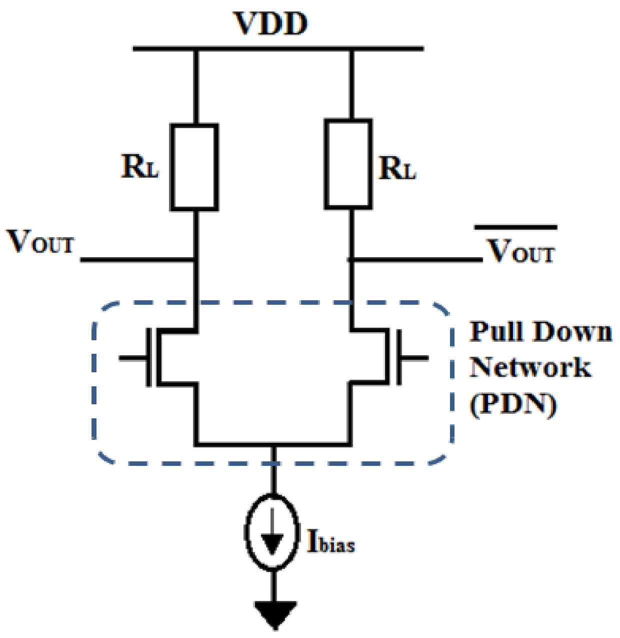


Fig. 2.1 Schematic of MCML

The differential MCML Inverter/Buffer is shown in fig.2.2. The resistive load in the circuit is modelled by equivalent PMOS transistors. The output voltage of the MCML gate is by the flow of drain current through the NMOS transistors in the differential pair

[16]. The input voltages $V_{IN}\ and\ \overline{V_{IN}}$ are applied to the gate terminal of the NMOS transistor M3-M4 while output voltages are extracted from their drain nodes. Hence, the differential input applied equals $V_{IN} - \overline{V_{IN}} = V_{ID}$.

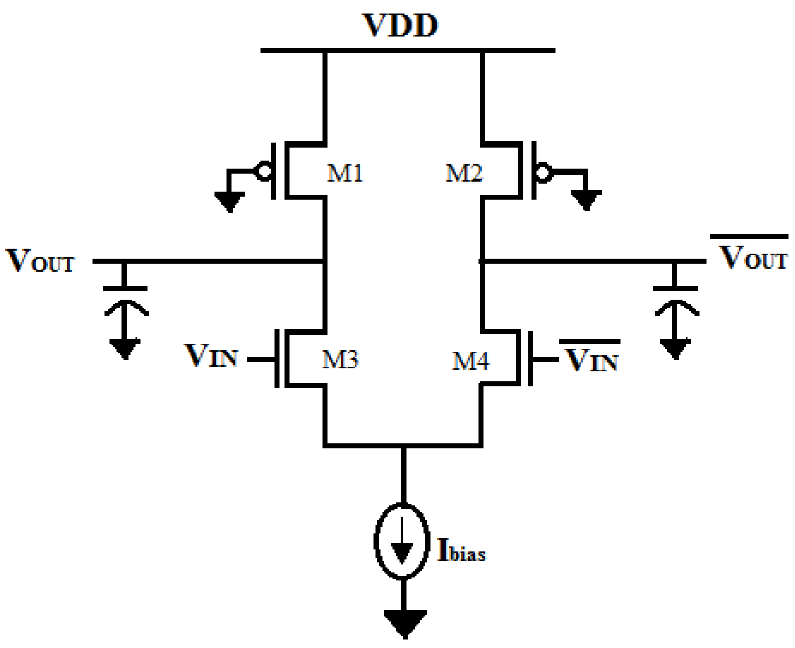


Fig. 2.2 Schematic of an MCML buffer/inverter

To bias NMOS transistor to operate in saturation region, the current through transistor M3 is derived to be [17].

$$\boldsymbol{I_{d3}} = \begin{cases} 0 & if\ V_{ID} < -\sqrt{\dfrac{2\,I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}} \\ I_{ds} = \dfrac{I_{Bias}}{2} + \dfrac{V_{ID}}{2}\sqrt{\mu_n C_{ox}\dfrac{W_n}{L_n} I_{Bias} - \left(\mu_n C_{ox}\dfrac{W_n}{L_n}\dfrac{V_{ID}}{2}\right)^2} & if\ |V_{ID}| \le -\sqrt{\dfrac{2\,I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}} \\ I_{Bias} & if\ \ V_{ID} > -\sqrt{\dfrac{2\,I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}} \end{cases} \quad (1.1)$$

$$\boldsymbol{I_{d4} = I_{Bias} - I_{d3}} \quad (1.2)$$

where I_{Bias} is the tail current generated by the buffer's current source, W_n and L_n are the channel width and length respectively of NMOS transistor M3, μ_n and Cox are device parameters i.e. NMOS carrier mobility and the oxide capacitance per unit area respectively.

When $V_{ID} > \sqrt{\frac{2I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}}$, the entire current is steered through one of the two output branches.

We can obtain the output voltage transfer characteristics of MCML buffer/ inverter by equivalent resistive load $\mathrm{R_L}$. The differential output voltage can be written as:

$$\mathrm{V_{OD}} = \mathrm{V_{OUT}} - \overline{\mathrm{V_{OUT}}} = -\mathrm{R_L}(I_{d3} - I_{d4}) \qquad (1.3)$$

First the transistor current equations were analysed and now the MCML inverter/ buffer's transfer characteristic is to be evaluated.

$$\boldsymbol{V_{OD}(V_{ID})} =$$

$$\begin{cases} \mathrm{R_L.I_{bias}} & if\ V_{ID} < -\sqrt{\frac{2\,I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}} \\ -V_{ID}\ \mathrm{R_L.I_{bias}}\sqrt{\mu_n C_{ox}\frac{W_n}{L_n}\mathrm{I_{Bias}} - \left(\mu_n C_{ox}\frac{W_n}{L_n}\frac{V_{ID}}{2}\right)^2} & if\ |V_{ID}| \le -\sqrt{\frac{2\,I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}} \\ -\mathrm{R_L.I_{bias}} & if\ V_{ID} > -\sqrt{\frac{2\,I_{Bias}}{\mu_n C_{ox}\frac{W_n}{L_n}}} \end{cases} \qquad (1.4)$$

The transfer characteristic of MCML inverter/buffer circuit pictured in fig 2.3 where Noise Margin (NM) is also shown which in explained in section 2.2.3. The load PMOS and tail NMOS transistors limit the voltage swing in MCML gate. To bias NMOS transistors in saturation region, it is required to keep $\mathrm{R_L.I_{bias}}$ low enough. This is obtained when the gate-drain voltage $\mathrm{V_{gd}}$ is kept lower than the transistor's threshold voltage [17], hence an upper limit is forced on voltage swing of $\mathrm{R_L.I_{bias}}$.

The logic swing is given as:

$$\mathrm{V_{gd}} = \mathrm{V_{DD}} - [\mathrm{V_{DD}} - \mathrm{R_L.I_{bias}}] = \mathrm{R_L.I_{bias}} \le \mathrm{V_{T,n}} \qquad (1.5)$$

The operation of MCML Buffer shown in fig. 2.2 is as follows. The complimentary signals $(V_{IN}, \overline{V_{IN}})$ are fed to the PDN consisting of two NMOS transistors (M3, M4).

When input V_{IN} is low, transistors M3 turns ON whereas transistor M4 turns OFF. Therefore, the bias current $\mathrm{I_{bias}}$ is steered through M3 thereby pulling down the output node to low voltage ($\mathrm{V_{OUT} = V_{DD} - R_L . I_{bias}}$).

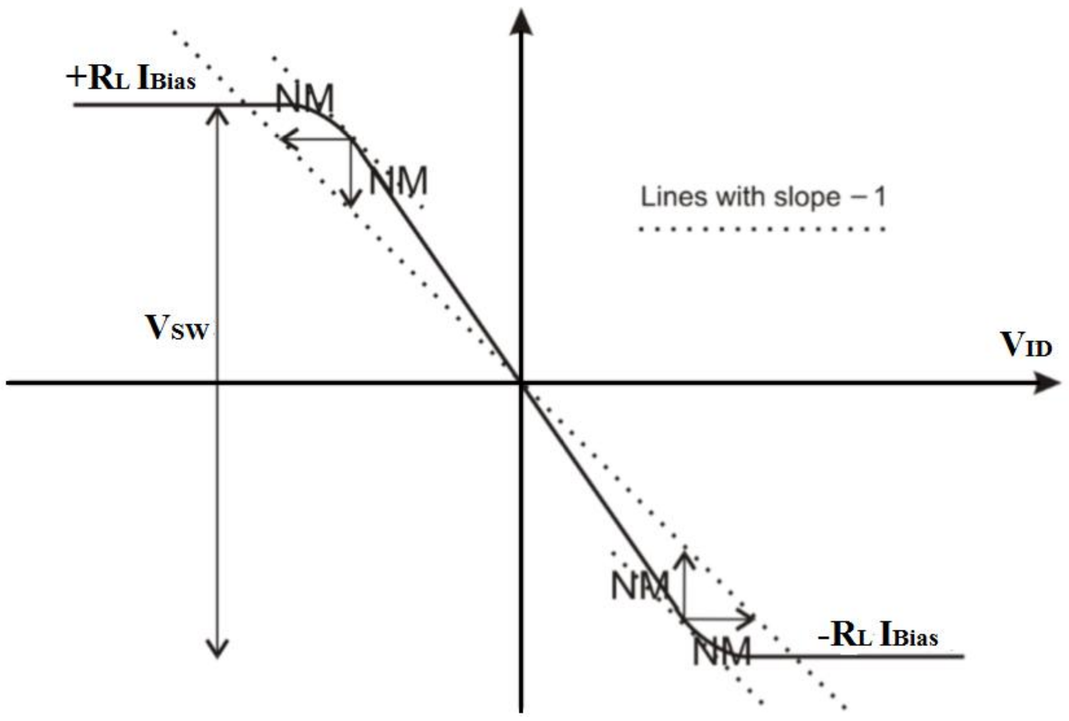


Fig. 2.3 Voltage transfer characteristics of MCML Inverter

The other branch with no current pulls up the output node to high voltage ($\mathrm{\overline{V_{OUT}} = V_{DD}}$) through PMOS transistors acting as load resistor. Hence, the overall differential output becomes low and is obtained as $\mathrm{V_{OD} = V_{OUT} - \overline{V_{OUT}} = -R_L . I_{bias} = -V_R}$. Conversely, for high input V_{IN}, current is steered through transistor M4 which makes the output node voltage low ($\mathrm{\overline{V_{OUT}} = V_{DD} - R_L . I_{bias}}$). The other branch is deprived of current flow in it and therefore the output node is pulled to ($\mathrm{V_{OUT} = V_{DD}}$). The overall differential output in this case becomes high ($\mathrm{V_{OD} = V_{OUT} - \overline{V_{OUT}} = R_L . I_{bias} = +V_R}$). Hence, the overall swing becomes $\mathrm{2. R_L . I_{bias}}$.

2.1.1 Static Modelling of PMOS load

The resistive load can be modelled by PMOS transistors. The grounded gate PMOS load transistors (M1, M2) can be used to replace of resistor and provide equivalent load resistive value. The PMOS transistors are always ON which results in constant static power dissipation in the circuit. The current to voltage conversion in the MCML is performed by these two PMOS transistors. Both these transistors have a source gate voltage equal to V_{DD} and a much smaller source-drain voltage [18]. Therefore the transistor operates in triode region, and can be modelled as an equivalent linear resistor, R_L [19].

It results in $R_L = \frac{R_{int}}{1-\frac{R_{DS}}{R_{int}}}$ (1.6)

where $R_{int} = \frac{1}{\mu_{eff,p} C_{ox} \frac{W_p}{L_p} (V_{DD}-|V_{T,p}|)}$ (1.7)

where R_{int} represents the "intrinsic" resistance of PMOS transistor biased in the triode region without taking parasitic drain/source resistance into account.

2.2 MCML Design Parameters

The parameters are used to design MCML gate and are used to characterize their performance. These parameters include Gain, Noise Margin, Voltage Swing Ratio, Power Dissipation and Delay. The variables used to design MCML gates are Voltage Swing, Bias current and Differential pair NMOS transistor dimensions W_n, L_n.

2.2.1 Voltage Swing:

The MCML topology offers major advantage of reducing the signal swing due to which the current required for charging and discharging the parasitic capacitances is also reduce that ultimately enhances the speed of operation. To function as logic circuit, the voltage swing of MCML at the input and output of the circuit should be sufficiently high to ensure the complete switching of the tail bias current to one of the two output branches [20]. In other words, the voltage swing at the output node, i.e.:

$V_{SW} = +\mathrm{V_R} - (-\mathrm{V_R}) = 2.(+\mathrm{V_R}) = 2. R_{L.} I_{Bias}$ (1.8)

should be high enough to completely switch the input differential pair of the next stage:

$V_{SW} = V_{SW,min}$ (1.9)

that is equivalent to say that the gain of each MCML circuit. It should be sufficiently high to use it as a logic circuit with adequate noise margin. The minimum voltage swing that is acceptable at the output of each MCML gate is $V_{SW,min}$, and it depends on the operating region of NMOS transistor [21],

$$V_{SW,min} = \sqrt{2}\, n\, V_{DS} \tag{1.10}$$

where n is the sub-threshold slope factor of NMOS transistor and V_{DS} is drain to source voltage of NMOS transistor.

2.2.2 Voltage Gain

The mid voltage swing gain, A_V is responsible for the gate robustness. To ensure gate stability and gate regeneration for next stages, the gain is chosen greater than one. The symmetrical property states that the logic threshold equals zero ($V_{LT} = 0$) and the small-signal voltage gain associated with it is $g_{m,n}.R_L$, (where $g_{m,n}$is the small-signal transconductance of NMOS transistor (M3-M4) of fig. 2.2 with $I_{D3} = \frac{I_{Bias}}{2}$ (i.e. the current through one of the branch) [18]. The gate is biased around logic threshold, therefore the input and output voltages will be $V_{IN} = \overline{V_{IN}} = V_{OUT} = \overline{V_{OUT}} = V_{DD} - \frac{\Delta V}{2}$ and $I_{D3} = \frac{I_{Bias}}{2}$ as given above, the voltage V_{DS} of transistor M3-M4 of fig. 2.2 is equal to their V_{GS}. Hence, the resulting expression of the voltage gain A_V is [16]:

$$A_V = g_m\, R_L = \Delta V \sqrt{\mu_{n,eff} C_{ox} \frac{W_n}{L_n} \frac{1}{I_{Bias}}} \tag{1.11}$$

Where $\Delta V = R_L.I_{Bias}$, g_m is transconductance of the differential NMOS pairs, C_{ox} is oxide capacitance of the MOS and $\mu_{n,eff}$ is the effective electron mobility of the differential pair.

2.2.3 Noise Margin

The Noise margin, NM, is also a measure of robustness of logic gate [22]. Generally, combinational logic circuits require nonnegative noise margin and sequential circuits needs positive noise margin (NM greater than zero).

The NM is equal to NML (for Low-Logic) and similarly to NMH (for High-Logic) due symmetrical property, which is defined as [21]:

$$NM_H = VO_{H,min} - VI_{H,min} \tag{1.12}$$

and

$$NM_L = VI_{L,max} - VO_{L,max} \tag{1.13}$$

where $VI_{L,max}$ and $VI_{H,min}$ are the input voltage values where ∂VO/∂VI = -1. $VO_{L,max}$ and $VO_{H,min}$ are the corresponding output voltages.

$VO_{L,max}$ = VO($VI_{H,min}$) and $VO_{H,min}$= VO($VI_{L,max}$). By differentiating VI and equating it to -1, $VI_{H,min}$ results is:

$$VI_{H,min} = \sqrt{\frac{2I_{Bias}}{\mu_{eff,n}C_{ox}\frac{Wn}{Ln}} - \frac{I_{Bias}}{2\mu_{eff,n}C_{ox}\frac{Wn}{Ln}}\frac{1}{A_V{}^2}\left(\sqrt{1+8A_V{}^2}+1\right)}$$

$$VI_H \approx \sqrt{\frac{2I_{Bias}}{\mu_{eff,n}C_{ox}\frac{Wn}{Ln}}}\left(1-\frac{1}{\sqrt{2A_V}}\right) \tag{1.14}$$

Where $A_V \gg {}^{1}/_{\sqrt{8}}$ can be assumed

Approximating $VO_{H,min}$ to -ΔV leads to the following expression of NM: [16]

$$NM = \Delta V\left(1-\frac{\sqrt{2}}{A_V}\sqrt{1-\frac{1}{\sqrt{2A_V}}}\right) \cong \left(1-\frac{\sqrt{2}}{A_V}\right) \tag{1.15}$$

2.2.4 Differential Pair Transistor Sizes

The parameters related to the designing of pull down network of MCML gate is transistor sizing which means setting an appropriate value for length and width of the MOS transistor. The choice of this depends on the designer but is of importance as it will directly or indirectly affect the performance of the circuit. Therefore, some restrictions are posed on transistor sizing. We can vary both length and width of each transistor but matching the same size for differential pair transistors is desirable. Also, the length of each transistor should be kept minimum as increasing it will outcome no benefit. So, the sizing of differential pair NMOS means to basically size the width of the transistor. The increase in width will enhance the gain but will also result in increment of capacitance (input and output). There is a trade-off in delay and voltage gain which signifies direct trade-off between performance and robustness respectively. Hence, minimum sized transistor should be used but also which has enough voltage gain.

In multi-level gates, there are many levels of differential pairs and therefore number of transistors to be sized is increased. There is also variation in voltage gain of the complete circuit and it is computed for worst case inputs. The formulation of design rules in this case is very difficult as the effects on gain depend on other levels of the PDN, hence a sort of interdependency observed among different levels. Under normal conditions, the width of the NMOS transistors slightly increases from bottom to the top of the hierarchy in PDN.

2.2.5 PMOS Load Transistor Sizing

The PMOS transistor acting as equivalent resistive load is hard to size as it is non linear and challenging task. There exist area trade-off and apart from it, other parameters which gets affected by PMOS transistor sizing are propagation delay (τ_{PD}), voltage gain (A_V) etc.

The increment in length of the PMOS devices (L_P) will increase the overall voltage gain of the circuit. This effect is strong for minimum length transistor and therefore, higher length transistors are preferred.

The device ratio i.e. the width to length ratio (W_P/L_P) of the transistors also imposes some impact listed below:

- The (W_P/L_P) ratio can be increased by either increasing width of PMOS W_P or by decreasing PMOS's length L_P and it has inverse impact on the equivalent resistance i.e. it reduces the effective load resistive value. Also, it results in improvement in propagation delay of the gate. When width of PMOS is increased, propagation delay stay's same but increases the capacitance at the output node. Rest of the impact of this ratio depends on the load capacitance.
- The increase in W_P/L_P ratio of PMOS load transistor decrements its $V_{D,sat}$ voltage and adds on non-linearity in the resistance. To get nominal SSR it is desirable to have small PMOS device ratio (less than 1) but that will have adverse effect in propagation delay. The choice of the individual width and length of load PMOS transistor is heavily dependent on minimum Rf_P voltage required to bias it.

Finally it can be concluded that W_P/L_P must be kept large to attain required DC resistance for the necessary voltage swing along with current but restricting the V_{sg} voltage to maximum 1.8V.

These constraints complicates the optimization of PMOS load transistor but generally we can it is desirable to keep minimum W_P fixed and vary L_P from minimum value to get higher current to larger value (around 1µm) to get small current.

2.2.6 NMOS Current Source Transistor Sizing

There is choice for sizing current source NMOS transistor for robustness or minimum area of the gate as there is trade-off between these two. The principal trade-off in the selection of the current source transistor sizes is between area and robustness. To get enhanced output impedance and reduced mismatch effects, the usage of NMOS transistor length greater than the minimum is desirable. Also the device ratio W/L of NMOS should be kept larger so that drain saturation voltage $V_{D,sat}$ is decreased and V_{DD} is reduced further. There is limit imposed on increasing the length and width of the NMOS transistor as it will result in expansion of gate’s area dramatically.

2.2.7 Delay analysis:

The NMOS transistors of PDN operate in saturation for maximum duration and have same source voltage for both input logic values which is fixed by the ON NMOS transistor. Thus, the modelling of the propagation delay,τ_{PD}, of the MCML inverter is done after its linearization around logic threshold $V_I = 0$ and by using half circuit concept due to symmetrical and differential nature of MCML gates [18]. The small-signal model depicting the half- circuit concept is shown in fig. 2.4.

By using open-circuit time constant method, the time constant (τ) of the circuit can be computed and results in delay equal to 0.69τ for a step input waveform and ignoring zero at high-frequencies. Hence, the propagation delay τ_{PD} of the MCML gate is given by: [23]

$$\tau_{PD} = 0.69 \times R_L(C_{gd,n} + C_{db,n} + C_{gd,p} + C_{db,p} + C_L) \qquad (1.19)$$

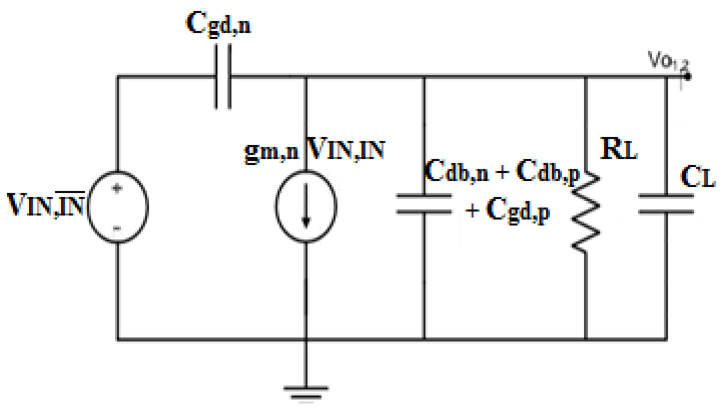


Fig. 2.4 Equivalent linear half circuit of the MCML Inverter

Where $C_{gd,n}$ and $C_{gd,p}$ are gate drain capacitance of NMOS and PMOS transistors respectively. $C_{db,n}$ and $C_{db,p}$ are the drain bulk capacitances of NMOS and PMOS transistors respectively and C_L is the external load capacitance.

2.2.8 Power Dissipation

MCML gates steers current through one of the branches of the gate. This current is constant giving rise to static power dissipation in the circuit given by:

$$P_D = V_{DD}\, I_{Bias} \tag{1.20}$$

Where I_{Bias} is the biasing current flowing in the circuit and V_{DD} is the supply voltage.

The dynamic power is not dissipated by MCML circuits as the outputs are differential therefore when one output is charging the output capacitance the other one is discharging an equal one.

2.3 Simulation Results of MCML Gates

The simulations throughout the thesis are performed in PSPICE using 0.18µm technology parameters. The load resistance R_L is of 4kΩ and the constant steering current, $I_{Bias.}$ is taken to be 50µA. So, this provides the voltage swing of 400mV. The simulation environments for the circuits are shown in table 2.1.

The basic gates used to implement the full adder are designed using MCML topology. These gates include MCML AND gate, MCML OR gate, MCML XOR gate, MCML multiplexer and also block generation circuit for parallel prefix adders discussed in chapter 4. These gates are simulated in PSPICE to check their functionality.

Table: 2.1 Simulation Environment

Technology	180nm
Temperature	27 ºC
Supply Voltage	1.8V
Output Load Capacitor	10fF

2.3.1 MCML INVETER gate

The MCML inverter circuit is shown in fig. 2.5. It is the simplest and most basic circuit of MCML gates. The differential inputs fed to the NMOS pair of the pull down network are A and its complement. The differential outputs are received at $V_O and \overline{V_O}$. The simulated MCML inverter output with resistive load is shown in fig. 2.6. The plot in fig. 2.6 (a) is showing the input fed to the gate and the plot of fig. 2.6(b) displays the respective output.

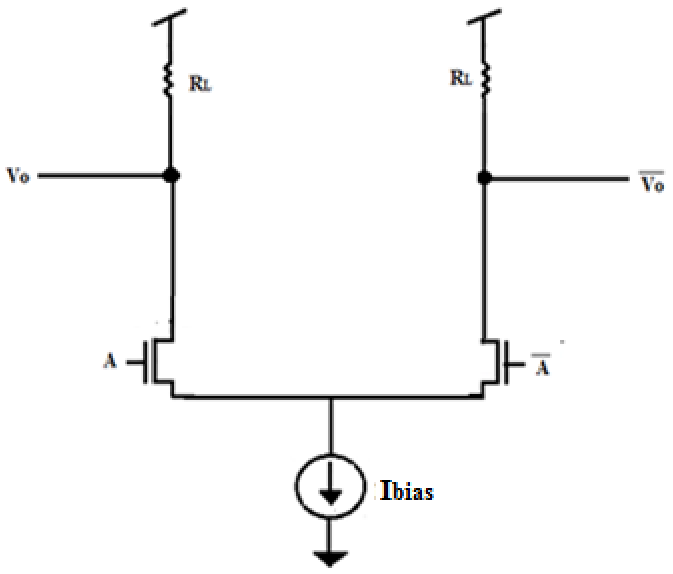
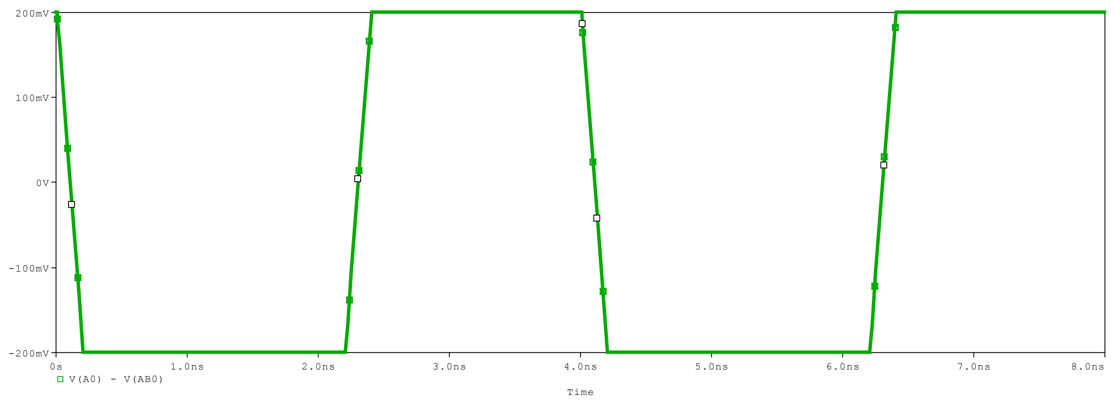
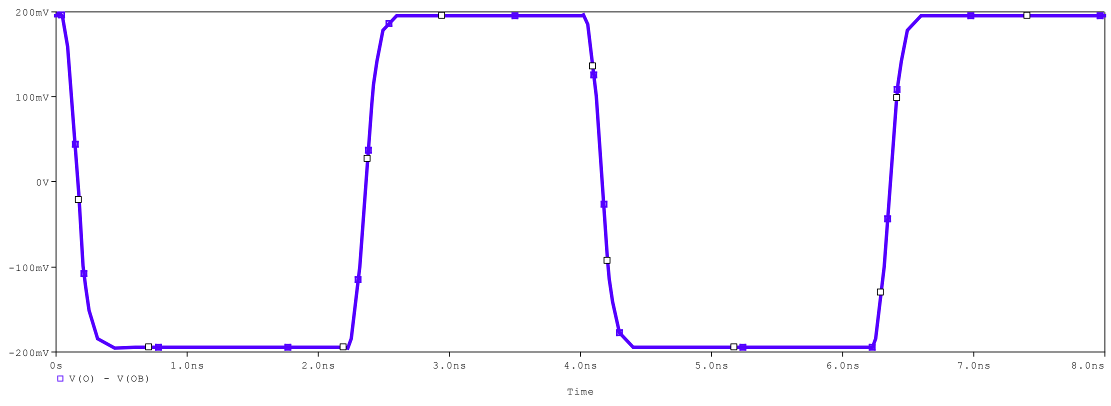


Fig. 2.5 MCML Inverter



(a)



(b)

Fig. 2.6 MCML inverter results (a) Input (A) (b) Output (O)

2.3.2 MCML AND gate

The circuit shown in fig. 2.7 is fed with differential inputs to the NMOS pair of the pull down network are A and B along with their complements and the differential output is obtained at $V_O and \overline{V_O}$. When A=0, NMOS with $\overline{A}$ at its gate terminal is switched ON due to which $V_O$is pulled down to ground and NMOS with A at its gate terminal is OFF due to which $\overline{V_O}$ is pulled up to VDD, hence differential output is LOW. When A=1, NMOS with $\overline{A}$ at its gate terminal is turned OFF and NMOS with A at its gate terminal is ON, in this case output depends upon the value of B. For A=1 and B=0, $V_O$pulls down to ground and $\overline{V_O}$ is pulled to VDD, hence differential output is LOW. For A=1 and B=1, $\overline{V_O}$ pulls down to ground and V_O is pulled to VDD, hence the differential output becomes HIGH.

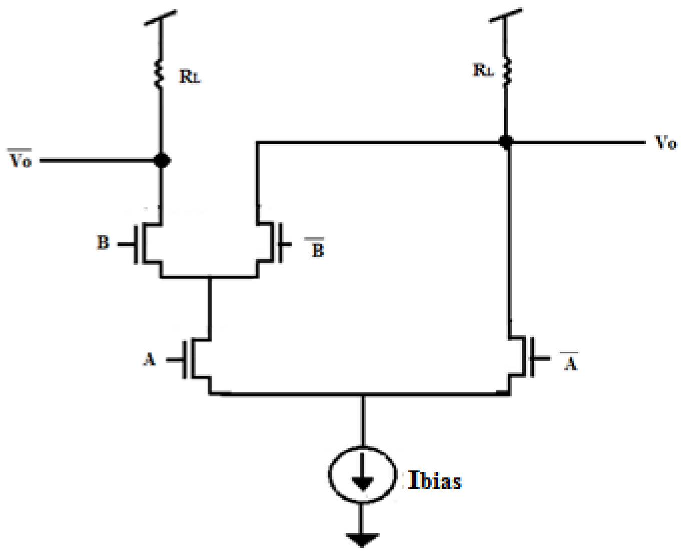
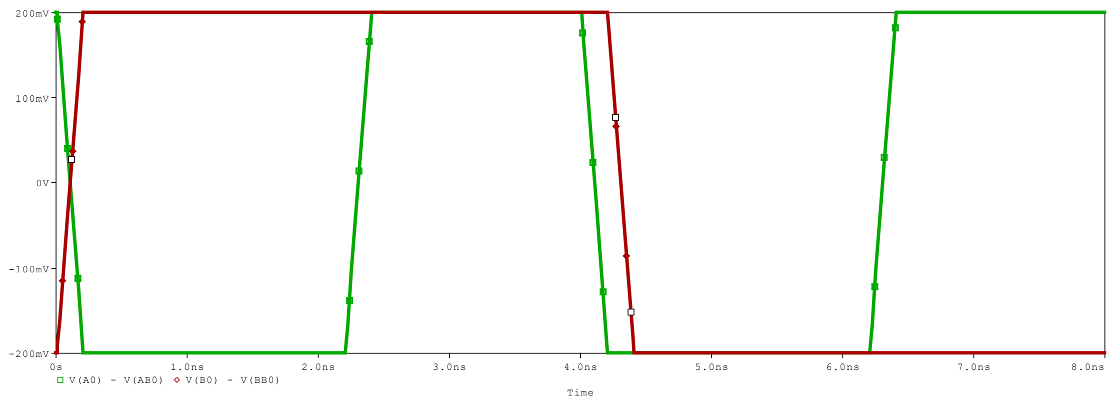
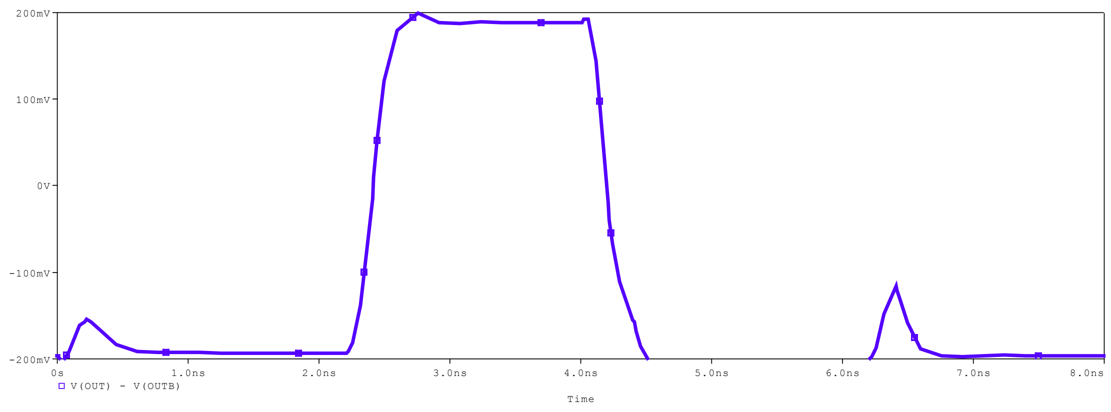


Fig. 2.7 MCML AND gate



(a)



(b)

Fig. 2.8 MCML AND gate results (a) Inputs (A and B) (b) Output (OUT)

Therefore, the circuit functions as an AND gate and is made using MCML style, hence named MCML AND gate. The simulated results of MCML AND gate with resistive load is shown in fig. 2.8. The plot in fig. 2.8(a) shows the inputs (A and B with different colours), which are fed to the gate and the plot in fig. 2.8(b) displays the corresponding output.

2.3.3 MCML OR Gate

The MCML OR gate circuitry with differential inputs (A and B) fed to NMOS pairs is shown in fig. 2.9. The differential outputs are obtained at $V_O and\ \overline{V_O}$. The circuits operates as follows, when B=1 $\overline{V_O}$ is pulled down to ground and its counterpart $V_O$is pulled up resulting in output HIGH, and when B=0, the output of the circuit depends on input A. So when B=0 A=1, the output is HIGH and when B=0 A=0, the output becomes LOW. Thereby the OR gate functionality is confirmed. The simulated waveforms of MCML OR gate with resistive load is shown in fig. 2.10. The plot in fig. 2.10(a) shows the inputs (A and B with different colours), which are fed to the gate and the plot in fig. 2.10(b) displays the corresponding output.

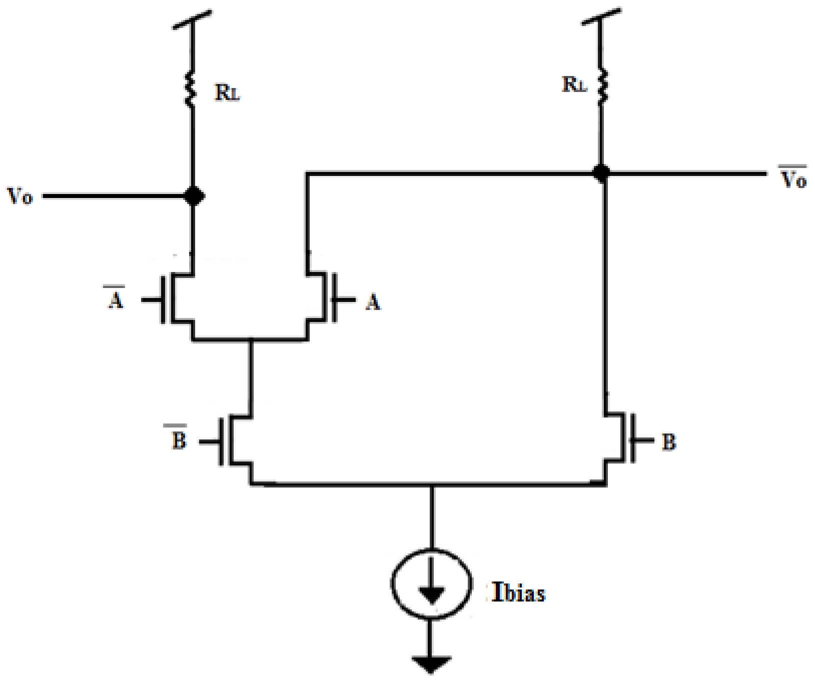
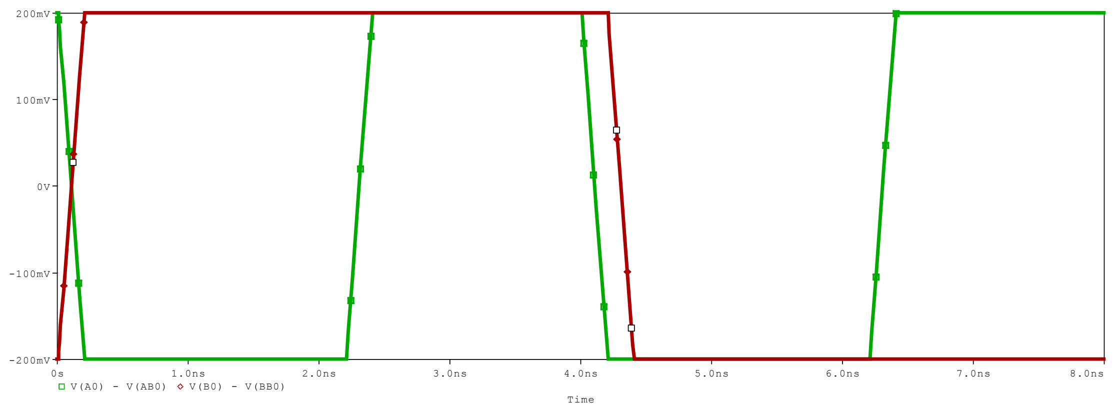
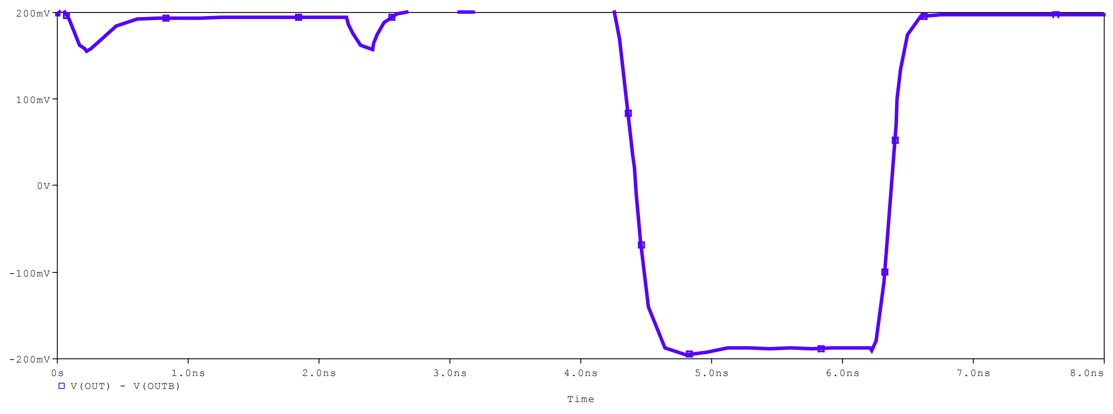


Fig. 2.9 MCML OR gate



(a)



(b)

Fig. 2.10 MCML OR gate results (a) Inputs (A and B) (b) Output (OUT)

2.3.4 MCML XOR Gate

The MCML XOR gate circuit is shown in fig. 2.11. The gate is fed with differential inputs to the NMOS pair of the pull down network and differential outputs is determined by $V_O - \overline{V_O}$. The circuit operation can be understood by current steering principle. All possible combinations of A and B as taken for previous gates can also be considered here to verify MCML XOR gate operation. The circuit operation is verified using PSPICE and the results for the same is presented in fig. 2.12. The plot in fig. 2.12(a) shows the inputs (A and B with different colours), which are fed to the gate and the plot in fig. 2.12(b) displays the corresponding output.

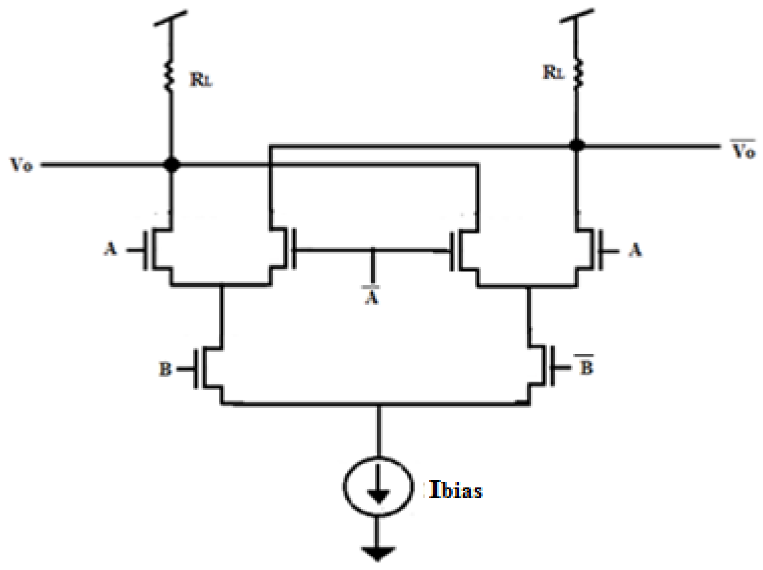
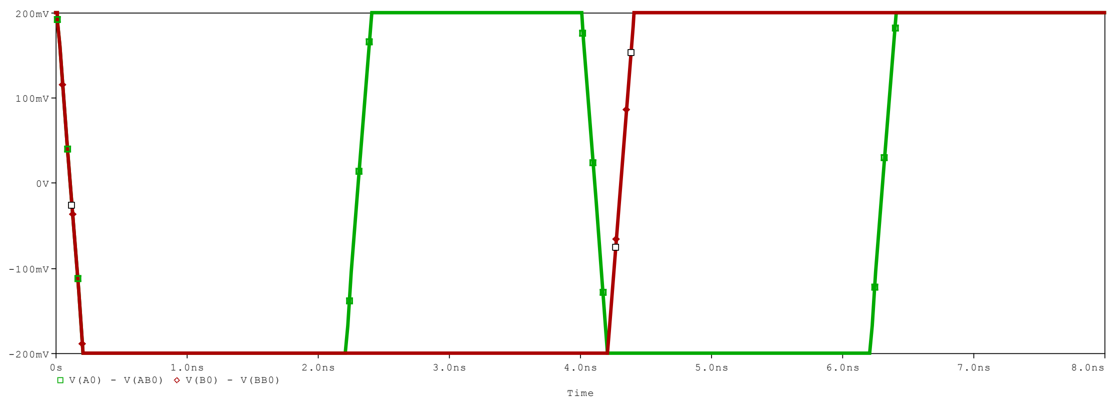
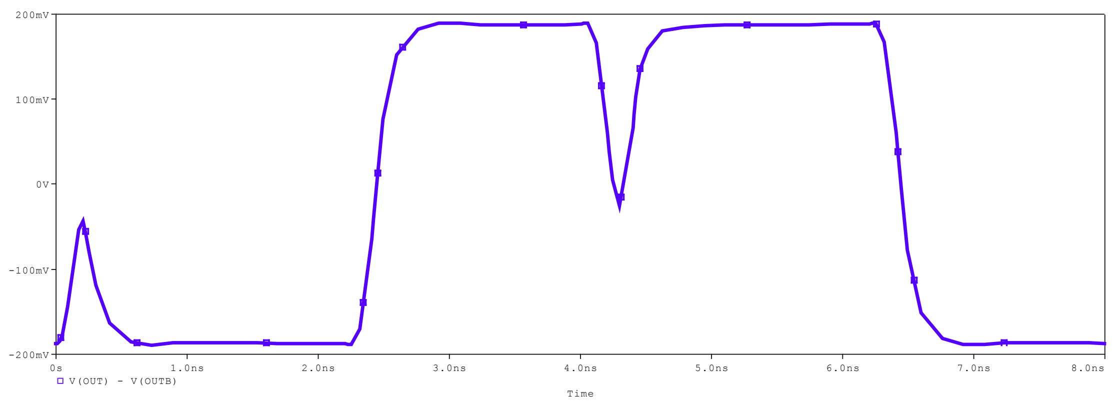


Fig. 2.11 MCML XOR gate



(a)



(b)

Fig. 2.12 XOR gate results (a) Inputs (A and B) (b) Output (OUT)

2.3.5 MCML Multiplexer

The multiplexer circuit is used to select the output depending upon the selection bit. The 2X1 Multiplexer circuit designed using MCML topology is shown in fig. 2.13. The circuit is fed with three differential inputs A, B and S. A and B are the two inputs to the multiplexer whereas S acts as the selection bit which selects one of the two inputs fed to the Multiplexer. So, depending upon the value of S bit, any one of the input is selected and obtained at output $V_O - \overline{V_O}$. The simulated results of this circuit are shown in fig. 2.14. The plot in fig. 2.14(a) shows the inputs (A, B and S with different colours) fed to the gate in which signal S coloured in grey is the selection input. The plot in fig. 2.14(b) displays the corresponding output.

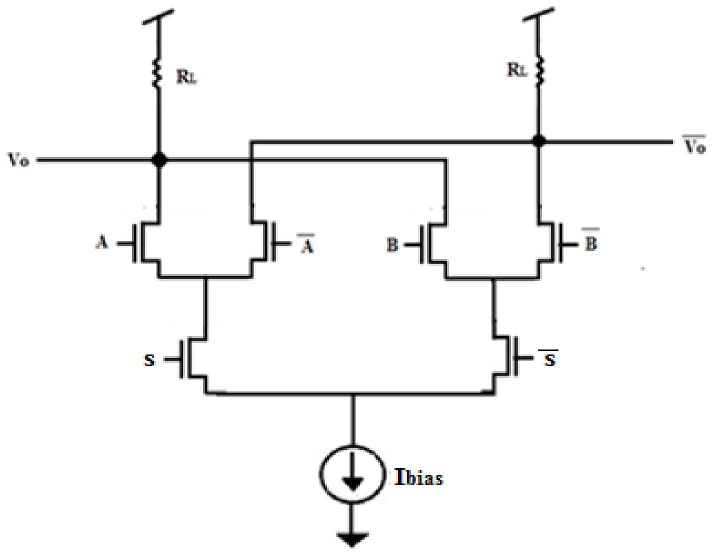
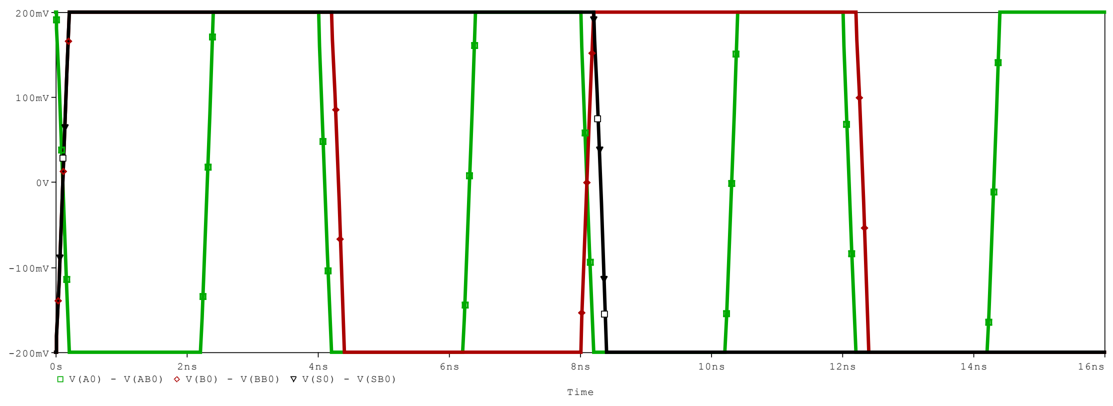
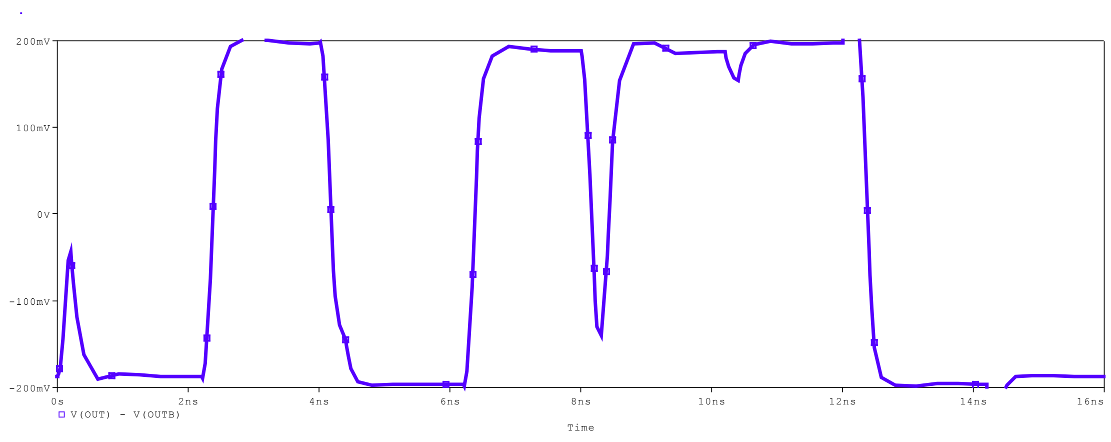


Fig. 2.13 MCML Multiplexer



(a)



(b)

Fig. 2.14 MCML Multiplexer results (a) Input (A, B and S) (b) Output (OUT)

2.3.6 MCML Group Generate Block circuit for parallel prefix adders

The fig. 2.15 is of group generate block circuit. The circuit is used in parallel prefix adders to compute group generate carry.

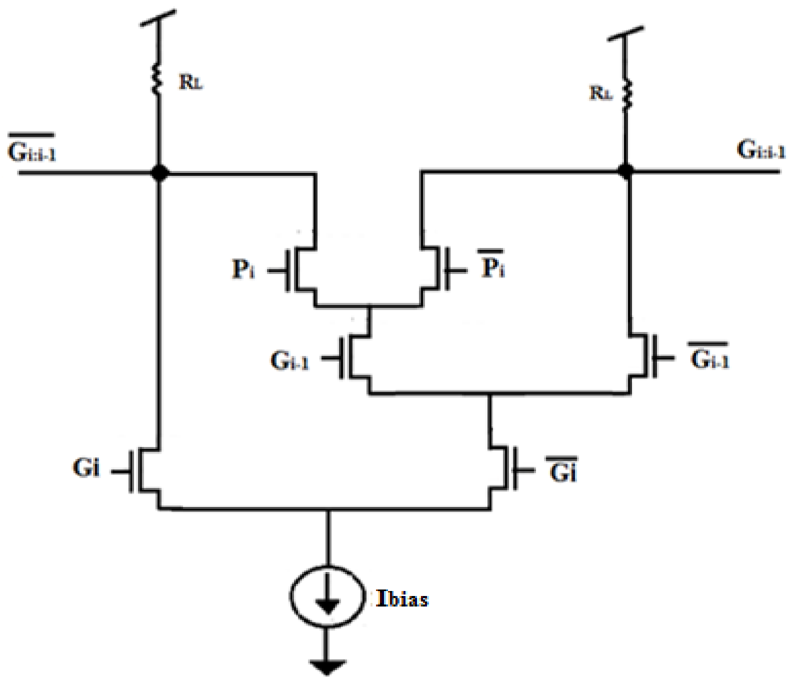
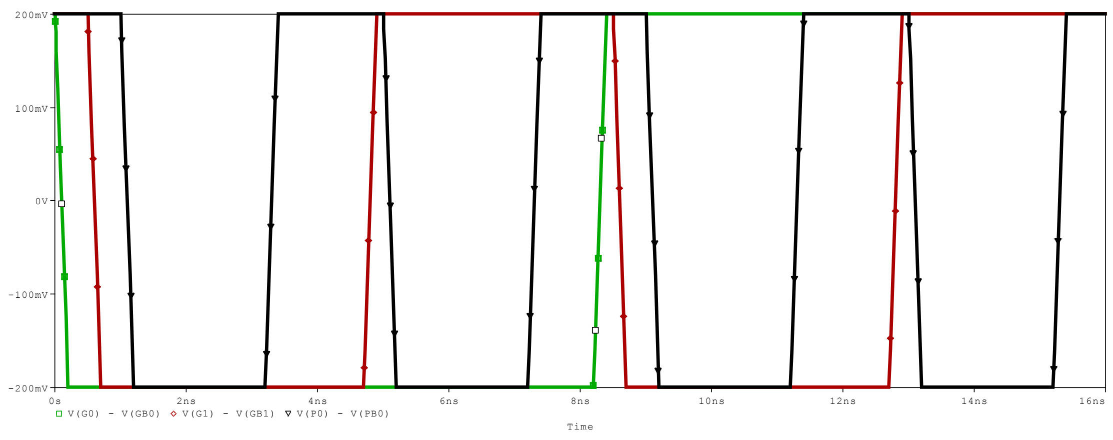


Fig. 2.15 MCML group generate block

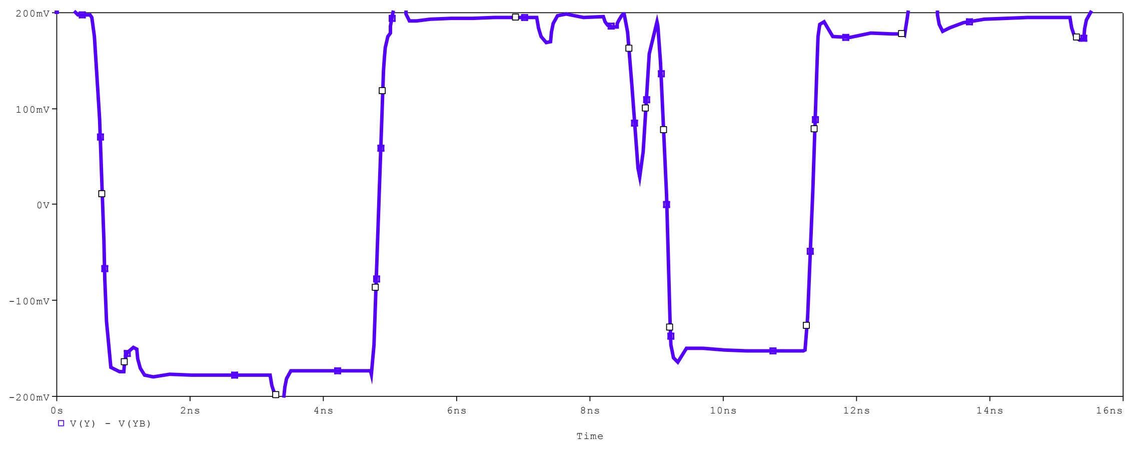
The usage of this block is elaborated in chapter 4. This circuit operates according to the following expression:

$$G_{i:i-1} = G_i + P_i . G_{i-1} \tag{1.21}$$

The simulated results of the above circuit are shown in fig. 2.16. The inputs fed to the circuit are shown in the plot of fig. 2.16(a) with different colours whereas the group generate carry is displayed in fig. 2.16(b).



(a)



(b)

Fig. 2.16 MCML Group generate block results (a) Inputs (G_1, G_0 and P_0) (b) Output (Y)

CHAPTER 3

BINARY ADDERS

An adder is a digital circuit that carry outs addition of numbers. In numerous computers and various other processors, adders find their use not only in the arithmetic logic unit, but also in different other sections of the processor, where they perform calculation of addresses, table indices, and execute similar other operations.

Adders are the basic fundamental components in digital systems. They act as the critical component as the overall performance of the system is affected by them. The construction of adders involves the Boolean equations at the logic level and then those are used for circuit implementation. There are varieties of adders stated in text; simplest and slowest among them is Ripple Carry Adders (RCA). The advancement over RCA is achieved by optimizing the linear carry chain and removing the linear dependence of delay on the number of bit. This advancement is seen in other adders like carry skip adders, carry select adders and carry increment adders. One of the fastest adders is Carry Look-ahead adder (CLA) as it has logarithmic delay dependence. These adders have adapted to parallel-prefix structures which makes the carry computation quicker. Further, speed enhancement schemes on CLA lead to adder structures like Ling Adders.

This chapter provides the background information of binary adders. Different types of adders with their analysis are described in the section to follow. Also, the adders are simulated to verify the functionality. All simulations are performed in PSPICE using TSMC 180nm technology parameters. The basic MCML gates explained in chapter 2 are employed here to design adder structures.

3.1 Binary Adder: Notations and definitions

The binary addition is done bitwise using Boolean equations. A full adder adds binary numbers. The block diagram of one bit full adder is shown in fig. 3.1 It is designed to add three one-bit numbers, written as a, b, and c_{in}; a and b are the input operands, and c_{in} is a bit carried in from the previous less significant stage. The circuit produces a two one-bit output, output carry and sum typically represented by the signals c_{out} and s. The truth table of Full adder is shown in table 3.1. [24] The Boolean equations of the full adder outputs i.e. Sum and output carry can be easily obtained from the truth table and are given in Eq. (3.1) and (3.2).

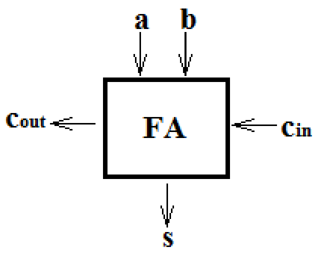


Fig. 3.1 1-bit full adder block diagram

Table: 3.1 Full Adder Truth Table

A	B	Cin	S	Cout	Carry Status
0	0	0	0	0	Delete
0	1	0	1	0	Delete
1	0	0	1	0	Propagate
1	1	0	0	1	Propagate
0	0	1	1	0	Propagate
0	1	1	0	1	Propagate
1	0	1	0	1	Generate
1	1	1	1	1	Generate

$$s = a \oplus b \oplus c_{in} \tag{3.1}$$

$$c_{out} = a.b + b.c_{in} + a.c_{in} \tag{3.2}$$

The above symbols represent the logical operations in Boolean algebra World. For addition of two numbers the operator like “AND”, “OR” and “XOR” are used. Here in this thesis, a dot between two single bit variables denotes AND operation i.e. A.B signifies A “AND” B. A “+” symbol denotes “OR” operation and “⨁” signifies a “XOR” operation. The gate level full adder structure is shown in fig. 3.2.

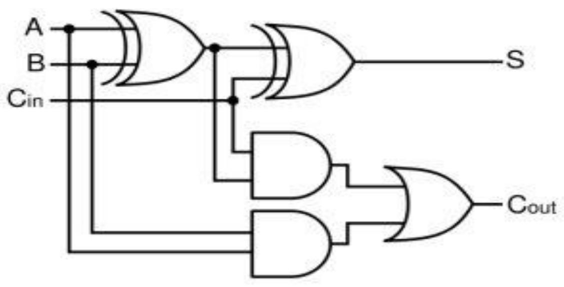
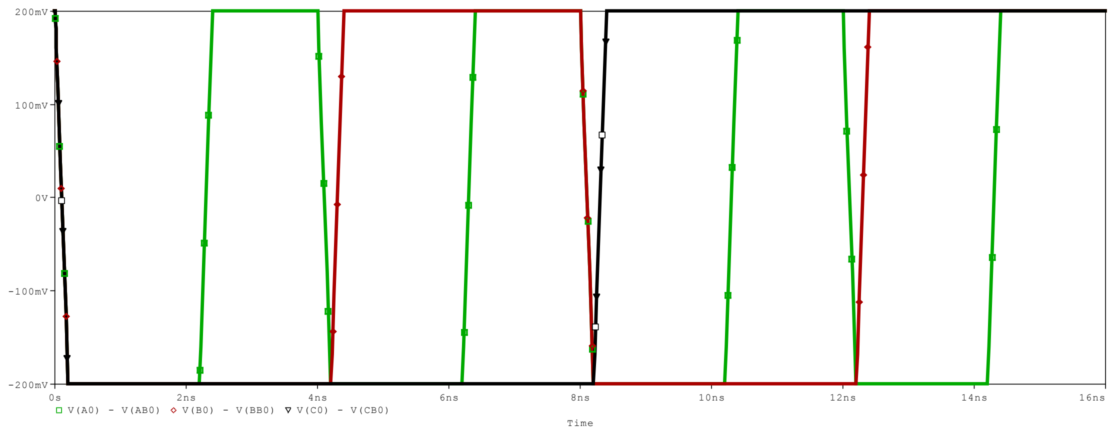


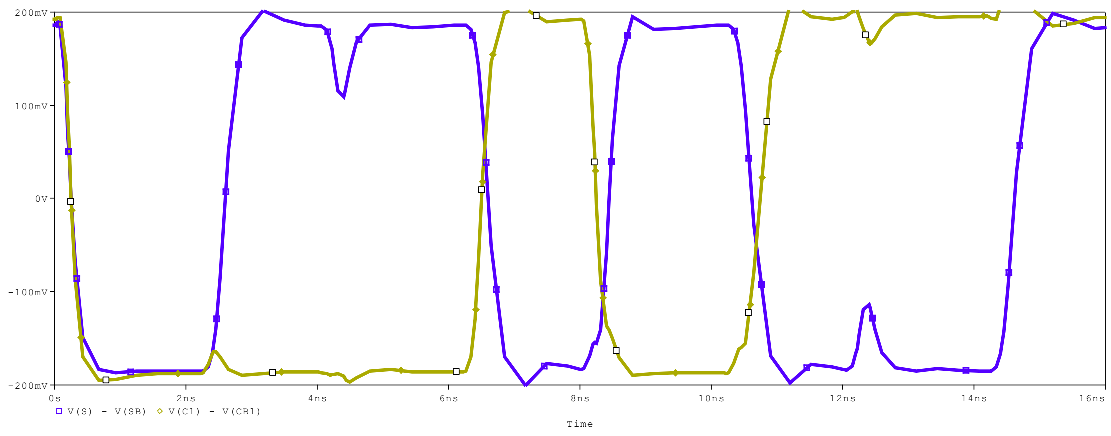
Fig. 3.2 1-bit Full Adder circuit diagram

The simulated results for 1bit full adder shown in fig. 3.2 using basic MCML gates are shown in fig. 3.3. The inputs and outputs of the adder are plotted in separate trace. A, B and C_0 are the inputs of the adder plotted in fig. 3.3(a) with different colours whereas S and C_1 are the outputs plotted in fig. 3.3(b) with unique colours.

To make the carry computation easies it is useful to define some intermediate signals. They are carry generate (g) and carry propagate (p) signals. These signals can be easily obtained from the full adder truth table analysis. As seen from the table, g=1 states that the carry bit will be generated at the output carry (c_{out}) without depending on the input carry (c_{in}) and p=1 ensures that the c_{in}will directly get transferred or propagated to c_{out}.



(a)



(b)

Fig. 3.3 1-bit full adder results (a) Inputs (A, B and C_0) (b) Output (S and C_1)

Additionally, another variable (d) known as half sum is used as well. The Boolean expression of these signals can be easily obtained by inspecting the truth table and are given as:

$$g = a.b \tag{3.3}$$

$$p = a + b \tag{3.4}$$

$$d = a \oplus b \tag{3.5}$$

Above intermediate signals can be employed to obtain the sum and output carry which are given as:

$$c_{out} = g \oplus p.c_{in} \tag{3.6}$$

$$s_i = d \oplus c_{in} \tag{3.7}$$

Let us consider N-bit inputs and outputs of the N-bit adder where A= a_0 ,a_1, a_2 ……a_n , B = b_0, b_1, b_2 ……b_n are N-bit input operands and S = s_0,s_1,s_2 …..s_n are the N-bit output sum.

The above equations Eq. (3.3) to (3.7) are the general terms for one-bit adder and can be used to get N-bit adder results.

Therefore,

$$g_i = a_i \, . b_i \qquad (3.8)$$

$$p_i = a_i + b_i \qquad (3.9)$$

and

$$d_i = a_i \oplus b_i \qquad (3.10)$$

where “i” is an integer whose value range from 0 to n.

Also, the outputs of the N-bit adder for any ith internal bit will be given as:

$$c_{i+1} = g_i + p_i . c_i \qquad (3.11)$$

$$s_i = d_i \oplus c_i \qquad (3.12)$$

where c_i is the input carry to ith full adder and c_{i+1} is the output carry from it.

3.2 Ripple Carry Adder

It is the simplest adder to carry out N-bit addition. The N-bit ripple carry adder is constructed by concatenating N number of one-bit full adder. In this way, the output carry from previous full adder stage is connected as an input carry to the next full adder stage. Since the carry ripples from one full adder stage to the next one, hence named as Ripple Carry Adder (RCA). A 16-bit RCA is shown in fig. 3.4 and it is formed by connecting 16 one bit full adders in a cascade. It can be seen from the figure that the carry ripples from $c_{in}$to c_{out} via all 16 full adder blocks.

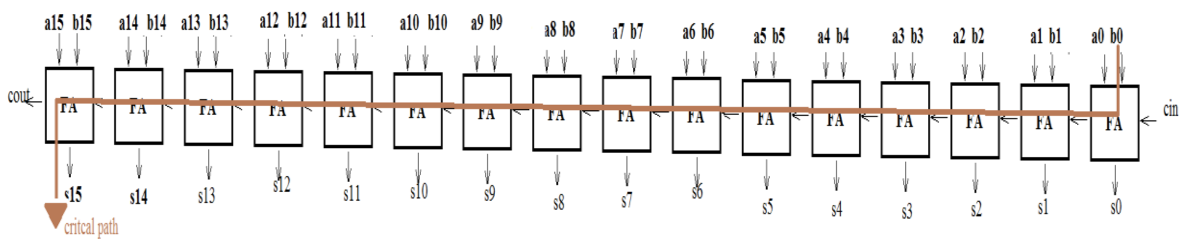
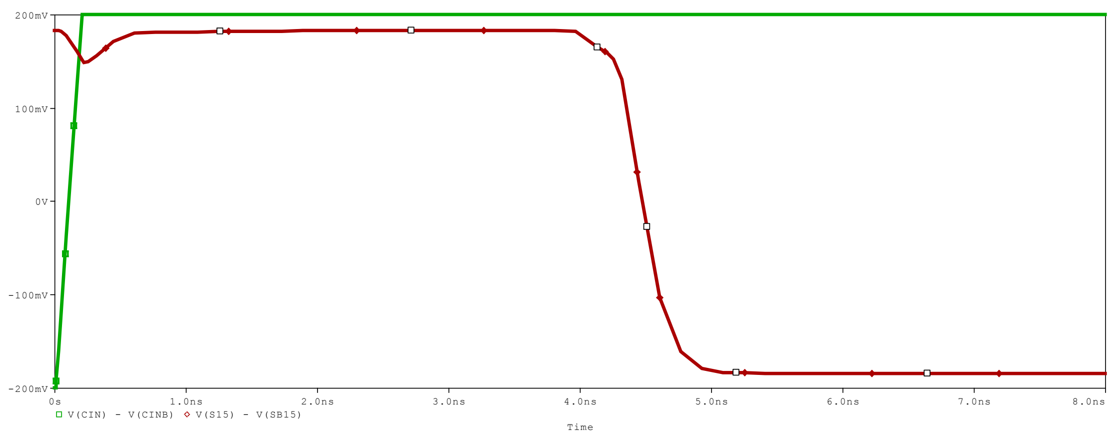


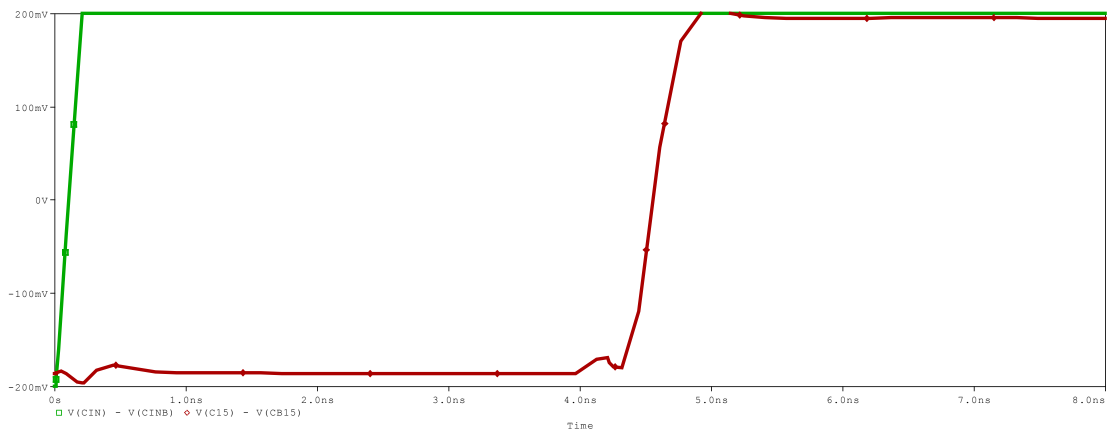
Fig. 3.4 16-bit Ripple Carry Adder

The critical path in RCA is highlighted with the solid line from the inputs of first full adder to the sum of the last full adder block. The serious drawback of this adder is that the delay dependents on the bit length and linearly increases with it. As mentioned above, each full adder block in RCA has to wait to get the input carry from the previous stage to achieve the steady-state output result. Therefore, even if the adder has generated a value at its output node, it has to pause till the propagation of the carry before the output reaches an accurate value [25].

The simulated result of 16-bit MCML RCA for worst case inputs i.e. values of input A and B for each bit taken as 1 and 0 respectively and input carry as 1, is shown in fig. 3.5 where the worst case delay of output carry with respect to incoming carry input is shown in fig. 3.5 (a) and the delay in sum with respect to input carry is plotted in fig. 3.5 (b). The worst case sum delay and output carry delay is measured to be 4.37ns and 4.45ns respectively. Also, the circuit dissipates power of 7.2mW.



(a)



(b)

Fig. 3.5 16-bit MCML RCA results worst case delay in (a) sum and (b) carry

Taking the example of 16-bit RCA in fig. 3.4, the addition of a_{15} and b_{15} cannot reach a steady state value until c_{15} becomes available. In turn, c15 has to wait for c14, and so on down to c_{in}. If one full adder takes T seconds to complete its operation, the final result will reach its steady-state value only after 16.T seconds.

The delay in RCA linearly depends on the number of stages of full adder that must be traversed. The propagation delay in RCA is given by [24]:

$$t_{adder} = (N-1)t_{carry} + t_{sum} \tag{3.13}$$

Where N is the number of stages in RCA.

3.3 Carry Select Adder

In ripple carry adder, input carry to each full adder block generates the output carry. This output carry generation depends on the delay of the previous input carry. So, this shows the linear dependence on the carry delay and the number of full adder block i.e. the number of inputs to the ripple carry adder. This linear dependency can be removed by anticipating both the carry values possible of the incoming carry and computing both results in advance. Once the actual value of the incoming carry is known the correct result is selected by using a simple multiplexer stage. This type of adder is called carry select adder (CSA). The basic block of carry select adder is shown in Fig. 3.6. Consider the basic block which is adding 4bits, instead of waiting for the carry to come, each adder block performs addition in advance by taking input carry as 1 and 0. Both blocks are operating in parallel. Then the final multiplexer stage selects the output with minimal delay.

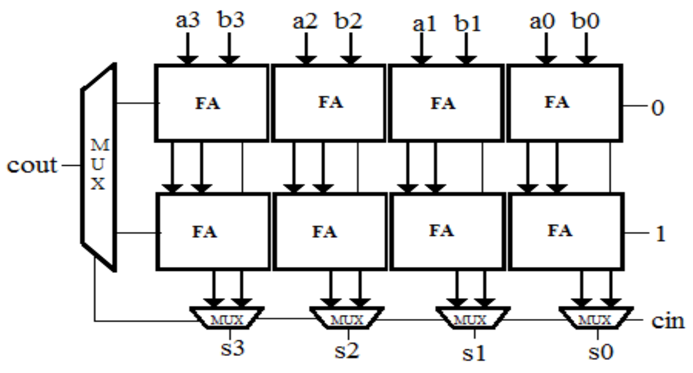


Fig.3.6 Basic Carry Select Adder block (4-bit)

3.3.1 Linear carry select adder

The linear carry select adder is constructed using the fixed size basic carry select adder blocks. An example of 16-bit full adder formed by connecting four basic CSA blocks in parallel is shown in Fig. 3.7.

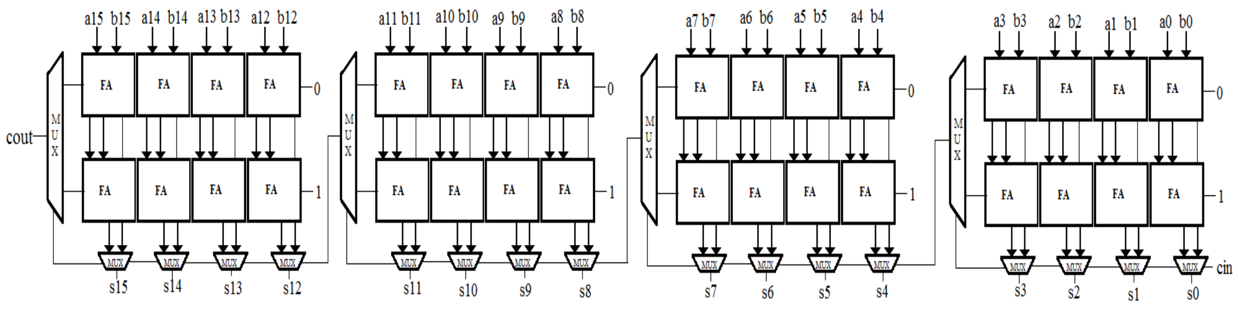

a15 b15 a14 b14 a13 b13 a12 b12
a11 b11 a10 b10 a9 b9 a8 b8
a7 b7 a6 b6 a5 b5 a4 b4
a3 b3 a2 b2 a1 b1 a0 b0
FA
MUX
cout
cin
0
1
s15 s14 s13 s12 s11 s10 s9 s8 s7 s6 s5 s4 s3 s2 s1 s0

Fig. 3.7 16-bit Linear CSA

Propagation delay in linear-CSA is given by [24]:

$$t_{adder} = t_{setup} + Mt_{carry} + \left(\frac{N}{M}\right) t_{mux} + t_{sum} \tag{3.14}$$

Where t_{setup}, t_{sum}, t_{mux} are fixed delays, N is total number of bits and M representing the total number of bits per stage. t_{carry} is the carry delay through one full adder cell. This delay depends on the number of bits in that full adder cell and equals Mt_{carry}.

3.3.2 Square-root Carry Select Adder

A Square-root carry-select adder (SR-CSA) as the name suggests can be understood as an improved form of carry select adder. As we have previously seen in linear carry select adder each block is provided four bits in parallel, SR-CSA being a derivative of CSA is provided bits in parallel but the number of bits given are of variable size [26]. This mismatch or the delay added in the CSA due to the difference of carry and sum from the previous block can be avoided by this technique of variable sized inputs in SR-CSA. This decreases the delay and hence SR-CSA can be used for large number of bits in input.

The 16 bit implementation of a square root adder consists of 5 blocks respectively used for adding 2,2,3,4 and 5 bits. The initial two bits are added using RCA followed by CSA blocks but with variable number of inputs. Two n-bit RCA blocks which perform additions for the respective cases of carry being zero and one, along with a 2:1 multiplexer used to select the correct result once the carry signal is known, together form a n-bit CSA block. A 4-bit CSA is shown in Fig. 3.6.

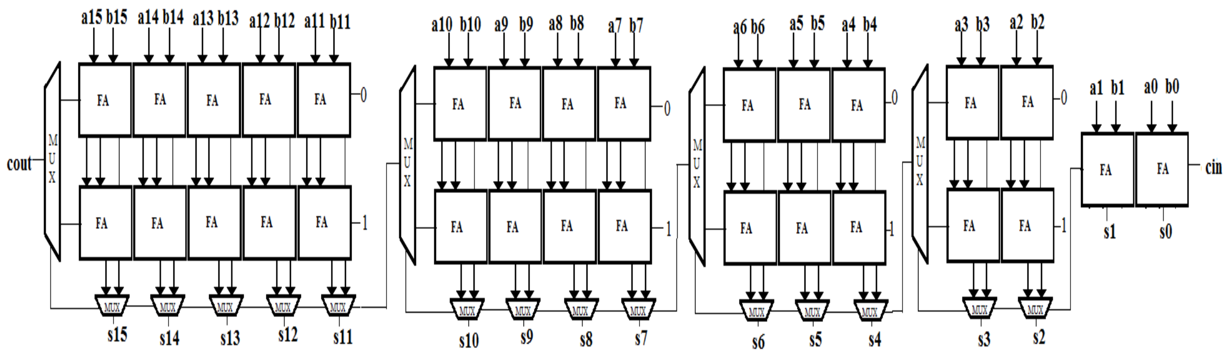


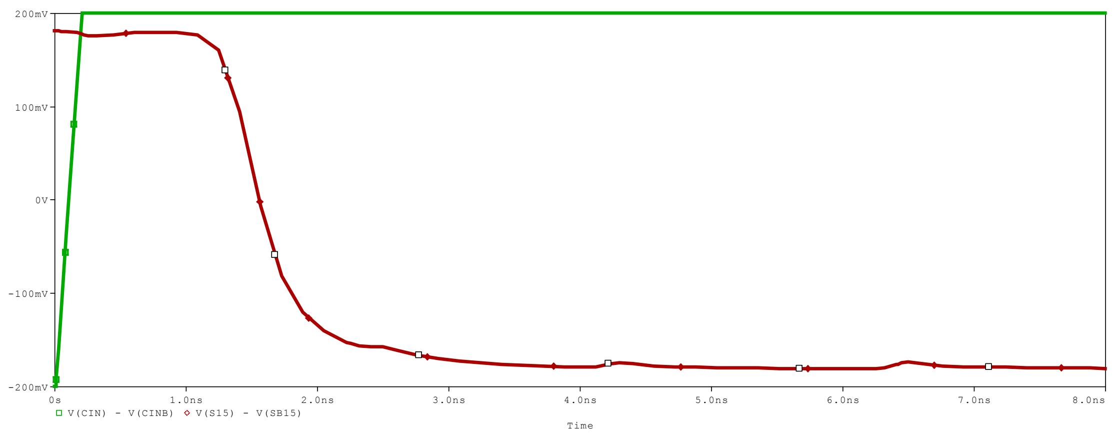
Fig. 3.8 16-bit Square Root Carry Select Adder

The simulated result of 16-bit MCML SR-CSA displaying the worst case delay of sum and output carry along with the input carry is shown in fig. 3.9(a) and fig. 3.9(b) respectively. The measured worst case delay in sum and output carry delay is 1.39ns and 1.45ns respectively. Also, the circuit dissipates power of 15.142mW.

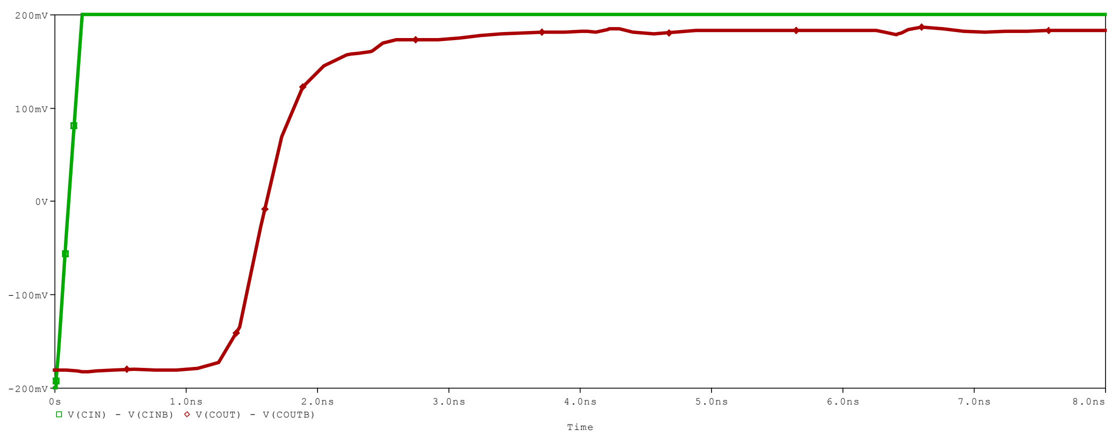
Propagation delay in SR-CSA is given by [24]:

$$t_{adder} = t_{setup} + Mt_{carry} + \left(\sqrt{2N}\right)t_{mux} + t_{sum} \tag{3.15}$$

Where $t_{setup}, t_{sum}, t_{mux}$ are fixed delays, N denotes adder's length and M represent the total number of bits per block. t_{carry} is the carry delay through one full adder cell.



(a)



(b)

Fig. 3.9 16-bit MCML SR-CSA results; worst case delay in (a) sum (b) carry

This delay depends on the number of bits in that full adder cell and is proportional to $\sqrt{N}$ for large adder size (N>>M).

3.4 Carry Look Ahead Adder

The rippling effects of carry are still present in some form or the other in carry skip adders and carry select adders. To design high speed adders it is essential to get off this rippling effect completely. Carry Look ahead concept can be utilised to achieve so. It is based on accelerated carry computation by carry generate / propagate logic equation.

For N-bit adder,

$$c_i = g_i + p_i \,.\, c_{i-1}$$

$$= g_i + p_i \,.\, (g_{i-1} + p_{i-1} \,.\, c_{i-2})$$

$$= g_i + p_i \,.\, g_{i-1} + p_i . p_{i-1} . (g_{i-2} + p_{i-2} \,.\, c_{i-3})$$

$$= g_i + p_i \,.\, g_{i-1} + p_i . p_{i-1} . g_{i-2} + p_i . p_{i-1} . p_{i-2} . (g_{i-3} + p_{i-3} \,.\, c_{i-4}) \qquad (3.16)$$

expanding completely,

$$c_i = g_i + p_i \,.\, (g_{i-1} + p_{i-1} . (g_{i-2} + (\ldots\, p_1 . (g_0 + p_0 \,.\, c_{in}) \qquad (3.17)$$

typically c_{in} is equal to 0.

This expanded equation can be utilized to design N-bit adder in which outputs i.e. Sum and carry out are independent of previous carry. Thus, the rippling effect is removed completely.

In CLA's, all carries are calculated together using a digital logic. This logic is formed using generate/ propagate signals which are employed to generate carry for the next blocks. The single bit generate and propagate signals can be elaborated for group formation. The relation employed for the same is given as:

$$G_{i:k} = G_{i:j} + P_{i:j} \,.\, G_{j-1:k} \qquad (3.18)$$

$$P_{i:k} = P_{i:j} \,.\, P_{j-1:k} \qquad (3.19)$$

Here, the suffixes i:k denote the group terms from ith bit to kth bit.

The final group carry can be computed by:

$$C_{i+1} = G_{i:k} + P_{i:k} \,.\, C_k \qquad (3.20)$$

The delay of this type of adder depends logarithmically on the number of inputs [27]. Therefore, they are considered as high speed adders.

3.5 Ling Adder

Ling adders proposed in [11] accelerate the computation of conventional CLAs. They act as an improved scheme on conventional CLA. The idea is to make use of carry generate and propagate signal which are defined in equations (3.8)-(3.10). They utilises simplified form of CLA equations that depends on adjacent bit pairs (a_i, b_i) and (a_{i+1}, b_{i+1}).

The inherent relation $g_i = g_i . p_i$ is used to extract p_i from group carry generate equation as shown,

$$G_{i:k} = p_i(g_i + g_{i-1} + p_{i-1} . g_{i-2} + \cdots + p_{i-1} \ldots p_{k+1} g_k) \qquad (3.21)$$

Where

$$g_i + g_{i-1} + p_{i-1} . g_{i-2} + \cdots + p_{i-1} \ldots p_{k+1} g_k = H_{i:k} \qquad (3.22)$$

In Ling adders, a new type of carry know as ling carries (H_i) given in the above equation were introduced to compute the carry faster. The Ling carry, H_i can be easily obtained by the logical OR operations of the two consecutive carry signals [27], i.e.

$$H_i = c_i + c_{i-1} \qquad (3.23)$$

The ling carries H_i can be computed faster than the corresponding carries c_i since they depend on reduced Boolean function. This can be proved by considering the case of c_4 and H_4,

$$c_4 = g_4 + p_4.g_3 + p_4.p_3.g_2 + p_4.p_3.p_2.g_1 + p_4.p_3.p_2.p_1.g_0 \qquad (3.24)$$

$$H_4 = g_4 + g_3 + p_3.g_2 + p_3.p_2.g_1 + p_3.p_2.p_1.g_0 \qquad (3.25)$$

The reduced number of logic levels can be observed in case of ling carry when considering two-level logic gates. The calculation of c_4 requires five logic levels while that of H_4requires only four. So this reduced number of logic levels makes the carry computation faster.

Even though the bit H_i is easy to calculate, the computation of final sum bit, s_i using ling carries is complex compared to conventional carries computation.

As we know, $c_i = p_i.H_i$, (3.26)

And it is also known that $s_i = d_i \oplus c_{i-1} = d_i \oplus p_{i-1}.H_{i-1}$, (3.27)

According to [28], the s_i bit can be modified to

$$s_i = \overline{H_{\iota-1}}.d_i + H_{i-1}.(d_i \oplus p_{i-1}) \ , \qquad (3.28)$$

This equation (3.28) can be implemented using a multiplexer that selects d_i or $d_i \oplus p_{i-1}$ depending upon the value of H_{i-1}. Assuming that generally an approximate delay of XOR gate is equal to that of multiplexer and also, both d_i and $d_i \oplus p_{i-1}$ are computed in fewer logic levels than H_{i-1}, then no extra delay is added by using ling carries for sum bit computation. In fact, they are computed at faster. Design of Ling adder employing parallel prefix architecture and also its MCML implementation is discussed in the next chapter.

CHAPTER 4

PARALLEL PREFIX ADDER

Parallel prefix adders fall in the category of high performance adders. They are the variants of CLA technique and provide more efficient implementation of CLAs. They are also known as multi-level look-ahead adders as a multi-level tree structure is constructed using small look-ahead blocks. The delay of such adders is logarithmically proportional to the number of inputs therefore may also termed as logarithmic adders [27]. This makes these adders suitable for VLSI implementation as high speed and reliable computation is need of VLSI chips. The feasibility of such adders for VLSI implementations attracts the interest of VLSI designers in this field. In present technology, they are listed as best adders for fast arithmetic operation. The computation in parallel prefix adders are done in three stages namely pre-processing stage, prefix stage and post-processing stage. In prefix stage, the small groups of intermediate prefixes are calculated and combined with each other to form larger and larger group prefixes till all the prefixes are merged together to get the final output carry. Therefore this stage is made up of tree structures which perform carry computation. There are many tree structures proposed in literature, some of them are Kogge Stone [29], Sklansky [30], Ladner Fischer [28], Brent Kung [31], Han Carlson [32], and Knowles [33].

This chapter provides sufficient description of parallel prefix adder scheme and then its utilization for CLA and Ling adders are discussed. Various adder analysis using different tree structures are performed along with the simulation results. All simulations are carried out in PSPICE using TSMC 180nm technology parameters. The basic gates used inside adder are designed using MCML topology and are presented in chapter 2.

4.1Basic of parallel prefix computation

As we know, parallel prefix computation comprises of three stages mentioned earlier.

In first stage or pre computation stage, carry generate bit and carry propagate bit along with half sum bit are computed. Fig. 4.1 shows the pre-computation block or Propagate and Generate (P&G) block along with its internal gate structure. The block consist of AND, OR and XOR gate. Single bit inputs (a,b) are fed to the block and propagate, generate and half sum bits are obtained as outputs. The output bits are calculated according to equations (3.3), (3.4) and (3.5).

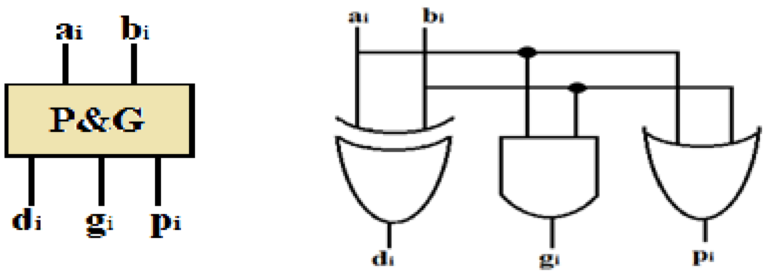


Fig. 4.1 Bit Propagate and generate in CLA

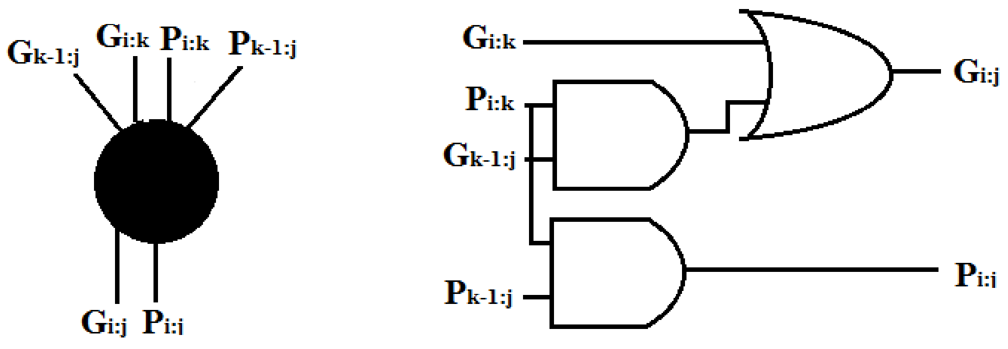
The computation of carry group generate and carry group propagate is carried out in second stage called prefix stage. These bits are calculated using equation (3.18) and (3.19).An associative operator "**o**" is used to associate two adjacent pairs of generate and propagate bits and is given in [27] as:

$$(g,p)\boldsymbol{o}(g',p') = (g + p.g', p.p') \tag{4.1}$$

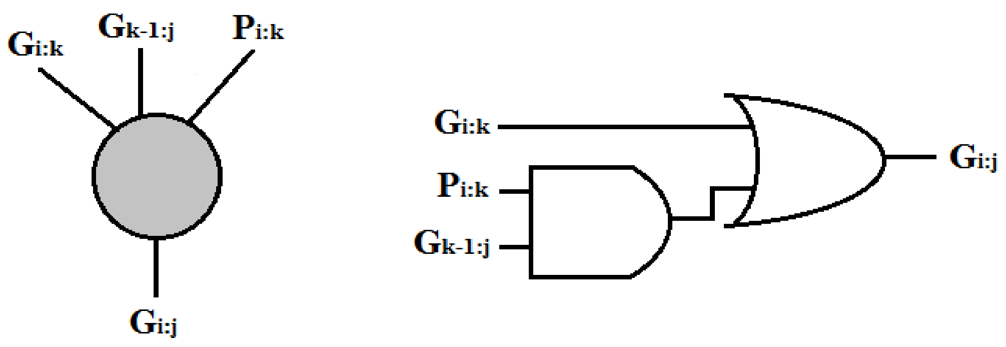
This expression is used to perform prefix stage analysis. Similarly, we can obtain the group term by the association of two overlapping terms:

$$\left(G_{i:j}, P_{i:j}\right) = \left(G_{i:k}, P_{i:k}\right)o\left(G_{m:j}, P_{m:j}\right) \tag{4.2}$$

These above written expressions are used to form a tree structure to perform carry computation. The "**o**" operator is implemented using Black cells and Grey cells shown in Fig. 4.2(a) and Fig. 4.2(b) respectively. Outputs from P&G block are fed to the cells and these cells perform operation as per equation (4.1). The cell combines two inputs and outputs block generate and block propagate bits. There are two types of cells used; black cell and grey cell. Grey cells are used when carry signals are not to be propagated. They generates the carry signal i.e. outputs the final carry signal. Whereas black cell generates and propagates the carry signal. Grey cells are used when no further carry propagation is required. Internally these cells consists of AND and OR gate organization. Both these cells are used to form the tree structure. So, a tree network is formed in this stage. This network can be realized by many structures proposed in literature [28-33] and are discussed in next section.



(a)



(b)

Fig. 4.2 Cell definitions (a) Black Cell (b) Grey Cell

The final stage or post computation stage is responsible for final output carry generation and sum production. This is accomplished by using the equations (3.10), (3.12) and (3.20). The sum generation block is shown in Fig. 4.3. It consists of a XOR gate which computes the full adder sum. Half sum bit produced by P&G block along with previous generate bit is fed the block to obtain the final sum.

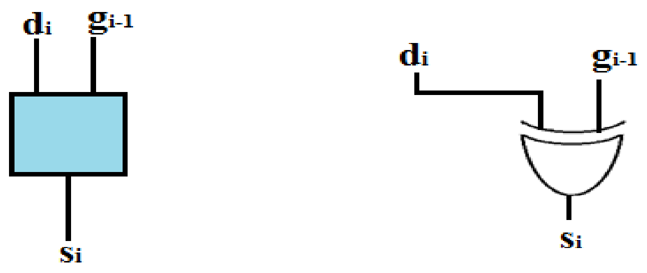


Fig. 4.3 Sum generation block

4.2 Tree structures

We have seen in section 4.1 that the consecutive union of generate and propagate pairs (g, p) are used to obtain group generate and propagate.

$$(G_{i:j}, P_{i:j}) = (G_{i:k}, P_{i:k}) o (G_{m:j}, P_{m:j}) \tag{4.3}$$

Also group bits can be expanded using "**o**" operator. For group term ($G_{k:j}, P_{k:j}$):

$$(G_{k:j}, P_{k:j}) = (g_k, p_k) o (g_{k-1}, p_{k-1}) o \ldots o (g_{j+1}, p_{j+1}) o (g_j, p_j) \tag{4.4}$$

There are certain rules that are to be followed while constructing tree structure. Only adjacent terms can be combined using "**o**" operator therefore, the sequence of the terms matters in the above equation. The groups in the prefix tree structure are always combined in the forward directing i.e. from LSB to MSB not the other way round.

Prefix tree employ the combination of adjacent bits using "**o**" operator which is implemented in the tree structure using black or grey cell shown in Fig. 4.2. There are various tree topologies; we are focusing on six of them that are explained in this section.

4.2.1 Kogge Stone prefix tree

Kogge Stone prefix tree uses minimum number of levels for implementing the tree structure. The maximum fan out at each level in this structure is bound to 2. It is a high performance structure and at the expense of large number of cells used in its construction. It requires $nlog_2n - n + 1$ number of cells where n is adder's length [29]. The delay of this adder is log_2n.

The 16-bit Kogge Stone prefix tree is shown in Fig. 4.4.The tree is fed with p,g signals of each bit and final carries are generated at the output. This tree structure computes carry in 4 levels. In level 1, all the adjacent p,g bits in groups of two are combined using black and grey cells. As mentioned in the previous section, grey cell is used when no further carry is propagated, its output provide the final carry and therefore it is used for two LSB's in level 1 to obtain c_1 . In Level 2, twelve black cells are used to propagate carry to next level and two gray cells are used to generate carries c_2 and c_3. Next four carries (c_4-c_7) are generated in level 3. Last level comprises of eight grey cells only which generates remaining eight carries (c_8-c_{15}).

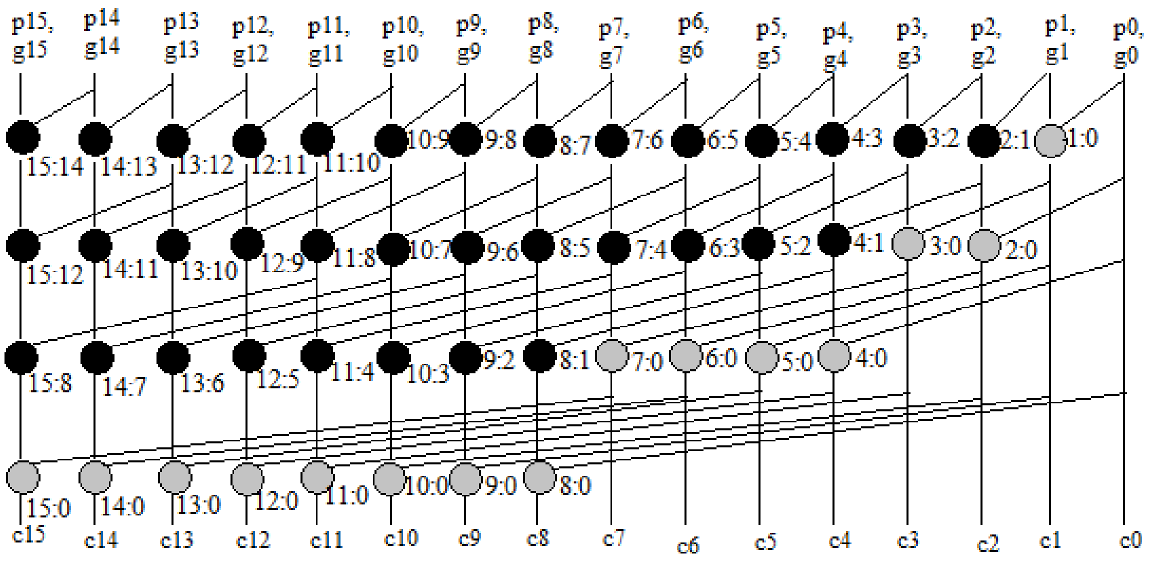


Fig. 4.4 Kogge Stone prefix tree

4.2.2 Sklansky prefix tree

Sklansky prefix tree has the simplest structure among the prefix trees having minimum number of levels. The fan out of Sklansky tree increases exponentially along the critical path from LSB to MSB. Fan out of this adder is given by $\frac{n}{2}$ and is responsible for excessive delay in the circuit with the increase in length of adder. The delay is given by log_2n and the number of cells required to built this structure is $\frac{n}{2}log_2n$[30]. The 16-bit sklansky prefix tree is shown in Fig. 4.5. The tree is fed with p,g signals of each bit and final carries are generated at the output.

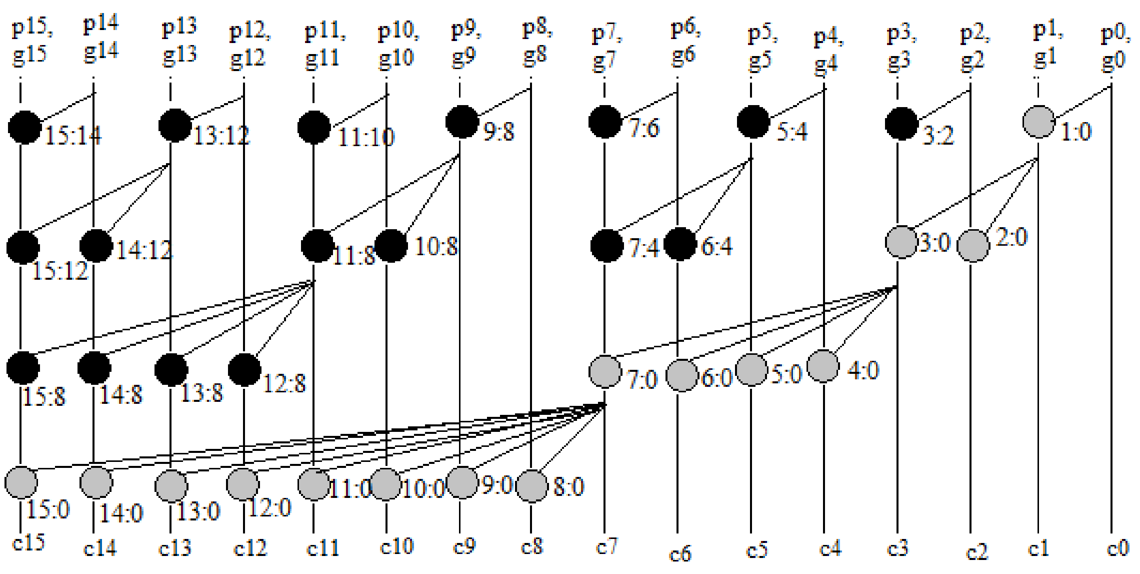


Fig. 4.5 Sklansky prefix tree

This tree structure also computes carry in 4 levels. In level 1, alternate adjacent p,g bits are combined in groups of two using one grey and seven black cells. So, here number of cells has reduced to half as compared to Kogge Stone tree. The carry bit c_1 is produced at this level. In Level 2, six black cells are used to propagate carry to next level and two gray cells are used to generate carries c_2 and c_3. Next four carries (c_4-c_7) are generated in level 3. Last level comprises of eight grey cells which generates remaining eight carries (c_8-c_{15}).

4.2.3 Ladner Fischer prefix tree

Ladner Fischer prefix tree is a compromise between Brent Kung and Sklansky trees. It relieves the high fan-out issue of Sklansky prefix tree structure. The delay of this structure is $log_2 n + 1$[28]. It has large fan out but has minimum logic depth. The 16-bit Ladner Fischer prefix tree is shown in Fig. 4.6. The tree is fed with p,g signals of each bit and final carries are generated at the output. The carry computation in this tree structure takes place in 5 levels. Level 1 of this tree is same as that of Sklansky where alternate adjacent p,g bits are combined in groups of two using one grey and seven black cells. The carry bit c_1 is produced at this level. In Level 2, three black cells, instead of six in previous tree, are used to propagate carry to next level and one grey cell produce c_3. Level 3 comprises of two black cells and two grey cells thereby producing two carries c_5 and c_7. Next four odd carries (c_9, c_{11}, c_{13}, c_{15}) are generated in level 4 by four grey cells.

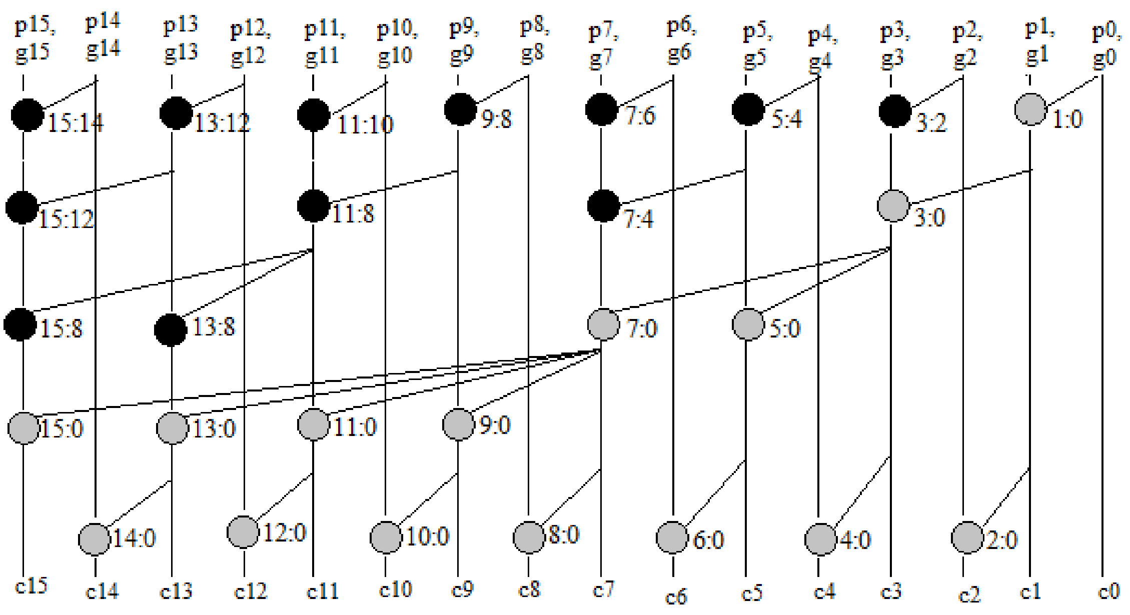


Fig. 4.6 Ladner Fischer prefix tree

Till this level all odd carries are being computed. So, remaining seven even carries (c_2, c_4, c_6, c_8, c_{10}, c_{12}, c_{14}, c_{15}) are generated by seven grey cells in Level 5 which is the last level of this tree.

4.2.4 Brent Kung prefix tree

Brent Kung structure consists of large number of levels due to which its operating speed reduces. But this adder is power efficient as it uses minimum number of cells. The delay of Brent Kung adder is given by $(2^{*}log_2 n) - 2$ [31]. It has high logic depth which characterise its high fan out. The 16-bit Brent Kung prefix tree is shown in Fig. 4.6. The tree is fed with p,g signals of each bit and final carries are obtained at the output. The carry computation in this tree structure is done in 7 levels. Level 1 is same as that of Sklansky tree where adjacent p,g bits are combined in groups of two alternately. The carry bit c_1 is produced at this level by the single grey cell. Level 2 resembles Ladner Fischer tree wherein three black cells and one grey cell are used. The carry bit c_3 generated at this level. Level 3 comprises of only two cells; one black and one grey, which generates carry c_7. Next four levels comprises of only grey cells. Level 4 and 5 generates c_{15} and c_{11} respectively. Remaining three odd bits (c_5, c_9, c_{13}) are computed in level 6. It can be observed that all odd carries are being computed till now. So, remaining seven even carries (c_2, c_4, c_6, c_8, c_{10}, c_{12}, c_{14}, c_{15}) are computed in last level named Level 7.

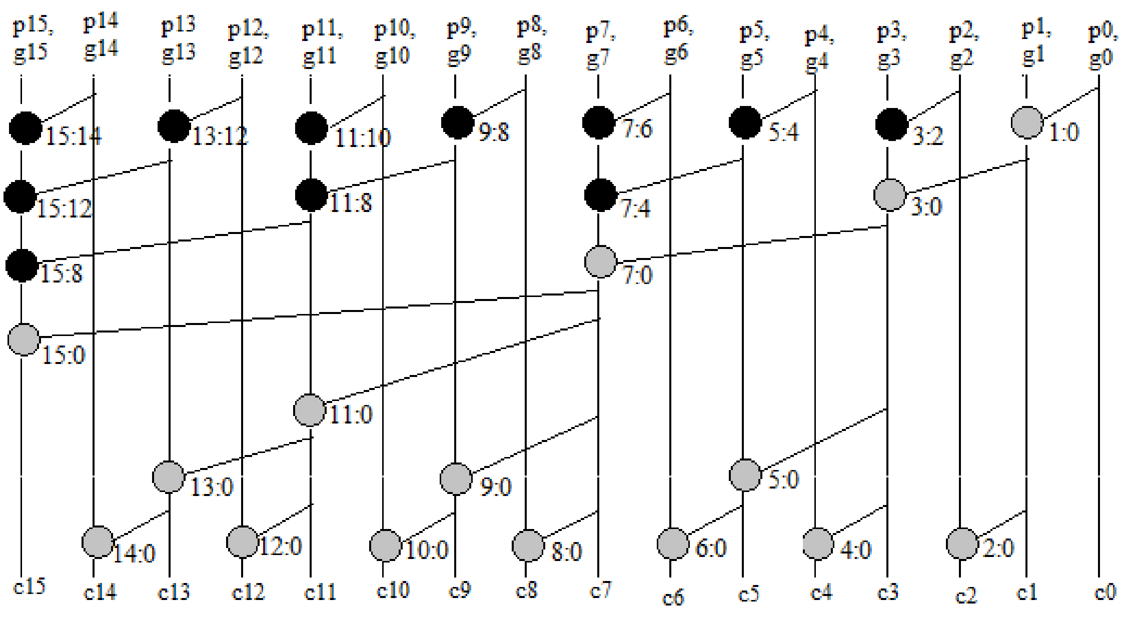


Fig. 4.7 Brent Kung prefix tree

4.2.5 Han Carlson prefix tree

Han Carlson prefix tree is a hybrid tree structure which is made up of both Kogge Stone and Brent Kung structure. The maximum fan out is same as that of Kogge Stone i.e. 2. This topology uses less number of cells but at cost of increase in logic level as in Brent Kung tree. It has $log_2 n + 1$ logic levels [32].

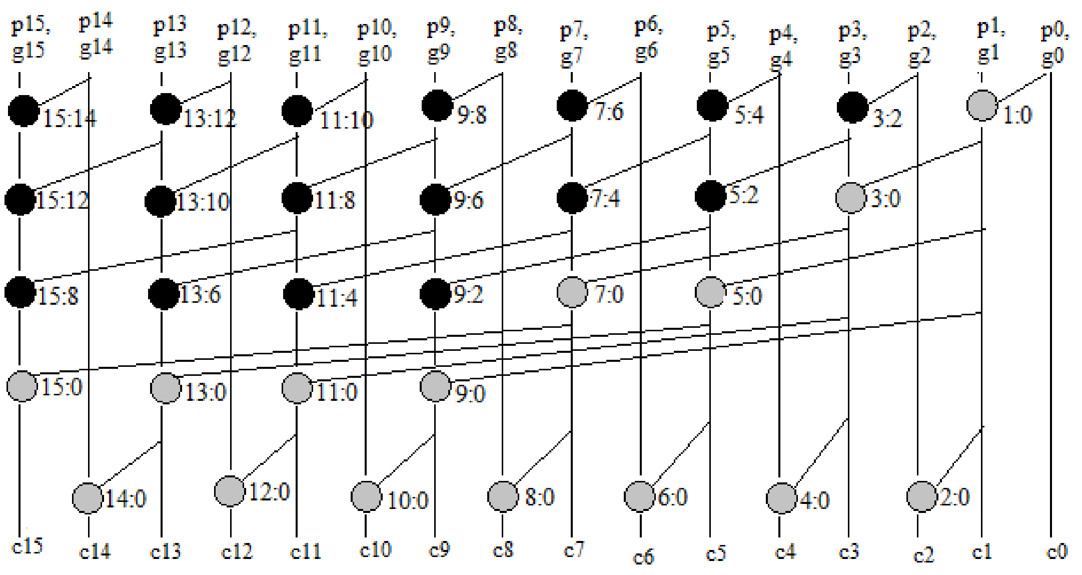


Fig. 4.8 Han Carlson prefix tree

The 16-bit Han Carlson prefix tree is shown in Fig. 4.8. The tree is fed with p,g signals of each bit and final carries are obtained at the output.

This tree structure computes carry in 5 levels. Level 1 of this tree combines alternate adjacent p,g bits in groups of two using one grey (which generates c_1) and seven black cells. Level 2 consists of one grey cell produce c_3 and seven black cells. Level 3 comprises of four black cells and two grey cells thereby producing two carries c_5 and c_7. Next four odd carries (c_9, $c_{11,}$ c_{13}, c_{15}) are generated in level 4 by four grey cells. Till this level all odd carries are being computed. So, remaining seven even carries (c_2, $c_{4,}$ c_6, $c_{8,}$ c_{10}, $c_{12,}$ c_{14}, c_{15}) are generated by seven grey cells in Level 5 which is the last level of this tree. The carry generation in this tree here seems similar to Ladner Fischer tree but here maximum fan-out is 2 which is not the case of Ladner Fischer structure.

4.2.6 Knowles prefix tree

A family of prefix trees with flexible structure was proposed by Knowles [33]. Kogge Stone is also included in Knowles family. The different Knowles trees are symbolized by using notation Knowles [x,x,x,x] where x is replaced with the fan out at each level, the levels from top to bottom are written from right to left in this notation. So, Kogge Stone is Knowles [1,1,1,1]. Other topologies can be Knowles[4,4,2,1], Knowles[4,2,1,1], Knowles[8,2,2,1] etc . The 16-bit Knowles [2,1,1,1] prefix tree is shown in Fig. 4.9. The tree is fed with p,g signals of each bit and final carries are obtained at the output.

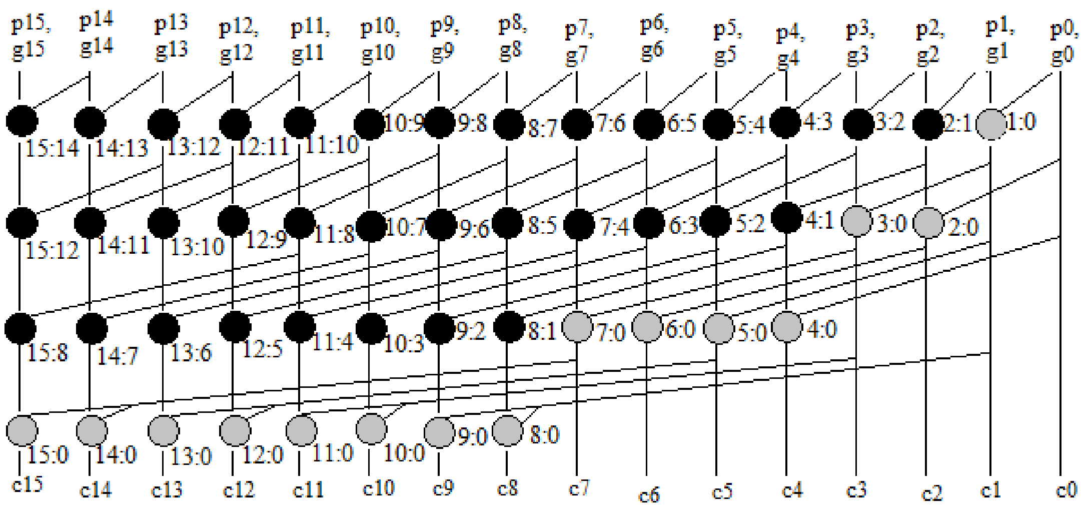


Fig. 4.9 Knowles [2,1,1,1] prefix tree

Kogge Stone is Knowles [1,1,1,1] and here Knowles [2,1,1,1] is shown, thus three levels are same. This tree structure also computes carry in 4 levels out of which first three levels are exactly same as Knowles structure. First eight carries (c_0-c_7) are generated till level 3. Last level comprises of eight grey cells only which generates remaining eight carries (c_8-c_{15}) but here fan out is 3 instead to 2, this is the difference in Kogge Stone tree and Knowles [2,1,1,1] tree .

4.3 Parallel prefix implementation in CLA

CLA's can be implemented with parallel prefix technique to get fast adder design. This can be accomplished by analysing the equations (4.1) and (4.2) which are used as base to form prefix architecture. As we have seen, small groups can be merged together using the associative operator "**o**"to form a larger group. Successive merging of these groups recursively will lead to final output generation. This scheme is utilize to form parallel prefix structure.

The basic blocks used to construct parallel prefix adders are explained in section 4.2. They include propagate and generate (P&G) block, sum generation block and the cells (grey and black) that forms tree structure. The parallel prefix adders are constructed by using the tree structure for intermediate stage. Six tree structures are described in previous section and they are used here for implementing parallel prefix adders.

4.3.1 Kogge Stone parallel prefix adder

It employs Kogge Stone tree/prefix structure in the intermediate stage to compute output carries. The 16-bit parallel prefix adder structure employing Kogge stone tree in it is shown in fig 4.10. The 16 bit inputs, $a_0, a_1, \dots \dots \dots .. a_{15}$ and $b_0, b, \dots \dots \dots .. b_{15}$ are fed to 16 P&G blocks individually which gives 16 propagate and generate signals. These signals are used by Kogge Stone tree to propagate and compute the output carries of each bit. Then last block which is sum generation block calculates the final sum using the carries generated by the tree structure.

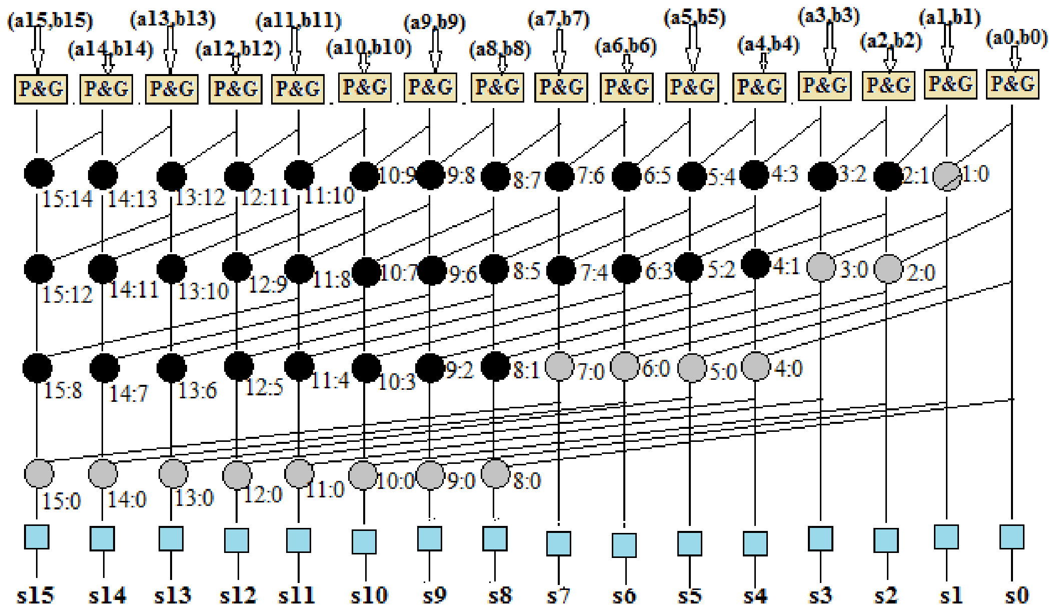
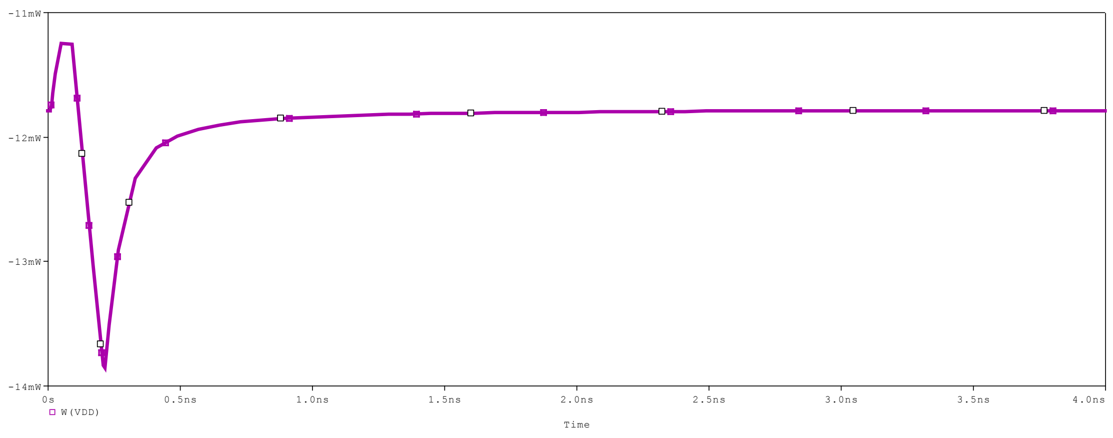
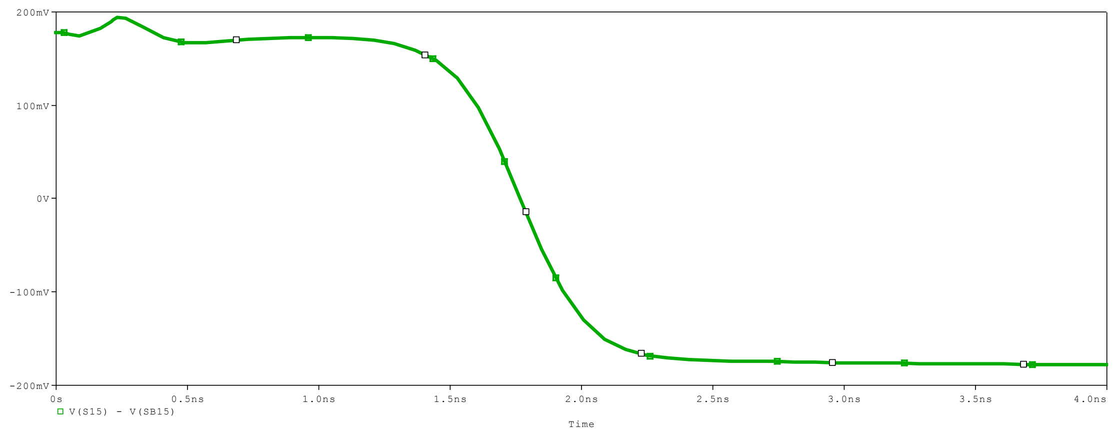


Fig. 4.10 16-bit Kogge Stone parallel prefix adder

The MCML realization for this adder was adopted to implement this adder. Also, the functionality was verified and satisfactory results were obtained. The adder results for worst case inputs i.e. input of LSB taken as 1 (A_0=1 and B_0=1) rest all inputs taken as 1 and 0 (A=1 and B=0) is shown in fig. 4.11. The power dissipation plot is shown in fig. 4.11(a) and the worst case delay in sum bit (S_{15}) is shown in fig. 4.11(b). The measured delay associate with the generation of output sum from simulation is 1.9257ns. Also, the circuit dissipates power equal to 13.855mW.



(a)



(b)

Fig. 4.11 Results of 16-bit MCML Kogge Stone parallel prefix adder (a) Power dissipation (b) Worst case delay in Sum

4.3.2 Sklansky parallel prefix adder

Being parallel prefix adder it has three stages. The second stage called prefix stage is formed using Sklansky tree structure. Hence, named Sklansky parallel prefix adder. The 16-bit parallel prefix adder structure employing Sklansky tree in it is shown in fig 4.12.

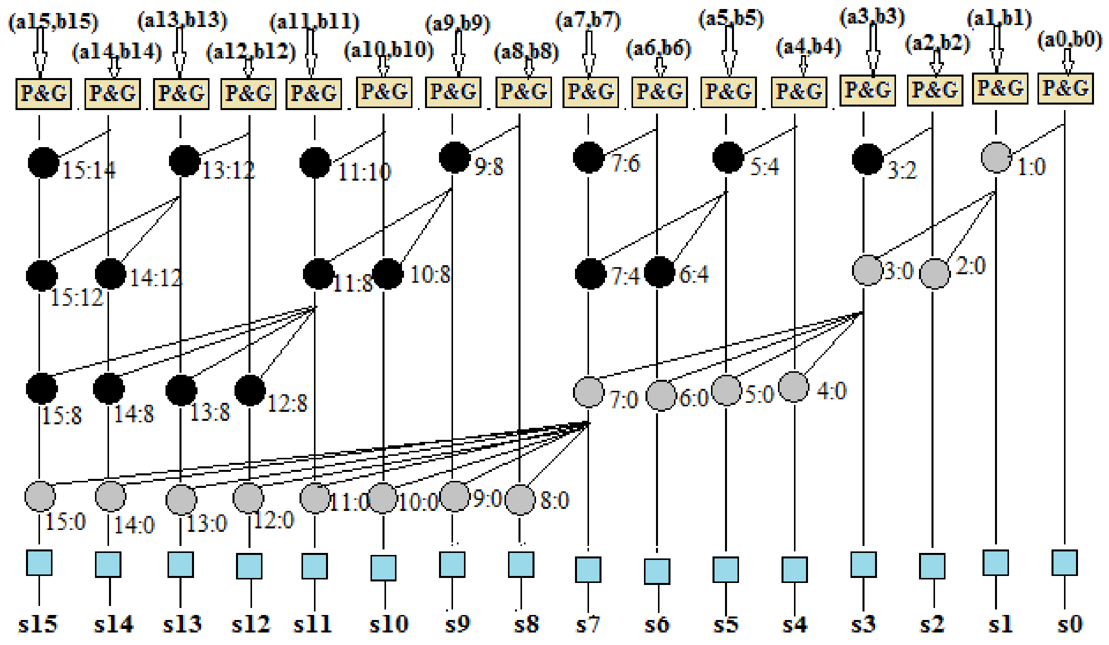
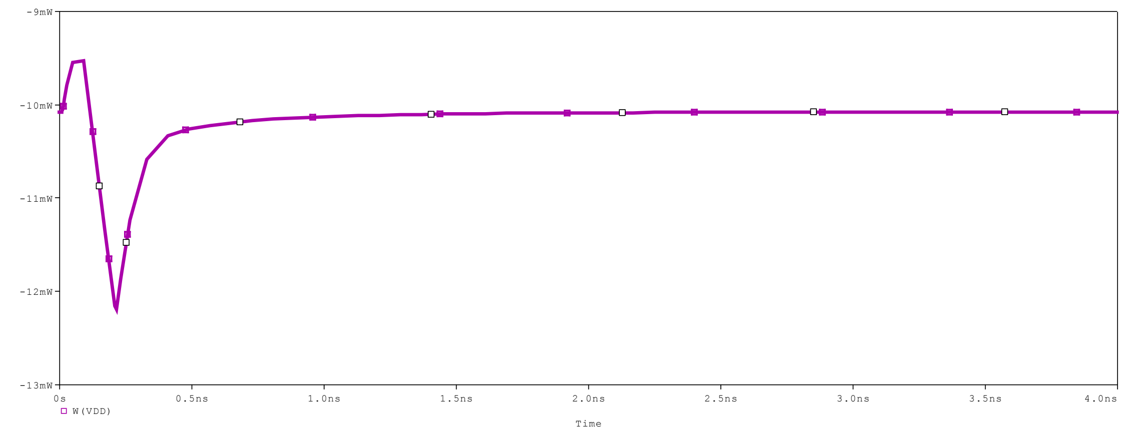
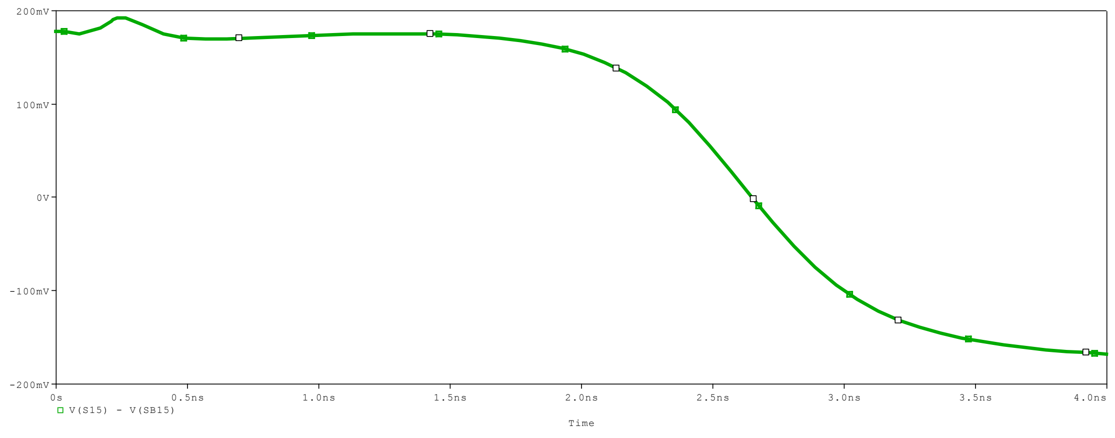


Fig. 4.12 16-bit Sklansky parallel prefix adder

The 16 bit inputs, $a_0, a_1, \dots \dots \dots . . a_{15}$ and $b_0, b, \dots \dots \dots . . b_{15}$ are processed in first stage by 16 P&G blocks individually to get 16 propagate and generate signals. These signals are fed to Sklansky tree to propagate the carry and generate the final ones. Then at third stage sum generation block is used to compute sum using the carries generated by the tree structure.



(a)



(b)

Fig. 4.13 Results of 16-bit MCML Sklansky parallel prefix adder (a) Power dissipation (b) Worst case delay in Sum

The realization of this adder with MCML gates is done to confirm the functionality and optimum results were obtained. The MCML realization of this adder was adopted to implement this adder. The adder results for worst case inputs i.e. input of LSB taken as 1 (A_0=1 and B_0=1), rest all inputs taken as 1 and 0 (A=1 and B=0) is shown in fig. 4.11. The power dissipation plot is shown in fig. 4.13(a) and the worst case delay in sum bit

(S_{15}) is shown in fig. 4.13(b). The measured delay associate with the generation of output sum from simulation is 2.5431ns. Also, the circuit dissipates power equal to 12.186mW.

4.3.3 Ladner Fischer parallel prefix adder

In this adder, the middle stage which computes output carries is made of Ladner Fischer tree. Further using P&G blocks for individual bits in first stage and sum generation block at last makes this structure named as Ladner Fischer parallel prefix adder. The 16-bit parallel prefix adder structure making use of Ladner Fischer tree is shown in fig 4.14. The 16 bit inputs ($a_0, a_1, \dots \dots \dots . . a_{15}$ and $b_0, b, \dots \dots \dots . . b_{15}$) are fed to the adder. Then the carry computation is carried out by Ladner Fischer tree, which generates the output carries for each bit. Finally, sum is generated in the last stage.

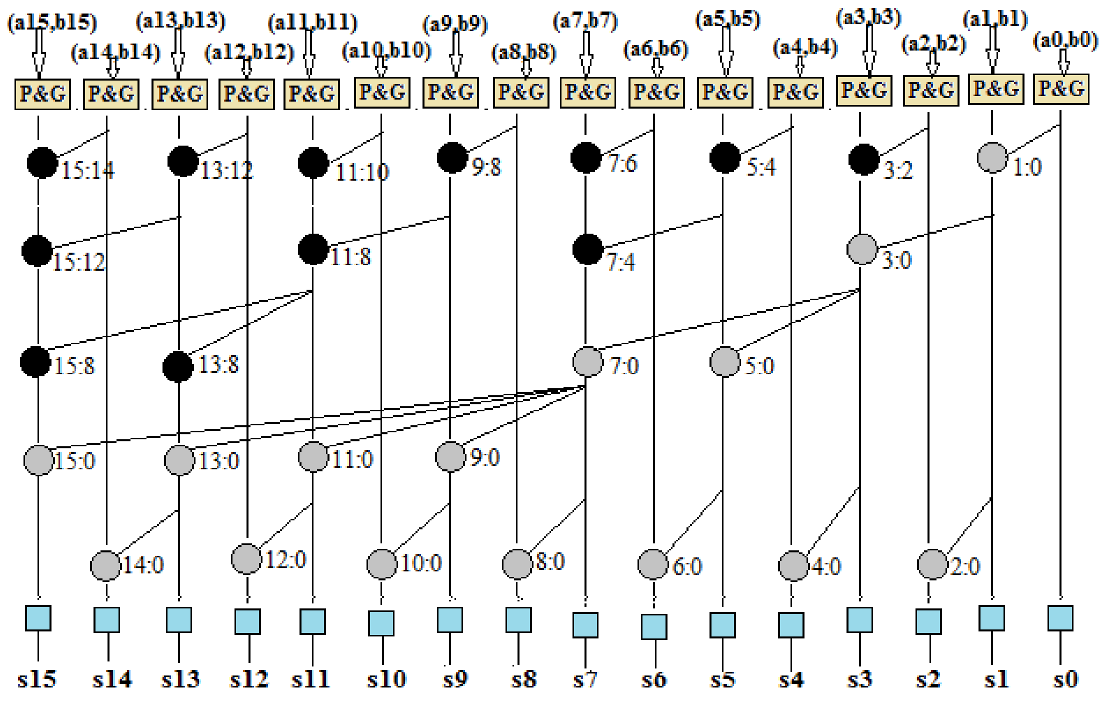
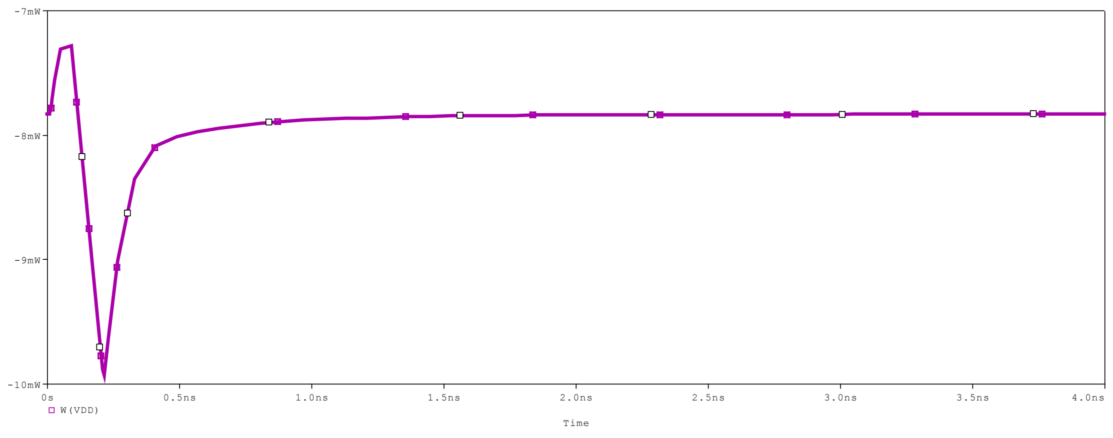
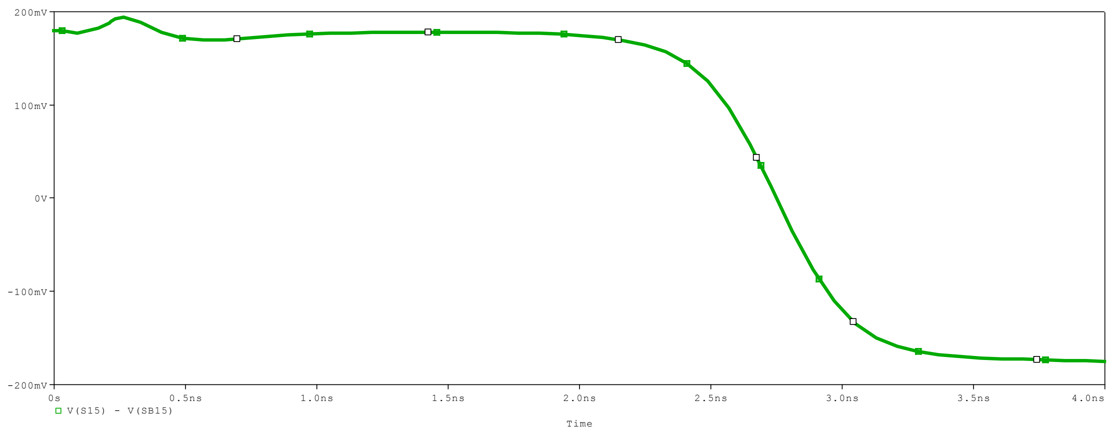


Fig. 4.14 16-bit Ladner Fischer parallel prefix adder

This adder was simulated with its internal components i.e. logic gates designed using MCML topology. The adder results for worst case inputs i.e. input of LSB taken as 1 (A_0=1 and B_0=1) rest all inputs taken as 1 and 0 (A=1 and B=0) is shown in fig. 4.15. The power dissipation plot is shown in fig. 4.15(a) and the worst case delay in sum bit (S_{15}) is shown in fig. 4.15(b). The measured delay associate with the generation of output sum from simulation is 2.6428ns. Also, the circuit dissipates power equal to 9.924mW.



(a)



(b)

Fig. 4.15 Results of 16-bit MCML Ladner Fischer parallel prefix adder (a) Power dissipation (b) Worst case delay in Sum

4.3.4 Brent Kung parallel prefix adder

This adder employs Brent Kung prefix structure in the middle stage to compute output carries. The 16-bit parallel prefix adder structure employing Brent Kung tree in it is shown in fig 4.16. Similarly processing of 16 bit inputs is done for this adder as discussed for the many other structures in this section.

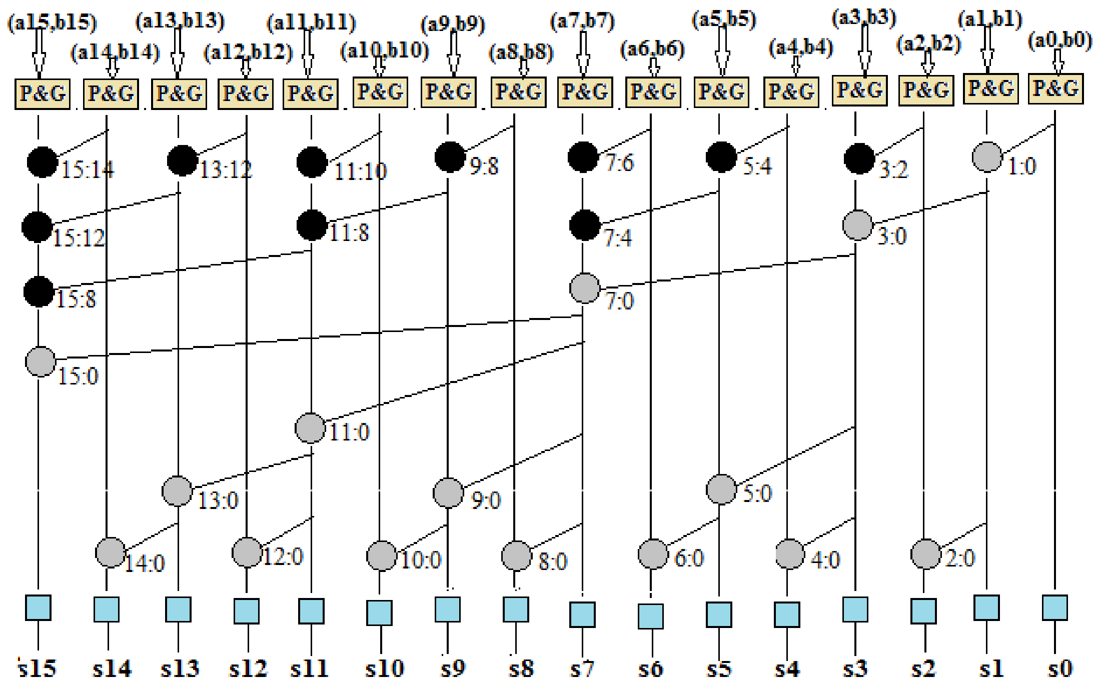
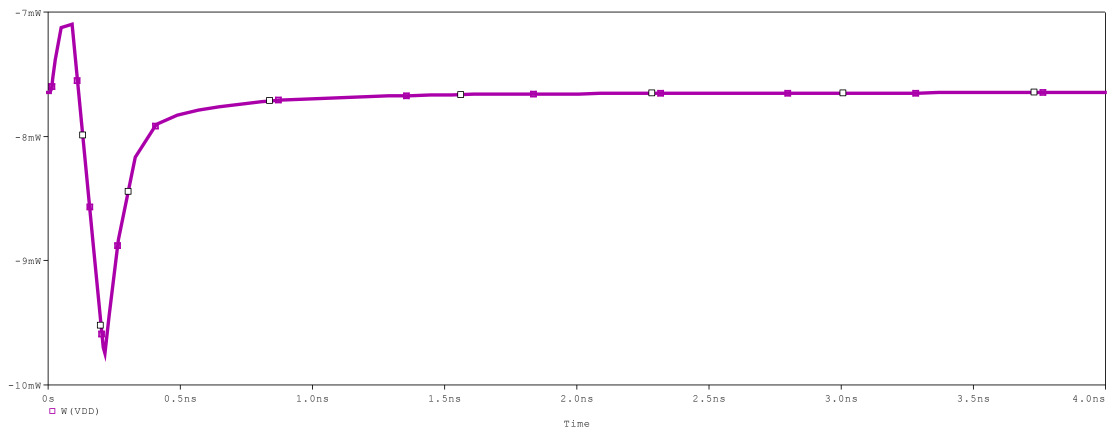
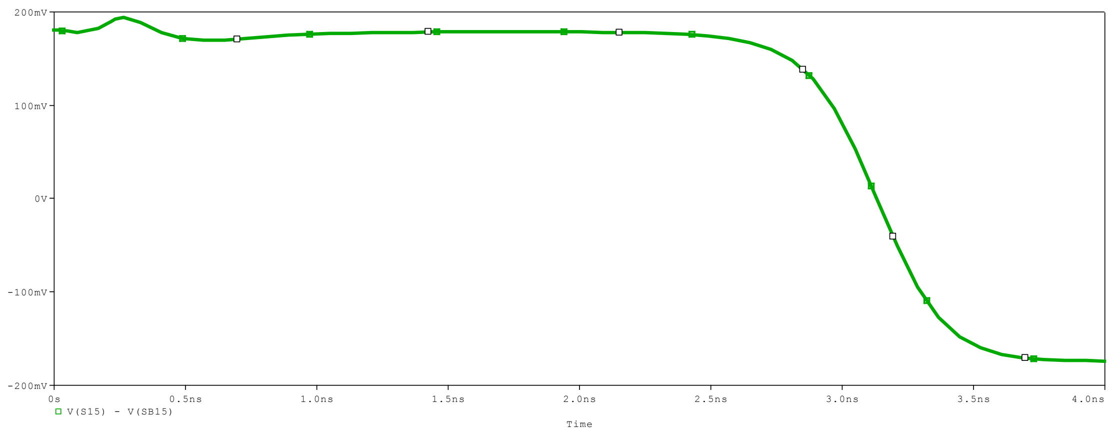


Fig. 4.16 16-bit Brent Kung parallel prefix adder

This adder was designed using MCML gates. The simulation results verify the performance of this adder. The adder results for worst case inputs i.e. input of LSB taken as 1 (A_0=1 and B_0=1) rest all inputs taken as 1 and 0 (A=1 and B=0) is shown in fig. 4.15. The power dissipation plot is shown in fig. 4.15(a) and the worst case delay in sum bit (S_{15}) is shown in fig. 4.15(b).The measured delay associate with the generation of output sum from simulation is 3.0211ns. Also, the circuit dissipates power equal to 9.943mW.



(a)



(b)

Fig. 4.17 Results of 16-bit MCML Brent Kung parallel prefix adder (a) Power dissipation (b) Worst case delay in Sum

4.3.5 Han Carlson parallel prefix adder

This adder shown in fig. 4.18 performs 16-bit addition. So, 16 bit inputs, $a_0, a_1, \dots \dots \dots . . a_{15}$ and $b_0, b, \dots \dots \dots . . b_{15}$ are fed to the adder.

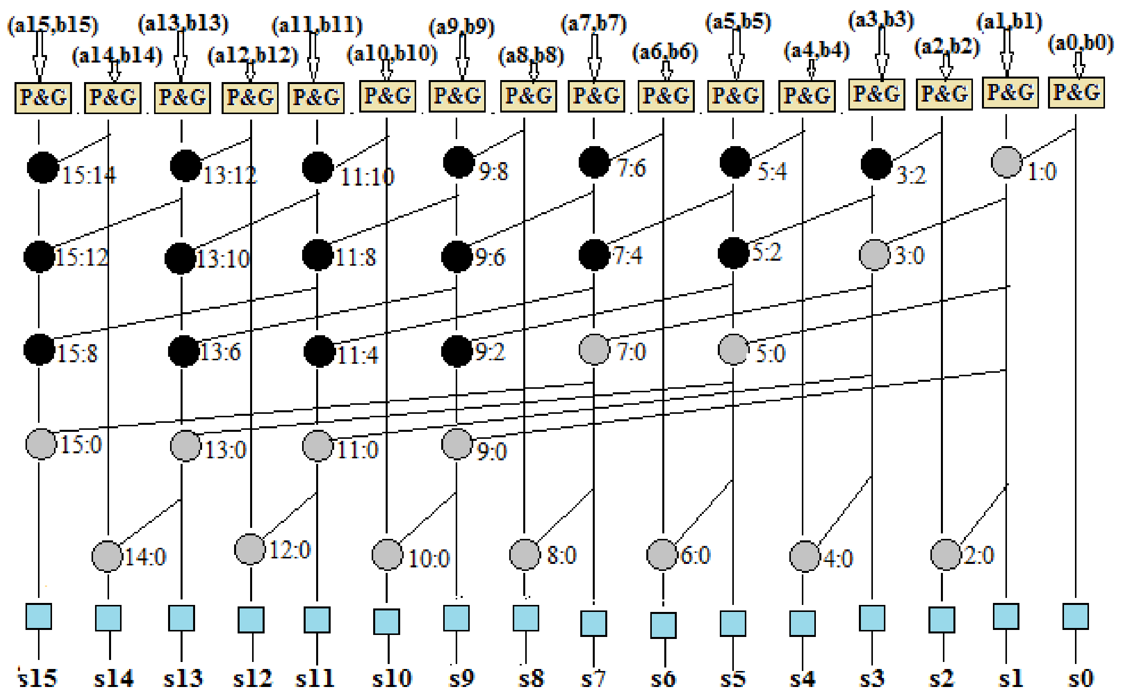
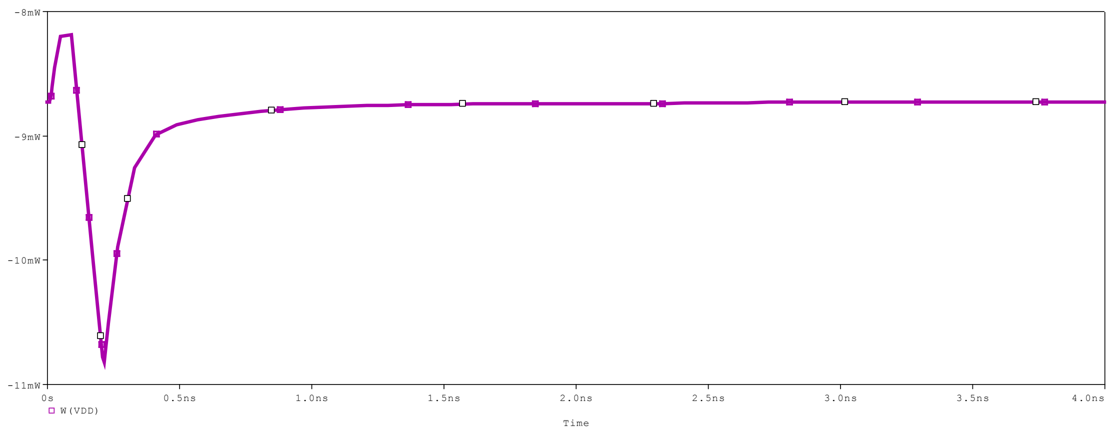
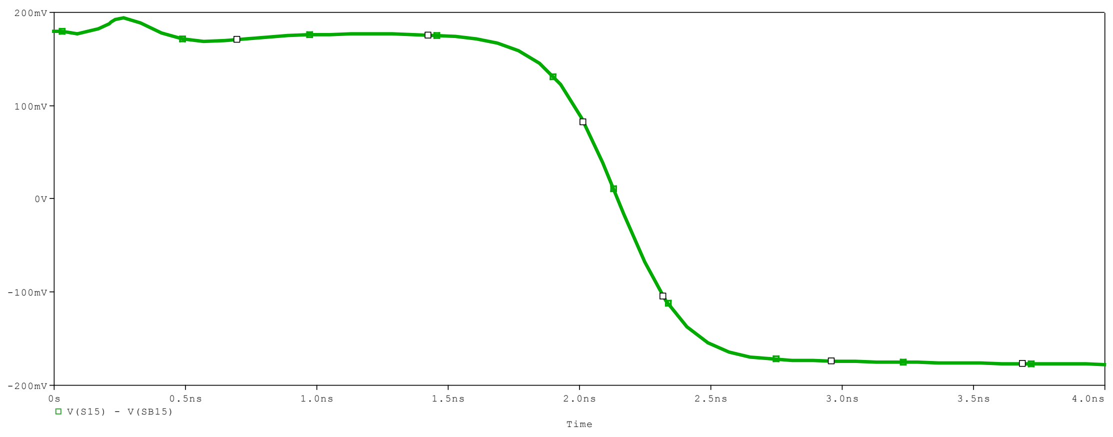


Fig. 4.18 16-bit Han Carlson parallel prefix adder

Initially, 16 P&G blocks individually produces 16 propagate and generate signals of each bit. These signals are processed by Han-Carlson tree to propagate and then generate the output carries for each bit. Hence, this adder is termed as Han Carlson parallel prefix adder. Then, at last sum generation block computes the final sum using the carries generated by the intermediate tree structure. This adder structure has logarithmic delay. The MCML realization of this adder was performed to verify the functionality and satisfactory results were obtained. The plot of power dissipation and worst case delay in sum is shown in fig. 4.19(a) and fig. 4.19(b) respectively. The worst case delay is attained when first LSB inputs are one (A_0=1 and B_0=1) and all rest are fed with A=1 and B=0.



(a)



(b)

Fig. 4.19 Results of 16-bit MCML Han Carlson parallel prefix adder (a) Power dissipation (b) Worst case delay in Sum

This produces the worst case delay at the output. The results are obtained for MSB i.e. S_{15}. Han Carlson tree structure lies between Kogge Stone and Brent Kung, therefore its delay also falls between them. The comparison of delay of parallel prefix adders with various tree structures is presented at the end of this chapter. The measured delay of this adder associate with the generation of output sum from simulation is 2.0104ns. Also, the circuit dissipates power equal to 10.862mW.

4.3.6 Knowles [2,1,1,1] parallel prefix adder

The Knowles prefix structure is explained in the previous section and here it is used to form the adder structure. The 16-bit parallel prefix adder structure employing Knowles [2,1,1,1] tree in it is shown in fig 4.20.

The 16 bit inputs, $a_0, a_1, \dots \dots \dots . . a_{15}$ and $b_0, b, \dots \dots \dots . . b_{15}$ are fed to 16 P&G blocks individually which gives 16 propagate and generate signals. These signals are used by Knowles [2,1,1,1] tree to propagate and compute the output carries of each bit. Then last block which is sum generation block calculates the final sum using the carries generated by the tree structure. The Knowles [2,1,1,1] is explained in 4.2.6. The MCML realization of this adder was performed and satisfactory results were obtained.

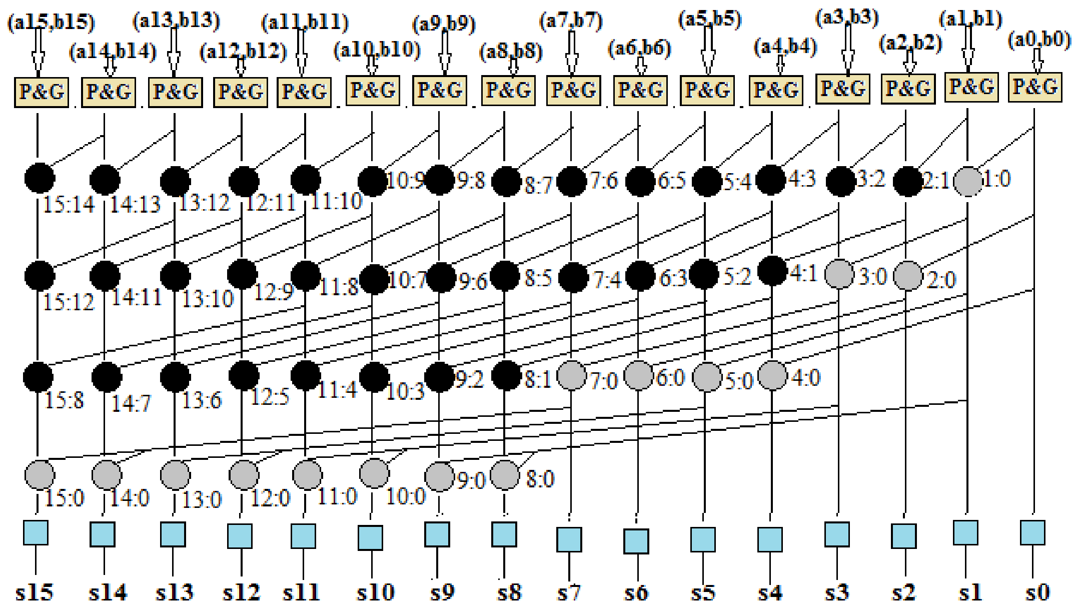
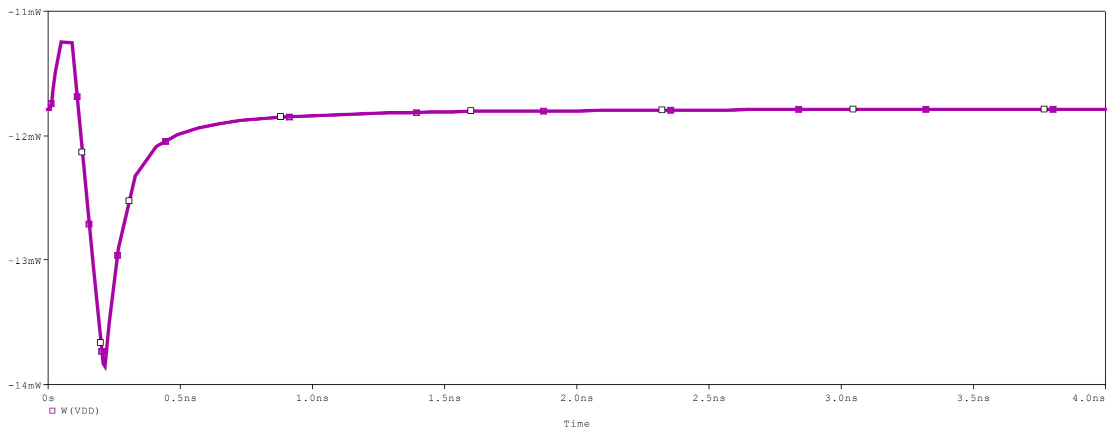
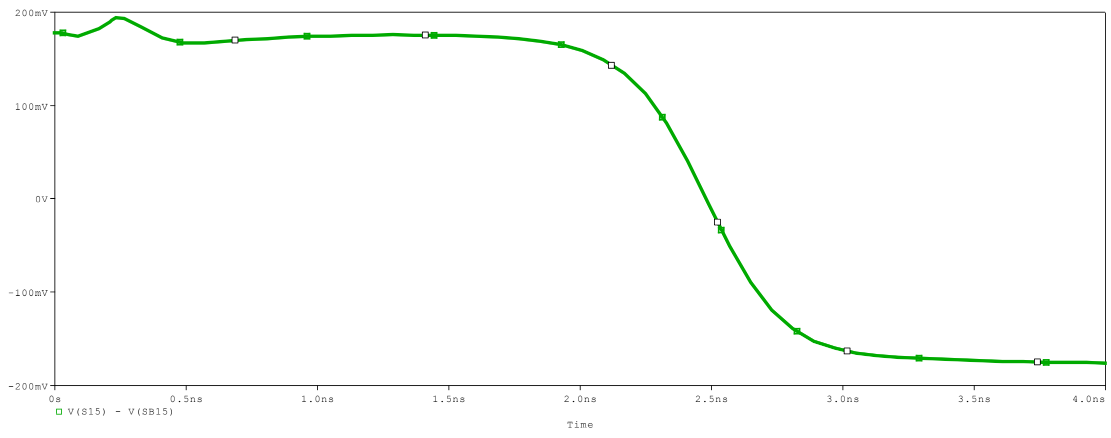


Fig. 4.20 16-bit Knowles parallel prefix adder

The plot of power dissipation and worst case delay in sum is shown in fig. 4.21(a) and fig. 4.21(b) respectively. The worst case condition is when LSB inputs are one (A_0=1 and B_0=1) and remaining are A=1 and B=0 for all bits. The measured delay associate with the generation of output sum from simulation is 2.3698ns. Also, the circuit dissipates power equal to 13.855mW.



(a)



(b)

Fig. 4.21 Results of 16-bit MCML Knowles parallel prefix adder (a) Power dissipation (b) Worst case delay in Sum

4.4 Parallel prefix implementation in Ling Adder

The basic theory of ling adder was discussed in section 3.5. Here its implementation for parallel-prefix approach is elaborated. Ling implementation uses modified generate and propagate bits to compute ling carry which are defined as [34]:

$$G_i{}^* = g_i + g_{i-1} \qquad (4.5)$$

and

$$P_i{}^* = p_i.p_{i-1} \qquad (4.6)$$

With $g_{-1} = p_{-1}$ =0 and $G_m{}^*$=$P_m{}^*$ =0 for m<0.

Consider the ling carries at fourth and fifth bit position is equal to

$$H_4 = g_4 + g_3 + p_3.g_2 + p_3.p_2.g_1 + p_3.p_2.p_1.g_0 \qquad (4.7)$$

$$H_5 = g_5 + g_4 + p_4.g_3 + p_4.p_3.g_2 + p_4.p_3.p_2.g_1 + p_4.p_3.p_2.p_1.g_0 \qquad (4.8)$$

As $g_i.p_i = g_i$, so we can rewrite eq. (4.7) and (4.8) as:

$$H_4 = g_4 + g_3 + p_3.p_2.(g_2 + g_1) + p_3.p_2.p_1.p_1.g_0 \qquad (4.9)$$

$$H_5 = g_5 + g_4 + p_4.p_3.(g_3 + g_2) + p_4.p_3.p_2.p_1.(g_1 + g_0) \qquad (4.10)$$

Using (4.5) and (4.6), eq. (4.7) and (4.8) can be rewritten as:

$$H_4 = G_4^* + P_3^*.G_2^* + P_3^*.P_1^*.G_0^* \qquad (4.11)$$

$$H_5 = G_5^* + P_4^*.G_3^* + P_4^*.P_2^*.G_1^* \qquad (4.12)$$

Now using dot operator (**o**) we can rewrite eq. (4.11) and (4.12) as:

$$H_4 = (G_4^*, P_3^*)\boldsymbol{o}(G_2^*, P_1^*)\boldsymbol{o}(G_0^*, P_{-1}^*) \qquad (4.13)$$

$$H_5 = (G_5^*, P_4^*)\boldsymbol{o}(G_3^*, P_2^*)\boldsymbol{o}(G_1^*, P_0^*) \qquad (4.14)$$

It can be seen that for an N-bit adder the Ling carries H_i and $H_{i+1}$of consecutive even and odd bit positions i and i+1 respectively, in general is given by:

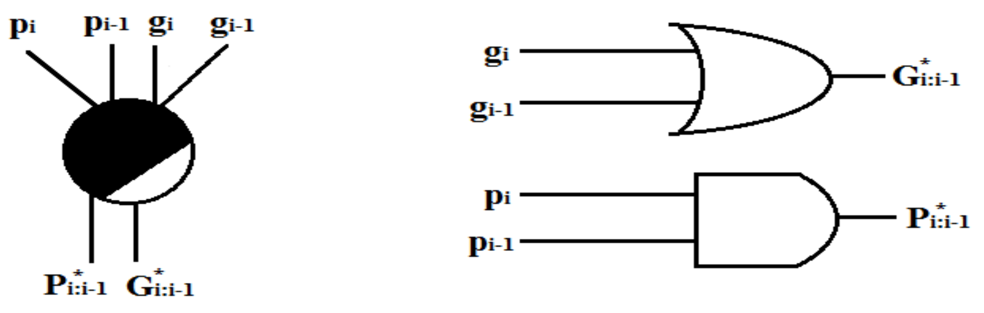
$$H_i = (G_i^*, P_{i-1}^*)\boldsymbol{o}(G_{i-2}^*, P_{i-3}^*)\boldsymbol{o} \dots \boldsymbol{o}(G_0^*, P_{-1}^*) \qquad (4.15)$$

$$H_{i+1} = (G_{i+1}^*, P_i^*)\boldsymbol{o}(G_{i-1}^*, P_{i-2}^*)\boldsymbol{o} \dots \boldsymbol{o}(G_1^*, P_0^*) \qquad (4.16)$$

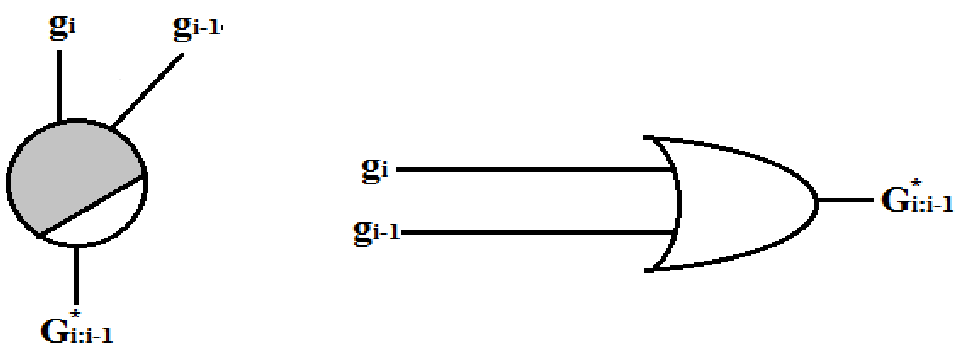
The parallel prefix-computation of ling carry, H_i using above formulation employs separate prefix trees for even and odd bit positions.

These terms are computed independently; thereby reducing the fan-out of the parallel-prefix structure, this fact also contributes to delay reduction [35].

The modified generate and propagate bits used in Ling scheme is implemented in prefix tree by new modified cells, they are; Ling black cell and Ling grey cell. The cells along with their internal gate structure are shown in fig. 4.22; also they are described by the equation (4.5) and (4.6).



(a)



(b)

Fig. 4.22 Ling Cells (a) Ling Black Cell (b) Ling Grey Cell

Implementation of parallel prefix ling adder is same as that of parallel prefix CLA except that, level one of these adder structures computes modified generate and modified propagate bits using ling cells instead of normal cells. Rest of the prefix structure remains same. So, Ling cells along with normal cells are used to construct the complete prefix structure. The prefix structure outputs ling carries instead to real carries. Therefore, to compute the actual carries modified sum generation block is used. This block is shown in fig. 4.23. This block computes the actual carry with the help of multiplexer. The expression presenting the sum generation is given by equation (3.28).

The parallel prefix ling adders are constructed using six modified tree structures and they are described in the section to follow. Also these adders are simulated to justify this approach using MCML blocks.

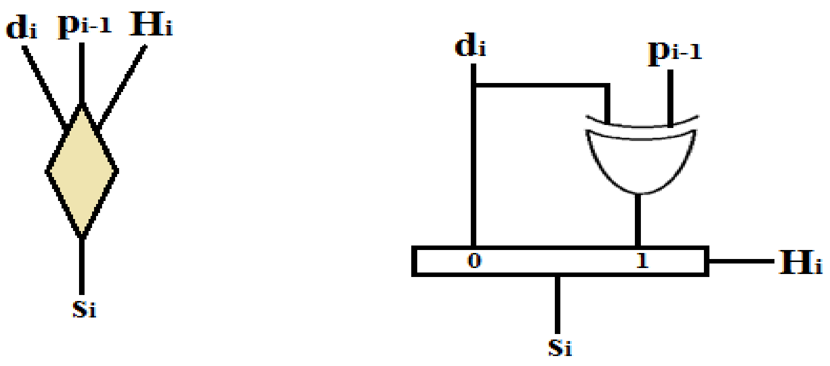


Fig. 4.23 Sum generation block for Ling adders

4.4.1 Kogge Stone parallel prefix Ling adder

The 16-bit parallel prefix adder constructed using Kogge Stone tree and employing ling modifications in it is shown in fig. 4.24.

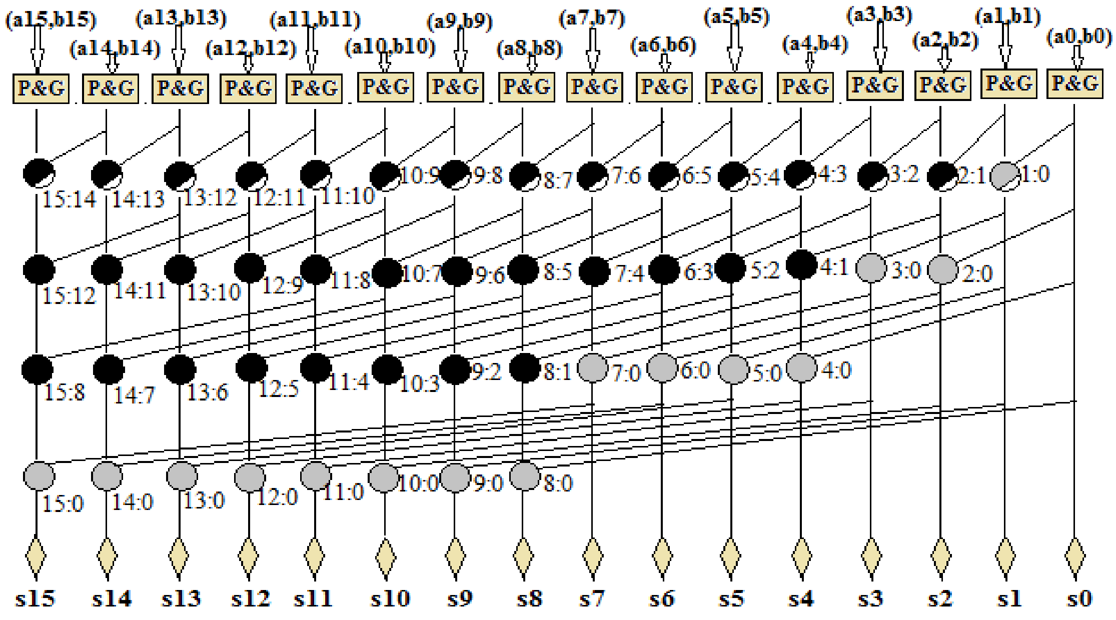
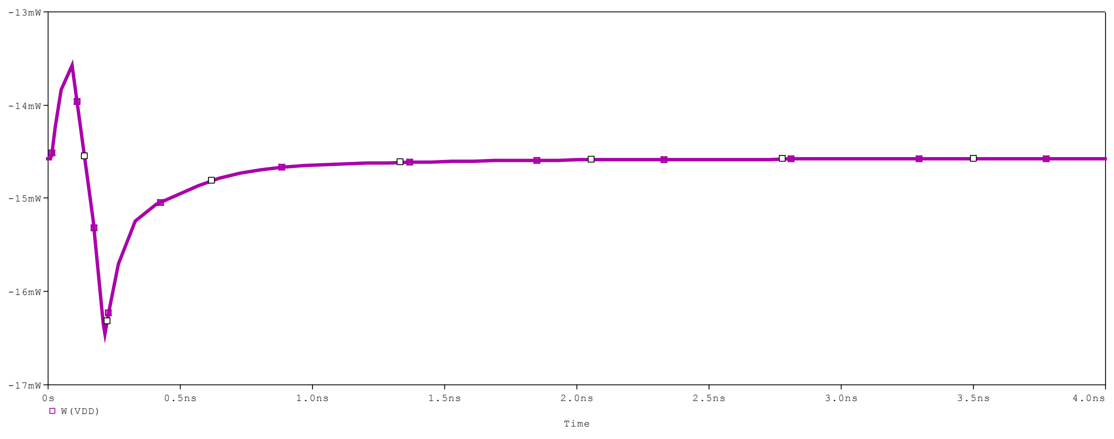
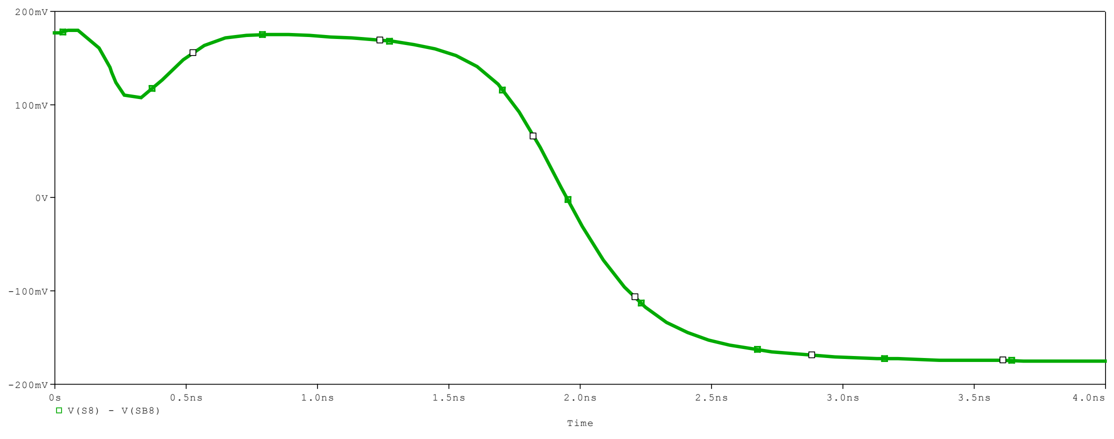


Fig. 4.24 16-bit Kogge Stone parallel prefix Ling adder

The simulated results of the adder displaying the power dissipation and worst case delay in MSB sum (S_{15}) is shown in fig. 4.25(a) and fig. 4.25(b) respectively. The inputs taken for simulation are A_0=1 and B_0=1 and the rest as A=1 and B=0. The measured delay associate with the generation of output sum from simulation is 1.8405ns. Also, the circuit dissipates power equal to 16.471mW.



(a)



(b)

Fig. 4.25 Results of 16-bit MCML Kogge Stone parallel prefix Ling adder (a) Power dissipation (b) Worst case delay in Sum

4.4.2 Sklansky parallel prefix Ling adder

The 16-bit parallel prefix adder constructed using Sklansky tree and employing ling modifications in it is shown in fig. 4.26.

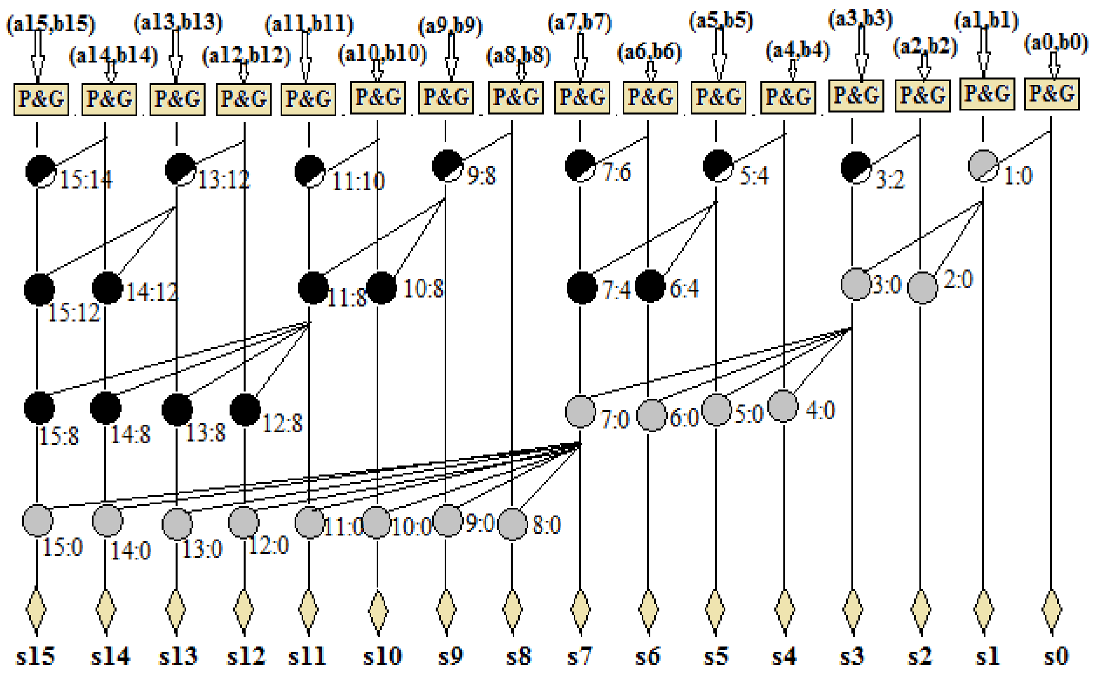
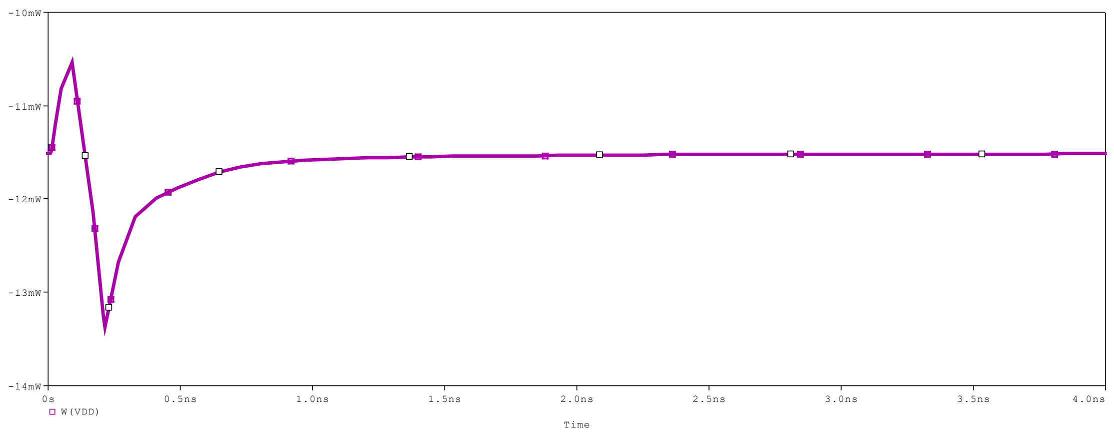


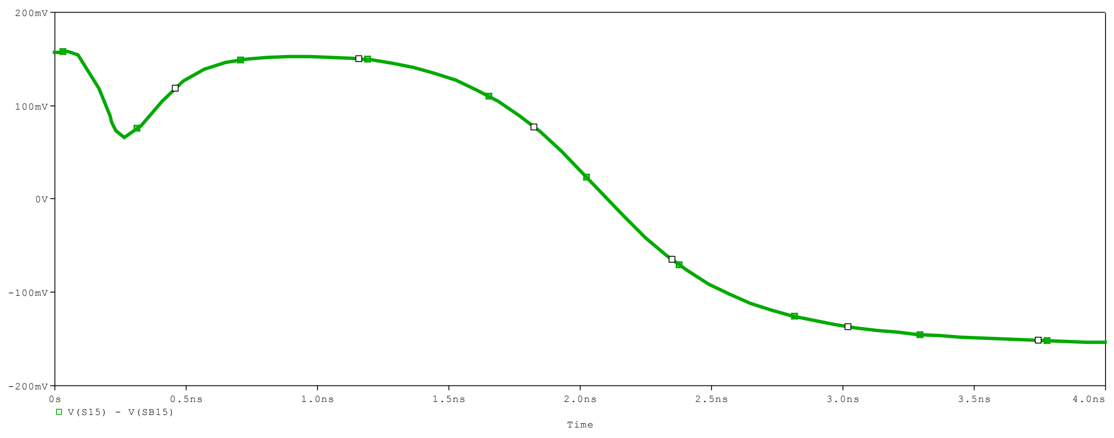
Fig. 4.26 16-bit Sklansky parallel prefix ling adder

The MCML realization of this adder was performed to verify the functionality and satisfactory results were obtained. The plot of power dissipation and worst case delay in MSB sum bit S_{15} is shown in fig. 4.19(a) and fig. 4.19(b) respectively. The worst case delay is taken as LSB inputs one (A_0=1 and B_0=1) and feeding all rest with A=1 and B=0.

The measured delay associate with the generation of output sum from simulation is 1.9882ns. Also, the circuit dissipates power equal to 13.371mW.



(a)



(b)

Fig. 4.27 Results of 16-bit MCML Sklansky parallel prefix Ling adder (a) Power dissipation (b) Worst case delay in Sum

4.4.3 Ladner Fischer parallel prefix Ling adder

The 16-bit parallel prefix adder constructed using Ladner Fischer tree and employing ling modifications in it is shown in fig. 4.28.

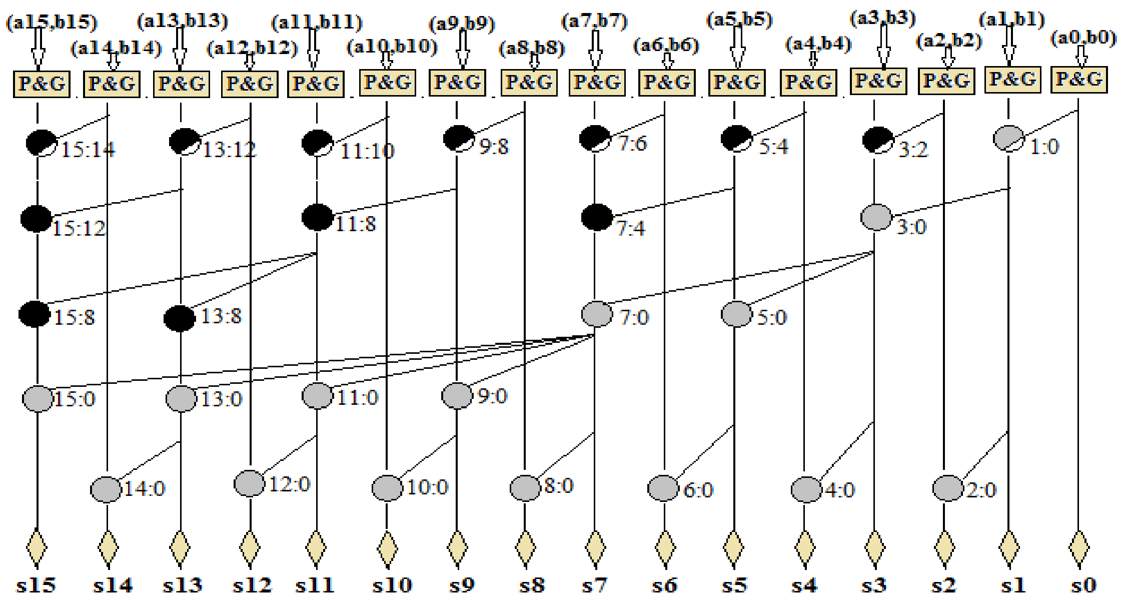
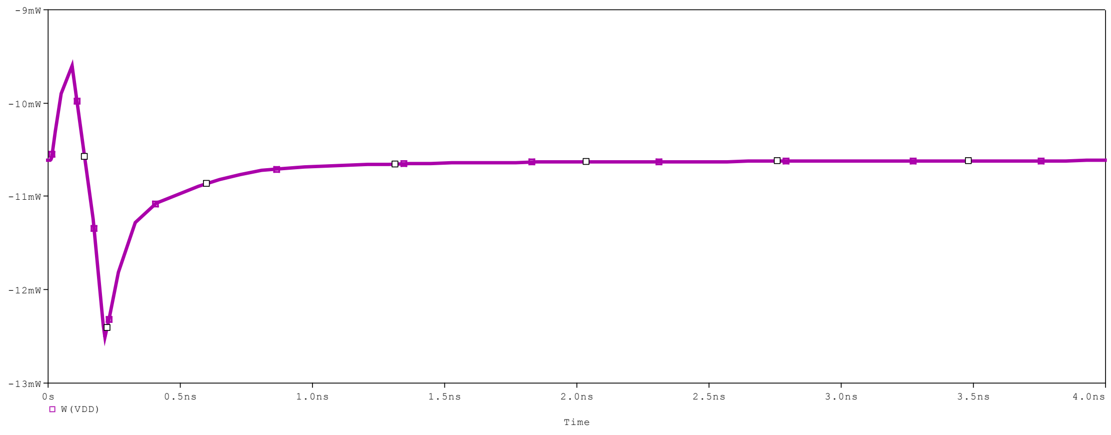
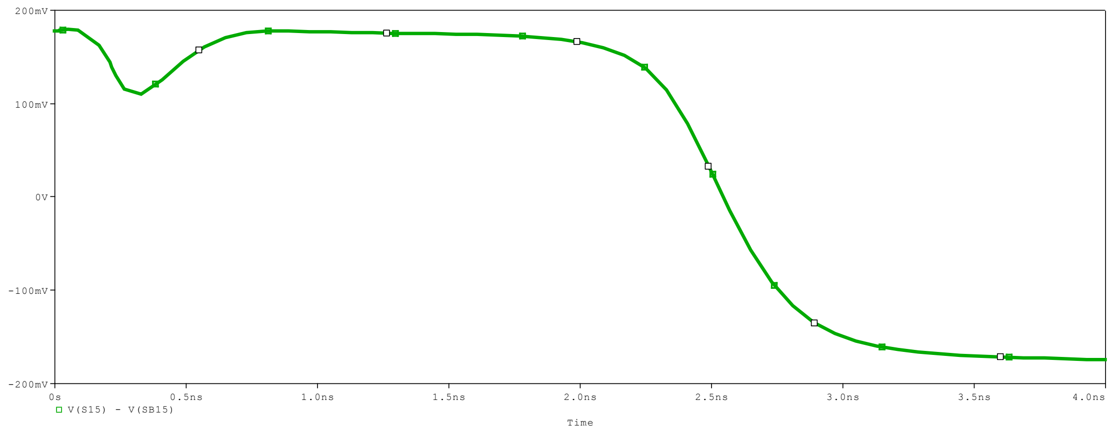


Fig. 4.28 16-bit Ladner Fischer parallel prefix Ling adder

Also the simulated results of the adder displaying the power dissipation and worst case delay in MSB sum (s_{15}) is shown in fig. 4.29(a) and fig. 4.29(b) respectively for inputs A_0=1 B_0=1 and A=1 B=0 for remaining left bits. The measured delay associate with the generation of output sum from simulation is 2.432ns. Also, the circuit dissipates power equal to 12.463mW.



(a)



(b)

Fig. 4.29 Results of 16-bit MCML Ladner Fischer parallel prefix Ling adder (a) Power dissipation (b) Worst case delay in Sum

4.4.4 Brent Kung parallel prefix Ling adder

The 16-bit parallel prefix adder constructed using Brent Kung tree and employing ling modifications in it is shown in fig. 4.30 and the simulated results of the adder displaying the power dissipation and worst case delay in MSB sum (s_{15}) is shown in fig. 4.31(a) and fig. 4.31(b) respectively. The worst case inputs are A_0=1 B_0=1 and A=1 B=0 for remaining all left bits. The measured delay associate with the generation of output sum from simulation is 2.798ns. Also, the circuit dissipates power equal to 12.352mW.

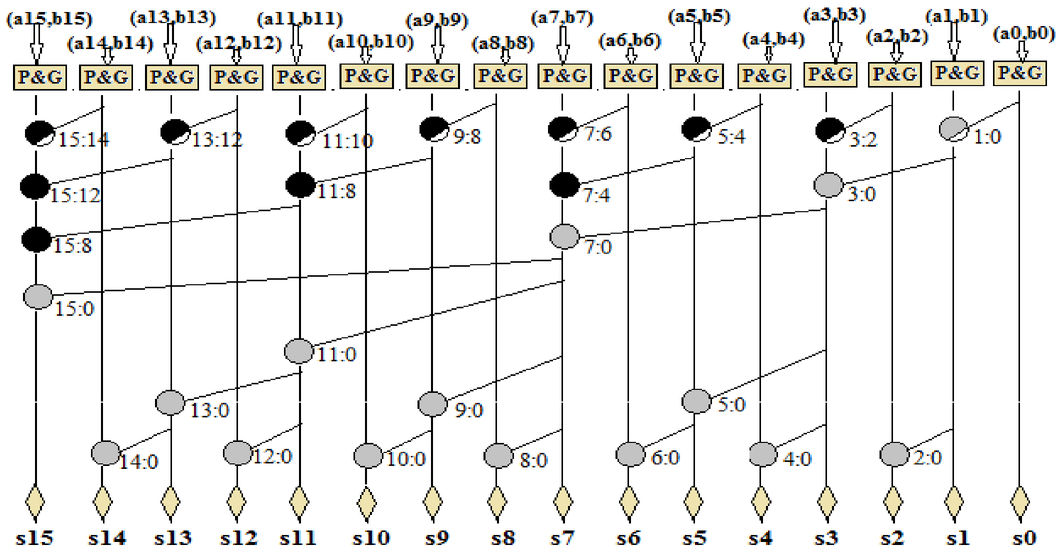
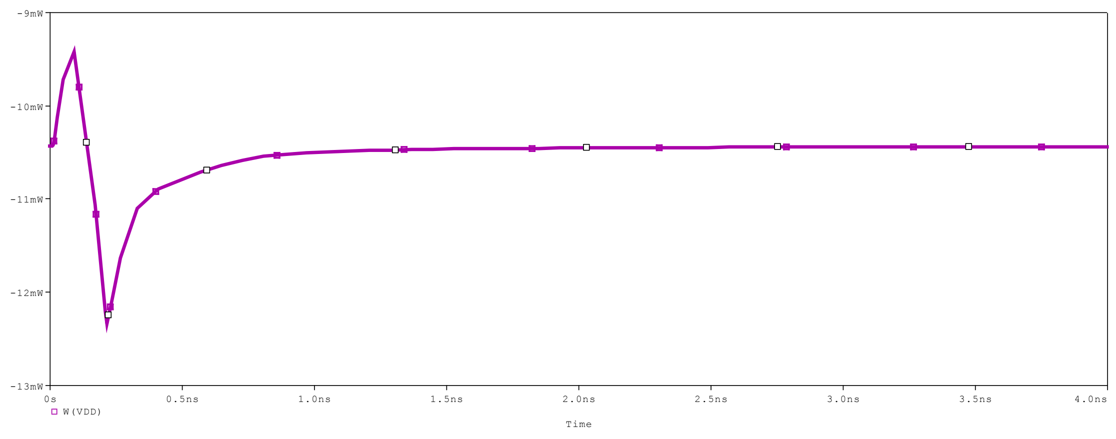
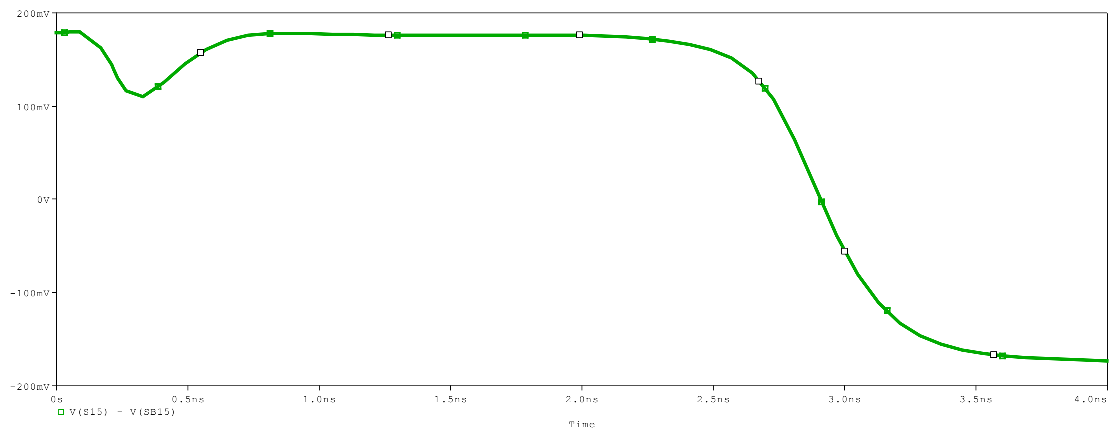


Fig.4.30 16-bit Brent Kung parallel prefix Ling adder



(a)



(b)

Fig. 4.31 Results of 16-bit MCML Brent Kung parallel prefix Ling adder (a) Power dissipation (b) Worst case delay in Sum

4.4.5 Han Carlson parallel prefix Ling adder

The 16-bit parallel prefix adder constructed using Han Carlson tree and employing ling modifications in it is shown in fig. 4.32 and the simulated results of the adder displaying the power dissipation and worst case delay in MSB sum (s_{15}) is shown in fig. 4.33(a) and fig. 4.33(b) respectively. The worst case inputs are A_0=1 B_0=1 and A=1 B=0 for remaining all left bits. The measured delay associate with the generation of output sum from simulation is 1.4816ns. Also, the circuit dissipates power equal to 13.371mW.

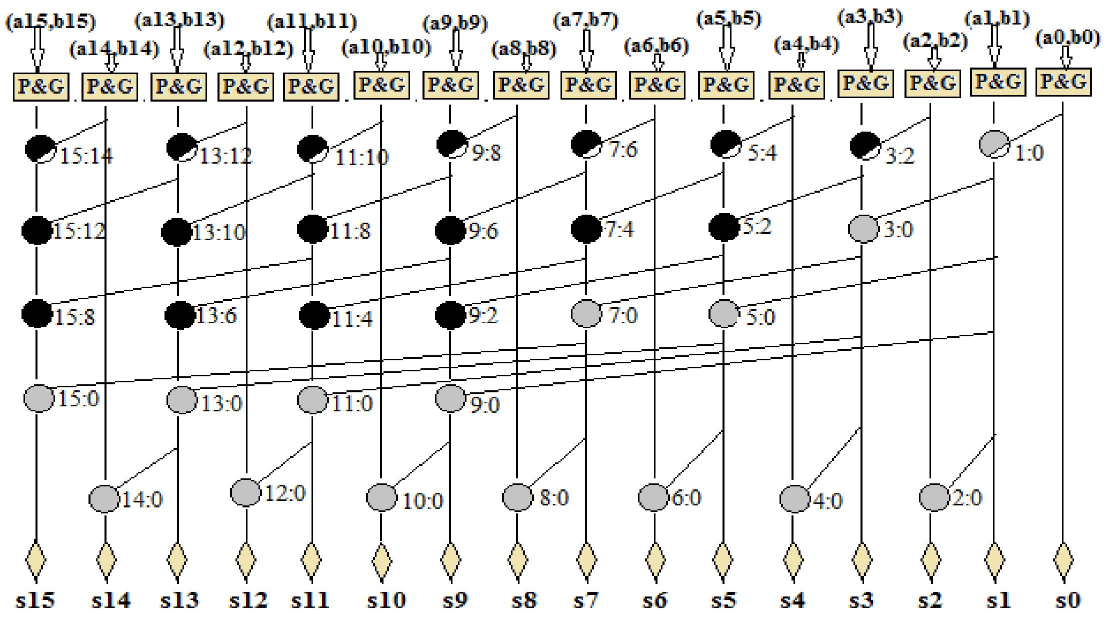
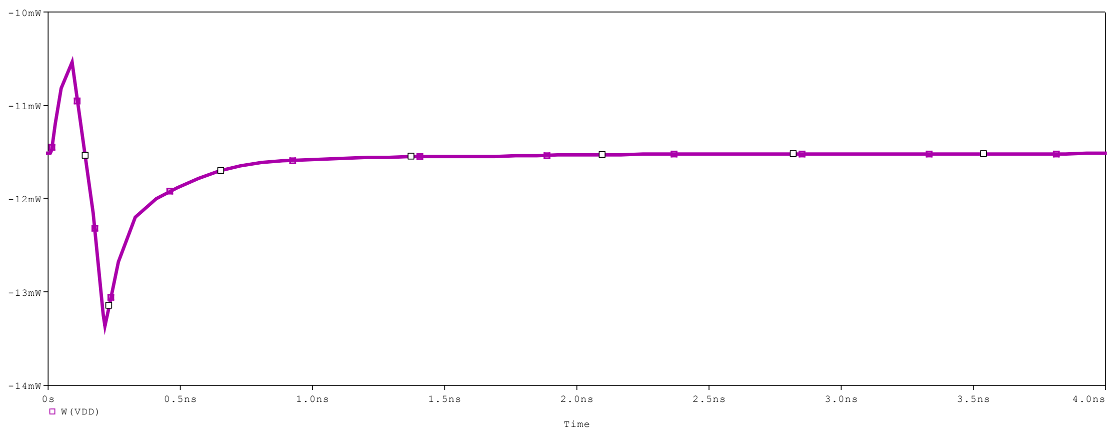
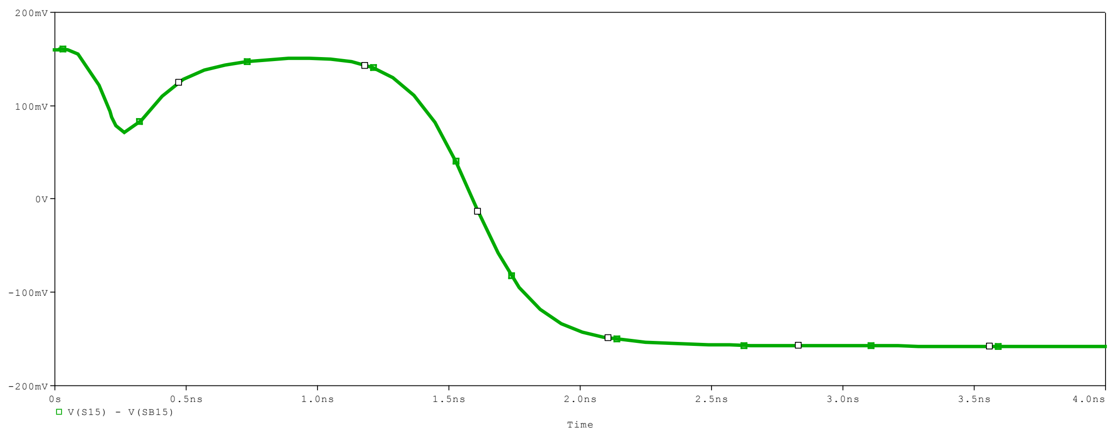


Fig. 4.32 16-bit Han Carlson parallel prefix Ling adder



(a)



(b)

Fig. 4.33 Results of 16-bit MCML Han Carlson parallel prefix Ling adder (a) Power dissipation (b) Worst case delay in Sum

4.4.6 Knowles parallel prefix Ling adder

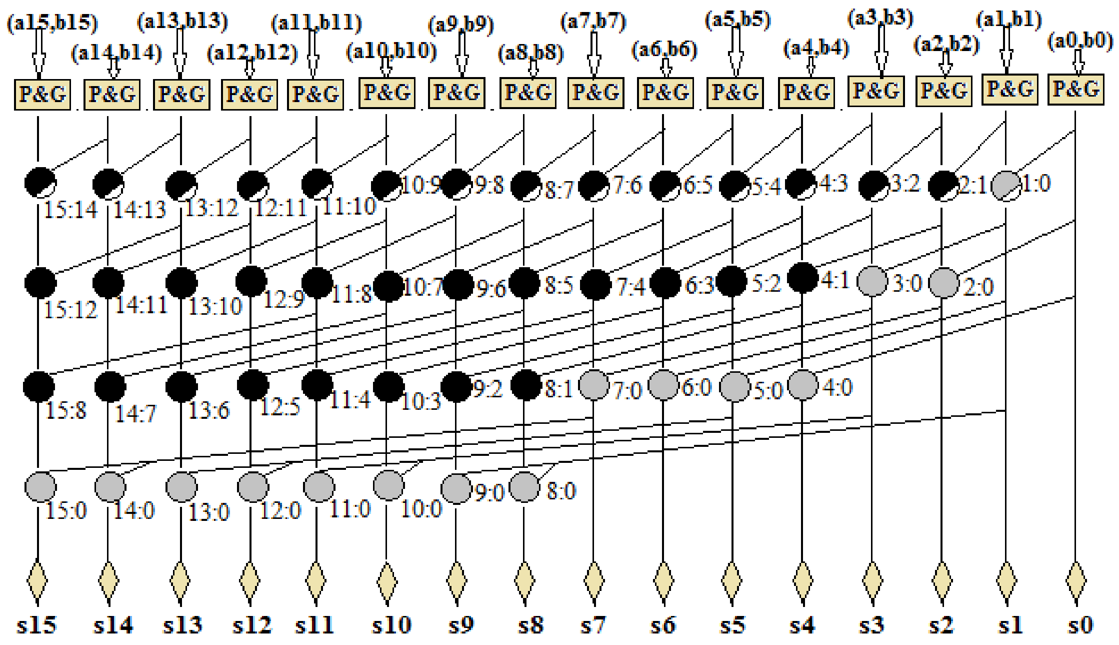
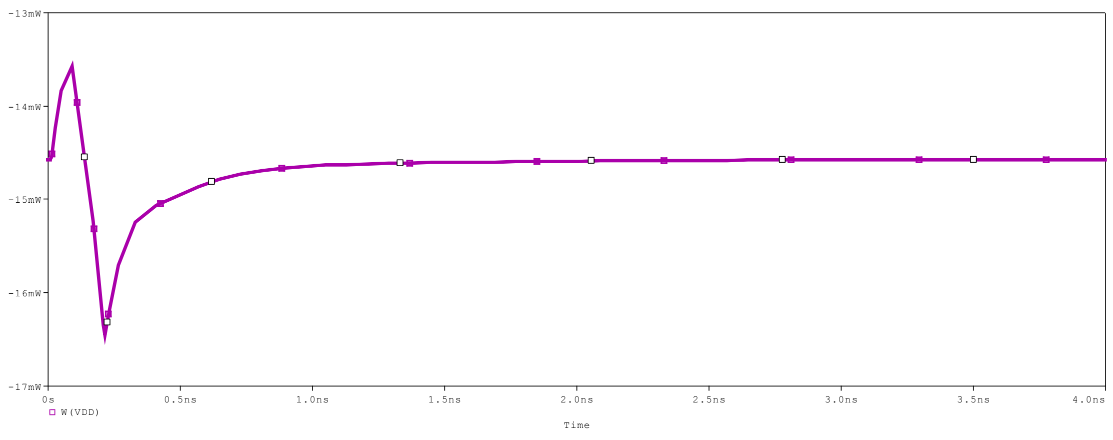
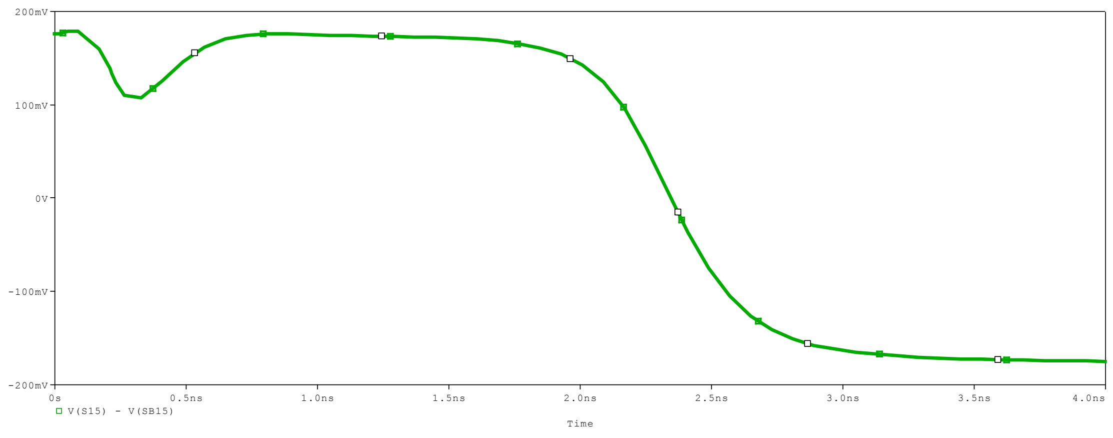


Fig. 4.34 16-bit Knowles parallel prefix Ling adder

The 16-bit parallel prefix adder constructed using Knowles [2,1,1,1] tree and employing ling modifications in it is shown in fig. 4.34 and the simulated results of the adder displaying the power dissipation and worst case delay in MSB sum (s_{15}) is shown in fig. 4.35(a) and fig.35(b) respectively. The worst case inputs are A_0=1 B_0=1 and A=1 B=0 for remaining all left bits. The measured delay associate with the generation of output sum from simulation is 1.8699ns. Also, the circuit dissipates power equal to 16.418mW.



(a)



(b)

Fig. 4.35 Results of 16-bit MCML Knowles parallel prefix Ling adder (a) Power dissipation (b) Worst case delay in Sum

4.5 Result Comparison

The comparison of various designed adders in this chapter is presented in the Table: 4.1.

Table: 4.1 Results of Various Parallel Prefix Adders

ADDER	**Output SUM (nS)**		**Power (mW)**	
Tree Structures	**Parallel-prefix adder**	**Parallel-prefix Ling adder**	**Parallel-prefix adder**	**Parallel-prefix Ling adder**
Brent Kung	3.0211	2.798	9.943	12.354
Sklansky	2.5431	1.9882	12.186	13.371
Kogge Stone	1.9257	1.8405	13.855	16.471
Han Carlson	2.0104	1.4816	10.862	13.371
Knowles(2,1,1,1)	2.3698	1.8699	13.855	16.418
Ladner Fischer	2.6428	2.432	9.924	12.463

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis focuses on in depth analysis and benefits of using MCML or MOS Current Mode Logic. The transistor level behaviour of MCML circuits were analysed and various gates were designed using it. The implemented basic MCML gates were utilized to design bigger adder structures.

First, the 16 bit RCA was designed using 16 1-bit full adders which were implemented with MCML gates and the worst case sum and output carry delays are stated. Then, the carry select adders for addition of large number of bits in the input words were also implemented using MCML topology they include 16-bit MCML Linear carry select adder and 16-bit MCML square root carry select adder design. The proposed 16-bit MCML square root carry select adder is faster by 67.41% and 68.19% in output carry computation and sum generation respectively in comparison to MCML RCA.

Also, MCML parallel-prefix ling adders for high speed addition are presented. The proposed 16-bit MCML parallel-prefix ling adders have been implemented and their performances are been compared with parallel-prefix adders. The adders are implemented with six tree structures namely Brent Kung, Kogge Stone, Sklansky, Han Carlson, Ladner Fischer and Knowles. It can be concluded from the delay observations that the proposed 16-bit MCML parallel-prefix ling adders reduce the worst case delay significantly in comparison to the MCML parallel-prefix adders. The maximum reduction in delay i.e. 35.69% was found in MCML Han Carlson ling adder whereas the minimum delay of 4.62% was observed in MCML Kogge Stone Ling adder. Also, the delay reduction by 26.73%, 26.40%, 8.667% and 7.9% was examined in 16-bit MCML parallel-prefix ling adders with Knowles, Sklansky, Ladner Fischer and Brent Kung tree structures respectively in comparison to MCML parallel prefix adders with same tree structures.

5.2 Future Work

While this work attempted to analyze many different facts of MCML operation, there were several areas which still require more research. The first goal of any future work would be to design other modified adder topologies using MCML gates which are not proposed yet, to reduce the carry propagation delay, which is critical for high speed applications.

The second area of future work would be to implement resistive load of MCML with other techniques like active shunt peaking to reduce the leakage and thereby reducing the power dissipation. Also included in this future work would be the design of the MCML with feedback for high gain and enhanced speed operation.

A third area for future research would be a much detailed analysis of the effects of subthreshold leakage in MCML. The MCML gate design procedure should be modified to make it operate in subthreshold regime. This will reduce the leakage power dissipation making the circuit operation feasible for power critical applications.

PUBLICATIONS

A research article titled “**A Novel High Speed MCML Square Root Carry Select Adder for Mixed-Signal Applications**” has been published in proceedings of IEEE International Conference on Multimedia Signal Processing and Communication Technologies, pp. 194-197, 2013.

REFERENCES

[1] O.J. Bedrij, “Carry-Select Adder”. *IEEE Transaction on Electronic Computers*, Vol. EC-11, Issue-3, pp. 340-346, June 1962.

[2] V. G. Oklobdzija, “Simple and efficient CMOS circuit for fast VLSI adder realization”, *proceedings of IEEE International Symposium on Circuit and System*, vol. 1, pp. 235-238, 1988.

[3] R.W. Doran, “Variants of an improved carry look-ahead adder”, *IEEE Transactions on Computers*, Vol. 37, Issue 9, pp. 1110-1113, 1988.

[4] A. Tyagi, “A reduced area scheme for carry select adders”, *proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 255-258, 1990.

[5] F. Lu and H. Samueli, “A high-speed CMOS full-adder cell using a new circuit design technique-adaptively-biased pseudo-NMOS logic”, *proceedings of IEEE International Symposium on Circuit and System*, vol. 1, pp. 562-565, 1990. .

[6] H. G. Tamar, A. G. Tamar, K. Hadidi and A. Khoei, “High speed area reduced 64-bit static hybrid carry-lookahead/carry-select adder”, *proceedings of IEEE International Conference on Electronics*, Circuit and System, pp. 460-463, 2011.

[7] Subodh Wairya, Rajendra Kumar Nagaria, and Sudarshan Tiwari, “Performance Analysis of High Speed Hybrid CMOS Full Adder Circuits for Low Voltage VLSI Design”, *VLSI Design*, vol. 2012, Article ID 173079, 18 pages, 2012.

[8] Deepa Yagain, Vijaya Krishna A, and Akansha Baliga, “Design of High-Speed Adders for Efficient Digital Design Blocks,” *ISRN Electronics*, vol. 2012, Article ID 253742, 9 pages, 2012.

[9] S. Parmar and K. P. Singh, "Design of high speed hybrid carry select adder", *Proceedings of IEEE 3rd International Advance Computing Conference*, pp.1656-1663, 2013.

[10] G. Caruso, "Power-aware design of MCML logarithmic adders", *proceedings of International Conference on Signals and Electronics System*, pp. 281-283, 2010.

[11] H. Ling, "High-speed binary adder", *IBM Journal of Research and Development*, Vol. 25, pp. 156-166, 1981.

[12] J. M. Musicer and J. Rabaey, "MOS Current Mode Logic for Low Power, Low Noise, CORDIC Computation in Mixed-Signal Environments", *Proceedings of the International Symposium of Low Power Electronics and Design*, pp. 102-107, 2000.

[13] M. Yamashina and H. Yamada, "An MOS current mode logic (MCML) circuit for low-power sub-GHz processors", *IEICE Transactions on Electronics*, Vol. E75-C, pp. 1181–1187, 1992.

[14] L. Li, S. Raghavendran and D. T. Comer, "CMOS Current Mode Logic Gates for High-Speed Applications," *proceedings of 12th NASA Symposium on VLSI Design*, 2005.

[15] T. K. Agarwal, A. Sawhney, A. K. Kureshi and M. Hasan, "Performance Comparison of Static CMOS and MCML gates in sub- threshold region of operation for 32nm CMOS Technology", *Proceedings of the International Conference on Computer and Communication Engineering*, pp. 284-287, 2008.

[16] Mohamed Azaga and Masuri Othman, "Source Couple Logic (MCML): Theory and Physical Design", *American Journal of Engineering and Applied Sciences 1*, Vol. 1, pp. 24-32, 2008.

[17] M. Alioto and G. Palumbo, "Model and Design of Bipolar and MOS Current-Mode logic (CML, ECL and MCML Digital Circuits)", *Kluwer Academic Publishers*, Springer, New York, 2005.

[18] M. Azaga and M. Othman, "Source-Coupled Logic (SCL): Operation and Delay Analysis", *proceedings of IEEE International Conference on Semiconductor Electronics*, pp. 392-396, 2006.

[19] M. Alioto, G. Palumbo and S. Pennisi, "Modelling of Source-Coupled Logic Gates", *International Journal of Circuit Theory and Applications*, Vol. 30, Issue 4, pp. 459–477, 2002.

[20] Giuseppe Caruso, "Design of MOS Current Mode Logic Gates – Computing the Limits of Voltage Swing and Bias Current", *proceedings of IEEE International Symposium on Circuit and System*, Vol. 6, pp. 5637 – 5640, 2005.

[21] Armin Tajalli, Yusuf Leblebici, "Extreme Low- Power Mixed Signal IC Design", Springer Science, US, 2010.

[22] A. Tajalli and Y. Leblebici, "Design Trade-offs in Ultra Low Power Digital Nanoscale CMOS", *IEEE Transactions on Circuits and Systems 1: Regular Papers*, Vol. 58, Issue 9, pp. 2189-2200, 2011.

[23] M. Alioto and G. Palumbo, "Design strategies for source coupled logic gates", *IEEE Transactions on Circuits and Systems 1: Fundamental Theory and Applications*, Vol. 50, Issue 5, pp. 640-654, 2003.

[24] J. Rabaey, and N. Chandrakaran, "Digital Integrated Circuits: A Design Perspective", Prentice Hall, 2003.

[25] K. Gupta, N. Pandey, M. Gupta, ''A Novel Active Shunt-Peaked MCML-based Four-Bit Ripple-Carry Adder,'' *in proceedings of IEEE International Conference on Computer and Communication Technology*, pp. 285-289, 2010.

[26] Kirti Gupta, Radhika, Neeta Pandey and Maneesha Gupta, "A Novel High Speed MCML Square Root Carry Select Adder for Mixed-Signal Applications", *Proceedings of IEEE International Conference on Multimedia Signal Processing and Communication Technologies*, pp. 194-197, 2013.

[27] Deepa Yagain, Vijaya Krishna A, and Akansha Baliga, “Design of High-Speed Adders for Efficient Digital Design Blocks,” *ISRN Electronics*, vol. 2012, Article ID 253742, 9 pages, 2012.

[28] R.E. Ladner and M. J. Fischer, “Parallel prefix computation*”, Journal of the ACM*, vol. 27, no.4, pp. 831-838, 1980.

[29] P.M. Kogge and H.S. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence equations”, *IEEE Transactions on Computers*, Vol. C-22, No. 8, pp. 786-793, 1973.

[30] J. Sklansky, “Conditional-sum addition logic”, *IRE Transactions on Electronic Computers*, vol.9, pp. 226-231, 1960.

[31] R. P. Brent Kung and H. T. Kung, “A regular Layout for Parallel Adders”, *IEEE Transactions on Computers*, vol. C-31, no.3, pp. 260-264, 1982.

[32] T. Han and D. Carlson, “Fast area-efficient VLSI adders”, in *Proceedings of IEEE Symposium on Computer Arithmetic*, pp. 49-56. 1987.

[33] S. Knowles, “A family of adders”, *in Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pp. 227-281, 2001.

[34] Giorgos Dimitrakopoulos, Dimitris Nikolos, “High-Speed Parallel-Prefix VLSI Ling Adders*”, IEEE Transactions on Computers*, Vol. 54, No. 2, 2005.

[35] Tso-Bing Juang, Pramod Kumar Mehra, Chung-Chun Kuan, “Area-Efficient Parallel-Prefix Ling Adders”, *IEEE Asia Pacific conference on circuits and systems*, 2010.

APPENDIX

(A) TSMC 0.18µm level 7 parameters for NMOS

```
LEVEL   = 7
+DSUB    = 0.0217897     + TNOM    = 27               TOX     = 4.1E-9
+XJ      = 1E-7                    NCH     = 2.3549E17        VTH0    = 0.3750766
+K1      = 0.5842025     K2      = 1.245202E-3        K3      = 1E-3
+K3B     = 0.0295587     W0      = 1E-7                       NLX     = 1.597846E-7
+DVT0W   = 0             DVT1W   = 0                  DVT2W   = 0
+DVT0    = 1.3022984     DVT1    = 0.4021873          DVT2    = 7.631374E-3
+U0      = 296.8451012   UA      = -1.179955E-9       UB      = 2.32616E-18
+UC      = 7.593301E-11  VSAT    = 1.747147E5         A0      = 2
+AGS     = 0.452647      B0      = 5.506962E-8        B1      = 2.640458E-6
+KETA    = -6.860244E-3            A1      = 7.885522E-4       A2      = 0.3119338
+RDSW    = 105           PRWG    = 0.4826             PRWB    = -0.2
+WR      = 1                       WINT    = 4.410779E-9       LINT    = 2.045919E-8
+XL      = 0                       XW      = -1E-8                     DWG     = -2.610453E-9
+DWB     = -4.344942E-9            VOFF    = -0.0948017        NFACTOR = 2.1860065
+CIT     = 0                       CDSC    = 2.4E-4            CDSCD   = 0
+CDSCB   = 0             ETA0    = 1.991317E-3        ETAB    = 6.028975E-5
+DSUB    = 0.0217897     PCLM    = 1.7062594          PDIBLC1 = 0.2320546
+PDIBLC2 = 1.670588E-3             PDIBLCB = -0.1              DROUT   = 0.8388608
+PSCBE1  = 1.904263E10             PSCBE2  = 1.546939E-8               PVAG    = 0
+DELTA   = 0.01          RSH     = 7.1                        MOBMOD  = 1
+PRT     = 0                       UTE     = -1.5                      KT1     = -0.11
+KT1L    = 0                       KT2     = 0.022                     UA1     = 4.31E-9
```

```
+UB1     = -7.61E-18        UC1     = -5.6E-11                   AT      = 3.3E4

+WL      = 0                        WLN     = 1                          WW      = 0

+WWN     = 1                UWL     = 0                          LL      = 0

+LLN     = 1                        LW      = 0                          LWN     = 1

+LWL     = 0                        CAPMOD  = 2                  XPART   = 0.5

+CGDO    = 6.7E-10          CGSO    = 6.7E-10          CGBO    = 1E-12

+CJ      = 9.550345E-4      PB      = 0.8                        MJ      = 0.3762949

+CJSW    = 2.083251E-10             PBSW    = 0.8                        MJSW    = 0.1269477

+CJSWG   = 3.3E-10          PBSWG   = 0.8              MJSWG   = 0.1269477

+CF      = 0                        PVTH0   = -2.369258E-3               PRDSW   = -1.2091688

+PK2     = 1.845281E-3      WKETA   = -2.040084E-3               LKETA   = -1.266704E-3

+PU0     = 1.0932981        PUA     = -2.56934E-11     PUB     = 0

+PVSAT   = 2E3              PETA0   = 1E-4             PKETA   = -3.350276E-3    )
```

(B) TSMC 0.18µm level 7 parameters for PMOS

```
  LEVEL   = 7

+ TNOM    = 27              TOX     = 4.1E-9

+XJ   = 1E-7                        NCH     = 4.1589E17                  VTH0    = -0.3936726

+K1   = 0.5750728                   K2      = 0.0235926                  K3      = 0.1590089

+K3B= 4.2687016                     W0      = 1E-6                       NLX     = 1.033999E-7

+DVT0W   = 0                DVT1W   = 0                          DVT2W   = 0

+DVT0 = 0.5560978                   DVT1    = 0.2490116                  DVT2    = 0.1

+U0= 112.5106786                    UA      = 1.45072E-9       UB      = 1.195045E-21

+UC      = -1E-10           VSAT    = 1.168535E5       A0      = 1.7211984

+AGS     = 0.3806925        B0      = 4.296252E-7      B1      = 1.288698E-6

+KETA    = 0.0201833        A1      = 0.2328472        A2      = 0.3

+RDSW    = 198.7483291              PRWG    = 0.5                        PRWB    = -0.4971827

+WR      = 1                        WINT    = 0                          LINT    = 2.943206E-8

+XL      = 0                        XW      = -1E-8                      DWG     = -1.949253E-8
```

```
+DWB     = -2.824041E-9           VOFF    = -0.0979832        NFACTOR = 1.9624066

+CIT     = 0                      CDSC    = 2.4E-4            CDSCD   = 0

+CDSCB   = 0             ETA0    = 7.282772E-4    ETAB    = -3.818572E-4

+DSUB    = 1.518344E-3            PCLM    = 1.4728931         PDIBLC1 = 2.138043E-3

+PDIBLC2 = -9.966066E-6   PDIBLCB = -1E-3                     DROUT   = 4.276128E-4

+PSCBE1  = 4.850167E10            PSCBE2  = 5E-10             PVAG    = 0

+DELTA   = 0.01          RSH     = 8.2                        MOBMOD  = 1

+PRT     = 0                      UTE     = -1.5                      KT1     = -0.11

+KT1L    = 0                      KT2     = 0.022                     UA1     = 4.31E-9

+UB1     = -7.61E-18     UC1     = -5.6E-11       AT      = 3.3E4

+WL      = 0                      WLN     = 1                         WW      = 0

+WWN     = 1             WWL     = 0                          LL      = 0

+LLN     = 1                      LW      = 0                         LWN     = 1

+LWL     = 0                      CAPMOD  = 2                 XPART   = 0.5

+CGDO    = 7.47E-10      CGSO    = 7.47E-10       CGBO    = 1E-12

+CJ      = 1.180017E-3   PB      = 0.8560642      MJ      = 0.4146818

+CJSW    = 2.046463E-10           PBSW    = 0.9123142         MJSW    = 0.316175

+CJSWG   = 4.22E-10               PBSWG   = 0.9123142         MJSWG   = 0.316175

+CF      = 0                      PVTH0   = 8.456598E-4               PRDSW   = 8.4838247

+PK2     = 1.338191E-3   WKETA   = 0.0246885                  LKETA   = -2.016897E-3

+PU0     = -1.5089586    PUA     = -5.51646E-11   PUB     = 1E-21

+PVSAT   = 50            PETA0   = 1E-4           PKETA   = -3.316832E-3
```