

CPAL BASED DESIGN OF ARITHMETIC AND LOGIC CIRCUITS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

MASTER OF TECHNOLOGY

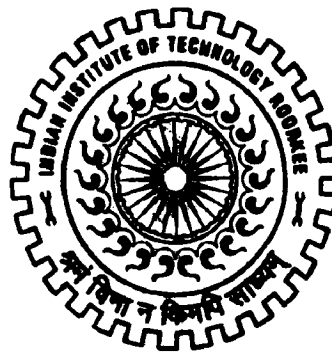
in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Semiconductor Devices and VLSI Technology)

By

A. RAJASEKHARA REDDY



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)

JUNE, 2008

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled “CPAL BASED DESIGN OF ARITHMETIC AND LOGIC CIRCUITS” in the partial fulfillment of the requirement for the award of the degree of **MASTER OF TECHNOLOGY** in Electronics and Communication Engineering with specialization in **SEMICONDUCTOR DEVICES AND VLSI TECHNOLOGY**, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, is an authentic record of my own work carried out during the period from July 2007 to June 2008, under supervision of **DR. S. DASGUPTA**, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee.

I have not submitted the matter, embodied in this dissertation report for the award of any other degree.

Date: 30/06/08

Place: Roorkee

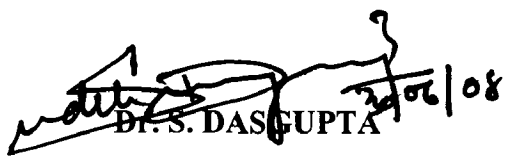

A. RAJASEKHARA REDDY

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief

Date :

Place: Roorkee


Dr. S. DASGUPTA
Assistant Professor,
Department of E& C
IIT Roorkee
Roorkee, Uttarakhand – 247 667

ACKNOWLEDGEMENT

All endeavors over a long period can be successful only with the support of many well-wishers; I take this opportunity to express my gratitude and appreciation to all of them.

It is my privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to my supervisor **Dr. S. Dasgupta**, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee, for his inspiration, guidance, constructive criticisms and encouragement during the entire course of this work. His cooperation and in depth knowledge have made my work possible.

I would like to thank Research scholars S.Vishvakarma, Balwinder Raj and V. Ramesh in SDVT group for their constant help and constructive criticisms.

Thanks to the Lab staff of SDVT Lab, Department of Electronics and Computer Engineering, IIT Roorkee for providing necessary facilities.

I am greatly indebted to my parents, who have graciously applied themselves to the task of helping me with ample moral support and valuable suggestions. Finally, I would like to extend my gratitude to my friends and all others who directly or indirectly helped me in the process and contributed towards this work.

(A. RAJASEKHARA REDDY)

ABSTRACT

As the density and operating speed of CMOS chips increase, power dissipation has become a critical concern in the design of VLSI circuits, especially in mobile and portable electronic systems. In conventional CMOS circuits popular approaches to low-power design include the reduction of supply voltage, node capacitance and switching activity. Adiabatic logic is a promising alternative low-power design technique which is compatible with the energy savings that can be achieved through reductions in supply voltage or node capacitance, yet achieves additional reductions in dissipated energy by avoiding the single-rail DC power supply architecture. Adiabatic circuit's uses AC power supply to achieve low power consumption by maintaining small potential drops across conducting devices and by recycling the energy stored in output node capacitors during their operation. The low-power digital circuits can be designed using adiabatic logic. Many DSP functional units such as FIR filters and FFT modules perform extensive sequences of multiplying and accumulating computations. In these applications multipliers are an important dissipation sources because they have high switching activity and contain large node capacitances. The adiabatic multiplier circuits can achieve a low-power dissipation even in the presence of large load capacitance. An adiabatic 8-bit Brent-Kung adder and 4-bit multiplier were implemented using CPAL and CMOS logic at 130nm. The power consumption of these circuits was observed for different frequencies up to 500MHz. The SPICE simulation results show that CPAL is efficient technique in terms of power consumption which has ≈ 35 to 75% less than CMOS counter parts depending on operating frequency and area needed for the design using CPAL is ≈ 20 to 25% more than the CMOS logic design.

LIST OF FIGURES

Figure	Description	Page No.
3.1	Schematic of CPAL buffer.	10
3.2	CPAL buffer chain and its two-phase power clock.	10
3.3	Simulation waveform of CPAL Buffer	11
3.4	4-stage cascaded CPAL buffers	13
3.5	four-phase power clock.	13
4.1	Block diagram of generic CPL circuit.	14
4.2	two-input gates using CPAL	15
4.3	Four-input gates using CPAL	16
4.4	CMOS Full adder.	17
4.5	Full adder using CPAL	18
4.6	P,G signals generation block (dot gate)	23
4.7	8-bit Brent-Kung adder	24
4.8	Block diagram of multiplier	26
4.9	Generation of partial products	27
4.10	Block diagram of 4-2 compressor	28
4.11	4-2 compressor based on full adders	29
5.1	Average power consumption of two-input and four-input logic gates	32
5.2	Average Power consumption of full adder	33
5.3	Average power consumption of 4-2 compressor	34
5.4	Average power consumption of buffer chain	35
5.5	Average power consumption of 8-bit Brent-Kung adder	36
5.6	Average power consumption of 4-bit multiplier	37

LIST OF TABLES

Table	Description	Page no
1	Power consumption of 8-bit Brent-Kung adder and 4-bit multiplier at 130nm, for CPAL and CMOS	36
2	Transistor count for Arithmetic and Logiccircuits in CMOS and CPAL design	39

INDEX

CANDIDATE'S DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
LIST OF FIGURES	iv
LIST OF TABLES	vi
INDEX	vii
1. INTRODUCTION	1
1.1 Introduction	1
1.2 Organization of dissertation	3
2. ADIABATIC LOGIC	4
2.1 The need for true adiabaticity	4
2.2 Power dissipation	5
2.3 Common mistakes in adiabatic logic design to avoid	6
3. COMPLEMENTARY PASS-TRANSISTOR ADIABATIC LOGIC	9
3.1 Two-phase CPAL	9
3.2 Four-phase CPAL	12
4. ARITHMETIC AND LOGIC CIRCUITS USING CPAL	14
4.1 Two-input and four-input logic gates	14
4.2 Full-adder	17
4.3 Carry-lookahead adder	18
4.3.1 Logarithmic-lookahead adder	20
4.3.2 8-bit logarithmic lookahead adder based on Brent-Kung tree	22

4.4 Multiplier	25
4.4.1 Partial products generation stage	26
4.4.2 Partial products addition stage	27
4.4.2.1 4-2compressor	28
4.4.3 The final addition stage	29
5. RESULTS AND DISCUSSION	30
5.1 Two-input and four-input logic gates	30
5.2 Full adder and 4-2 compressor	33
5.3 Four-stage buffer chain	34
5.4 8-bit Brent-Kung adder and 4-bit multiplier	35
6. CONCLUSIONS AND FUTURE SCOPE	40
6.1 Conclusions	40
6.2 Future scope	40
REFERENCES	41
LIST OF PUBLICATIONS	43

1.1 INTRODUCTION

The importance of reducing power dissipation in digital systems is increasing as the range and sophistication of applications in portable and embedded computing continues to increase. System-level issues such as battery life, weight, and size are directly affected by power dissipation. The classical approaches to achieve low-power design are to reduce the supply voltage, the loading capacitances of gates, and the switching activity; however they have limits [1]. According to the formula:

$$P_{dyn} = V_{dd}^2 \cdot f_{clk} \cdot \sum_n \alpha_n \cdot c_n + V_{dd} \cdot \sum_n i_{scn}$$

the dynamic power dissipation of a digital CMOS circuit depends on the supply voltage V_{DD} , the clock frequency f_{clk} , the node switching activities α_n and the number of nodes n . A reduction of each of these parameters results in a reduction of dissipated power. However, clock frequency reduction is only feasible at the architecture level, whereas at the circuit level clock frequency f_{clk} is usually regarded as constant in order to fulfill some given throughput requirement. All the other parameters are influenced to some degree by the logic style applied.

1) *Load capacitance reduction*: Capacitive load, originating from transistor capacitances (gate and diffusion) and interconnect wiring, is to be minimized. This is achieved by having as few transistors and circuit nodes as possible, and by reducing transistor sizes to a minimum. In particular, the number of (high capacitive) inter-cell connections and their length (influenced by the circuit size) should be kept minimal. Transistor downsizing is an effective way to reduce switched capacitance of logic gates on noncritical signal paths [2]. For that purpose, a logic style should be robust against transistor downsizing, i.e., correct functioning of logic gates with minimal or near-minimal transistor sizes must be guaranteed (*ratio less* logic).

2) *Supply voltage reduction*: The supply voltage and the choice of logic style are indirectly related through delay-driven voltage scaling. That is, a logic style providing fast logic gates to speed up critical signal paths allows a reduction of the supply voltage in order to achieve a given throughput. For that purpose, a logic style must be robust against supply voltage reduction, i.e., performance and correct functioning of gates must be guaranteed at low voltages as well. This

becomes a severe problem at very low voltages of around 1 V and lower, where noise margins become critical [3,4].

3) *Switching activity reduction*: Switching activity of a circuit is predominantly controlled at the architectural and register transfer level (RTL). At the circuit level, large differences are primarily observed between static and dynamic logic styles. On the other hand, only minor transition activity variations are observed among different static logic styles and among logic gates of different complexity, also if glitching is concerned.

4) *Short-circuit current reduction*: Short-circuit currents (also called dynamic leakage currents or overlap currents) may vary by a considerable amount between different logic styles. They also strongly depend on input signal slopes (i.e., steep and balanced signal slopes are better) and thus on transistor sizing. Their contribution to the overall power consumption is rather limited but still not negligible, except for very low voltages $V_{DD} \leq V_{TN} + V_{TP}$ where the short-circuit currents disappear. A low-power logic style should have minimal short-circuit currents and, of course, no static currents besides the inherent CMOS leakage currents.

Low-power circuit design methodologies inspired by thermodynamics have recently attracted considerable attention. The circuits designed with these methodologies are generically referred to as adiabatic circuits. Adiabatic circuits strive to achieve low power consumption by maintaining small potential drops across conducting devices and by recycling the energy stored in output node capacitors during their operation. Energy flows into and out of an adiabatic circuit by means of controlled, slowly-changing waveforms.

The current adiabatic circuits can be classified into two types

- Full-adiabatic logic
- Quasi-adiabatic logic.

The full-adiabatic circuit, which can be implemented by using reversible logic, is much more complex than the quasi-adiabatic circuit. The quasi-adiabatic logic circuit has simple architecture and power clock system. Moreover, the quasi-adiabatic circuit can operate at a higher frequency. Therefore, the quasi-adiabatic circuit is a promising scheme for practical applications [5].

Several quasi-adiabatic logic architectures have been reported, such as, 2N-2N2P, ECRL, DTGAL (using four-phase power-clocks), etc [6,7]. Although they consume lower power than the conventional CMOS, the charge of output loads can't be completely recovered, and their

energy dissipation is highly dependent on the output load capacitance. Also transistor count for these adiabatic logics is much larger than the conventional CMOS logic, which is again the other concern of area overhead. These adiabatic logics needs different phase power-clocks to energy recovery, but generation of efficient power-clocks with different phase is the major concern.

The present Complementary Pass-transistor Adiabatic Logic (CPAL) using two-phase and four-phase power-clocks has more efficient energy transfer and recovery, because the non-adiabatic energy loss of output loads has been completely eliminated by using complementary pass-transistor logic for evaluation and transmission gates for energy-recovery. For any adiabatic logic family we have to use multi-phase power clocks. As the phase number increases the complexity of power clock generation module increase. But this CPAL uses two-phase and four-phase power clocks there by reducing the complexity in the power clock generation. The two-phase CPAL is more suitable for design of flip-flops and sequential circuits, also CPAL use fewer transistors than conventional CMOS transmission gate-based implementations and other adiabatic logic circuits such as 2N-2N2P and PAL-2N [7,8]. The Digital circuits such as adders and multipliers are the some of most power-consuming blocks in a microprocessor. Four- phase CPAL is more suitable for design of these circuits.

8-bit Brent-Kung adder and 4-bit multiplier circuits which have less power consumption even in the presence of large capacitance on the output nodes than the conventional CMOS counter ones are implemented using CPAL at 130nm technology. The Power consumption of these circuits is compared for different frequencies up to 500MHz.

1.2 ORGANIZATION OF DISSERTATION

This work is opened with an introduction about the classical approaches to reduce power consumption in CMOS logic circuits and types of adiabatic circuits in chapter 1. In chapter 2, need for adiabaticity, power consumption in adiabatic logics and common mistakes in adiabatic logic are discussed. In chapter 3, CPAL which is one of the adiabatic techniques is discussed. In chapter 4, Arithmetic and Logic circuits (full-adder, 4-2 compressor, 8-bit Brent-Kung adder, 4-bit multiplier, buffer chain etc.) are implemented using CPAL and CMOS at 130nm technology node. The power consumption of these circuits using CPAL and CMOS logic is compared by varying operating frequency and results are presented in chapter 5. The conclusion is given in chapter 6 along with the future scope of work.

2.1 THE NEED FOR TRUE ADIABATICITY

First, simple economic arguments show that over the long run, as manufacturing process efficiency improves, and the cost of raw hardware resources (*e.g.*, gate-cycles) decreases, the cost of energy dissipated must eventually become the dominant part of the total cost of any computation. Even today, energy transport systems (power supplies, packaging, fans, enclosures, air-conditioning systems) comprise a significant fraction of the manufacturing and installation cost in many computing applications. However, an even more dominant consideration is that the practical limits on cooling-system capacity (in terms of the total Watts of power that may be dissipated harmlessly in a system of given size) imply that practical hardware efficiency in any limited-size system is itself immediately impacted by the energy efficiency of the system's components. Moreover, the cooling problem for a given logic technology is not one that can be solved by mere engineering cleverness in one's cooling system design, as there exist absolutely fundamental and unavoidable quantum-mechanical limits on the rate at which entropy can be exported from a system of given size by a coolant flow of given power [9].

Still, engineering cleverness in the logic, via truly adiabatic design, can enable us to avoid the energy efficiency limits suffered by traditional irreversible technology, allowing us to continue improving hardware efficiency for cooling-limited applications by many orders of magnitude beyond the limits that would apply if non-adiabatic or even non-truly adiabatic techniques (such as most "adiabatic" techniques in the literature) were used. The *degree of adiabaticity* of any process can be defined as equal to its quality factor Q , in the sense used in electrical and mechanical engineering, *i.e.*, the ratio between the amount of free energy involved in carrying out the process, and the amount of this energy that gets dissipated to heat. In contrast, the other dissipation mechanisms typically require non circuit-level solutions such as (1) electromagnetic shielding, (2) high threshold devices and/or low temperature devices, (3) thicker, high-k gate dielectrics. We should emphasize that both general areas must be addressed in the long run: that is, not only the intentional sources of dissipation (*e.g.*, $\frac{1}{2}CV^2$ switching energy of

irreversible transitions), which can be prevented through adiabatic circuit design methodologies, but also the parasitic sources of dissipation, which must be addressed through engineering device physics and package-level shielding/cooling. Both intentional and parasitic dissipation must eventually be addressed to meet the fundamental long-term requirement for maximally energy-efficient computation. The efficiency benefits that be gained by working at the circuit level alone are limited, but we can foresee that in the long run, further improvements can and will be made in all of these areas, so that an unlimited degree of adiabaticity in the circuit design will be beneficial. Adiabaticity will be an essential element of our logic design methodology if it is to retain long-term relevance. [10].

2.2 POWER DISSIPATION

The minimum power dissipation of CV_{dd}^2 in ordinary switching circuits is primarily due to the fact that such circuits charge a node by connecting it to a constant voltage power supply. Charges can be distinguished as either controlling or controlled charge. For MOSFET's, the controlling charge is on the gate, while the controlled charge flows through the channel. Dissipation is caused by the resistance encountered by the controlled charge. If charge transport is slowed down, energy that would otherwise be dissipated in the channel can be conserved for later reuse. The energy advantage can be readily understood by assuming a constant current source that delivers the charge $C_L V_{dd}$ over a time period T . The dissipation through the channel resistance R is then:

$$\begin{aligned} E_{diss} &= P \cdot T = I^2 \cdot R \cdot T = \left(\frac{C_L V_{DD}}{T} \right)^2 \cdot R \cdot T \\ &= \left(\frac{RC_L}{T} \right) \cdot C_L V_{DD}^2 \end{aligned} \quad (1)$$

Above relations shows that it is possible to charge and discharge a capacitance through a resistance while dissipating less than $C_L V_{dd}^2$ of energy. It also suggests that it is possible to reduce the dissipation to an arbitrary degree by increasing the switching time to ever-larger values. We refer to this as the principle of adiabatic charging. We use the term "adiabatic" to indicate that all charge transfer is to occur without generating heat. As is the typical usage of the term in thermo dynamics, fully adiabatic operation is an ideal condition that is asymptotically approached as the process is slowed down. Switching circuits that charge and discharge their

load capacitance adiabatically are said to use adiabatic switching. The circuits rely on special power supplies that provide accurate pulsed-power delivery. It is important to note that adiabatic switching techniques can be an attractive alternative to other low-power design approaches only if the supplies can deliver power efficiently and recycle the power fed back to them.

2.3 COMMON MISTAKES IN ADIABATIC LOGIC DESIGN TO AVOID

Most so-called “adiabatic” digital logic circuit families reported in the low-power design literature are actually not truly adiabatic, in that they do not satisfy the general definition of adiabatic physical processes, as ones whose energy dissipation tends towards zero as their speed and/or parasitic interactions are decreased. The most common departures from true adiabaticity in the logic designs that have been published to date, and discuss how these problems can be avoided in the future. The most common problems are: (1) use of diodes, (2) turning off transistors when there is nonzero current across them, (3) failure of the design style to accommodate arbitrarily much logical reversibility, which can be proven to be required to approach truly adiabatic operation, and (4) failure to accommodate the asymptotically most cost-efficient possible circuit algorithms, in terms of both hardware time and energy. [10]

Don't use diodes:

The first and simplest rule of true adiabatic design is: *never use diodes*. At the very least, one should always recognize that whenever one includes a diode as a necessary functional element in part of one's circuit (in contrast to, for example, junction diodes that are used only for device isolation or ESD protection), then that part of the design has no long-term viability and will eventually have to be replaced, as the requirements for energy efficiency become ever more stringent. The reason is that diodes, in their role as a one-way valve for current, are fundamentally thermodynamically irreversible, and cannot operate without a certain irreducible entropy generation. For example, ordinary semiconductor diodes have a built-in voltage drop across them, and this “diode drop” results in an irreversible energy dissipation of QV for an amount of charge Q carried through it. To the extent that the diode drop is less than logic voltage swings, so that the diode losses are much less than non adiabatic $\frac{1}{2}CV^2$ losses, this approach may still be useful in the short run, but it must eventually be abandoned when we need still greater energy efficiency. [10]

Don't disobey transistor rules:

Although diodes are fundamentally non-adiabatic, fortunately, transistors, despite being non-ideal switches, remain acceptable for adiabatic operation, so long as two basic rules are followed:

- (1) Never turn on a transistor when there is a significant (non-negligible) voltage difference between its source and drain terminals.
- (2) Never turn off a transistor when there is significant current flowing through its channel.

The first rule is fairly obvious, because, for example, when a dynamic node of capacitance C is directly connected to a static reference signal of voltage different from it by V , we all know there is an unavoidable dissipation of $\frac{1}{2}CV^2$ in the node's transition to its new level. Even in the best case, where both nodes are isolated and both of capacitance C , the dissipation as they converge to their average level is still $\frac{1}{4}CV^2$. In the worst case, when the two nodes are connected to differing voltage sources, turning on the transistor results in a continuous power dissipation there after. Nearly all adiabatic logic styles obey this rule, at least approximately—in light of noise considerations, leakage, *etc.*, it will in general be impossible to ensure that voltages exactly match before the transistor is turned on. But we should try to get as close as possible to a match. [10]

The second rule is less obvious, and most of the purportedly adiabatic logic styles in fact fail to follow it. Why does it necessarily cause dissipation to shut off a flow of current by turning off a transistor that it is flowing through? The reason comes from the fact that real transistors are not perfect switches, which go instantaneously from perfect-on to perfect-off the moment the gate-to source voltage crosses some threshold (however slowly). In fact, such ideal switches can be shown thermodynamically impossible, because they can be used to build lossless diodes [11].

As we all know from looking at I - V curves, real transistors turn off only gradually. This is especially so when the gate voltage itself is changing only gradually over time, which is the case when the gate voltage is being controlled adiabatically, as will be the case for most of the transistors in any mostly-adiabatic digital circuit. Because of this, during part of any on/off transition, the transistor will have an intermediate level of effective resistance— not the ~ 10 k Ω of a minimum sized saturated MOSFET, nor the many giga ohms or more of a low leakage, turned-off device, but some intermediate level, perhaps in the mega ohms, at which the voltage drop across the device increases substantially, but the resistance is not yet so high as to bring the

$P = V^2/R$ power dissipation back down towards zero. This can lead to a significant non-adiabatic dissipation that does not scale down very much as the overall frequency is decreased. [16].

COMPLEMENTARY PASS-TRANSISTOR ADIABATIC LOGIC (CPAL)

CPAL circuits have more efficient energy transfer and recovery, because the non-adiabatic energy loss of output loads has been completely eliminated by using complementary pass-transistor logic for evaluation and transmission gates for energy-recovery. Reversible-logic techniques may be used to avoid storage elements and thus make it possible to build fully adiabatic systems.

3.1 TWO-PHASE CPAL

The two-phase CPAL is more suitable for design of flip-flops and sequential circuits [12], and they use fewer transistors than conventional CMOS transmission gate-based implementations and other adiabatic logic circuits such as 2N-2N2P and PAL-2N [7,8]. The basic structure of the CPAL buffer (inverter) is shown in Figure.3.1. It is composed of two main parts:

- Logic function circuit
- Load driven circuit.

The logic circuit consists of four NMOS transistors (N_1 - N_4) with complementary pass-transistor logic (CPL) that was proposed by K. Yano [13]. The load driven circuit consists of a pair of cross-coupled CMOS transmission gates (N_5, P_1 and N_6, P_2). The CPAL gate is supplied by a single phase power-clock. Cascaded CPAL gates are driven by two-phase power-clocks, as shown in Figure.3.2. The simulation waveforms of the CPAL buffer are illustrated in Figure.3.3. By referring the schematic shown in Figure.3.2 and the waveforms in Figure.3.3, the operation of the two-phase CPAL buffer can be summarized as follows.

During the time interval T_1 and T_2 , the voltage of the input IN_b is low and the voltage of the input IN goes high. Therefore, N_1 and N_3 are turned on. As the voltage of the input IN goes up, the voltage of the node X is charged to about $V_{DD} - V_{TN}$, where V_{TN} is the threshold voltage of the NMOS transistor, while the node Y is clamped to ground. During the time interval T_3 and T_4 , as the voltage of the input IN falls, N_1 and N_2 are turned off. Thus, the voltage of the node X will keep its state because the node X is isolated. During the period T_5 , as the voltage of the clock CLK goes up, the voltage of the node OUT begins to go high via N_5 . When the voltage of

the OUT rises above V_{TN} , N_8 will be turned on, and $OUTb$ will be clamped to ground. When the voltage of the clock CLK rises above V_{TP} , where V_{TP} is the threshold voltage of the PMOS transistor, P_1 will be turned on, so the node OUT is charged through transmission gate (N_5, P_1) without non-adiabatic loss.

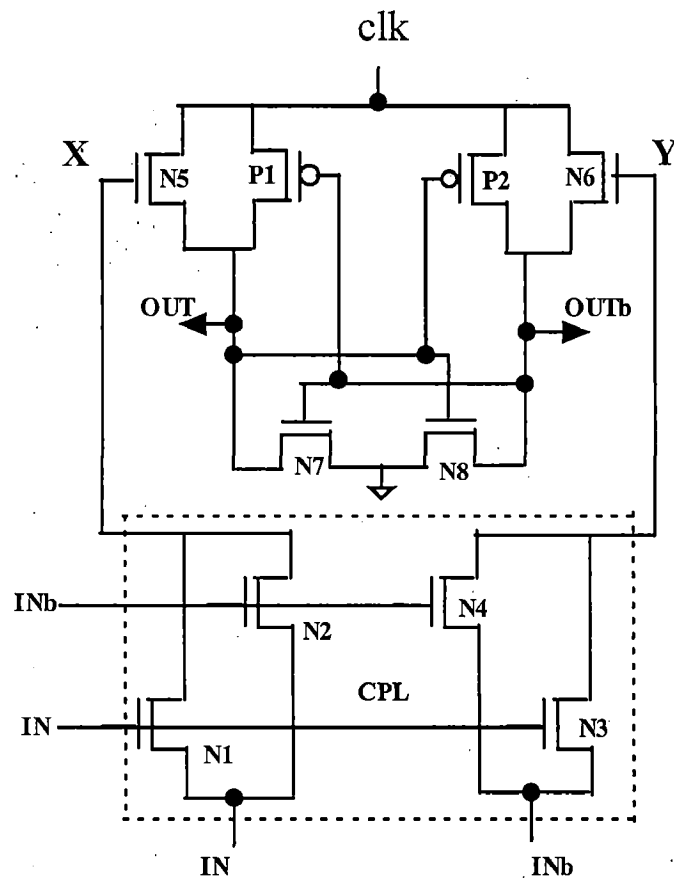


Figure 3.1. Schematic of CPAL buffer.[12]

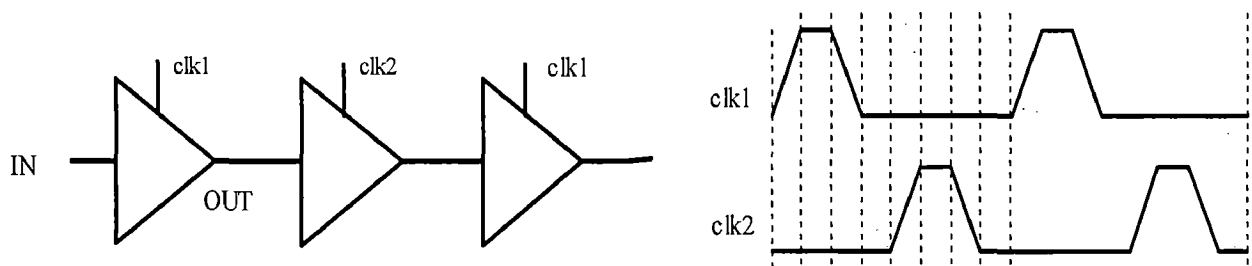


Figure 3.2. CPAL buffer chain and its two-phase power clock.[12]

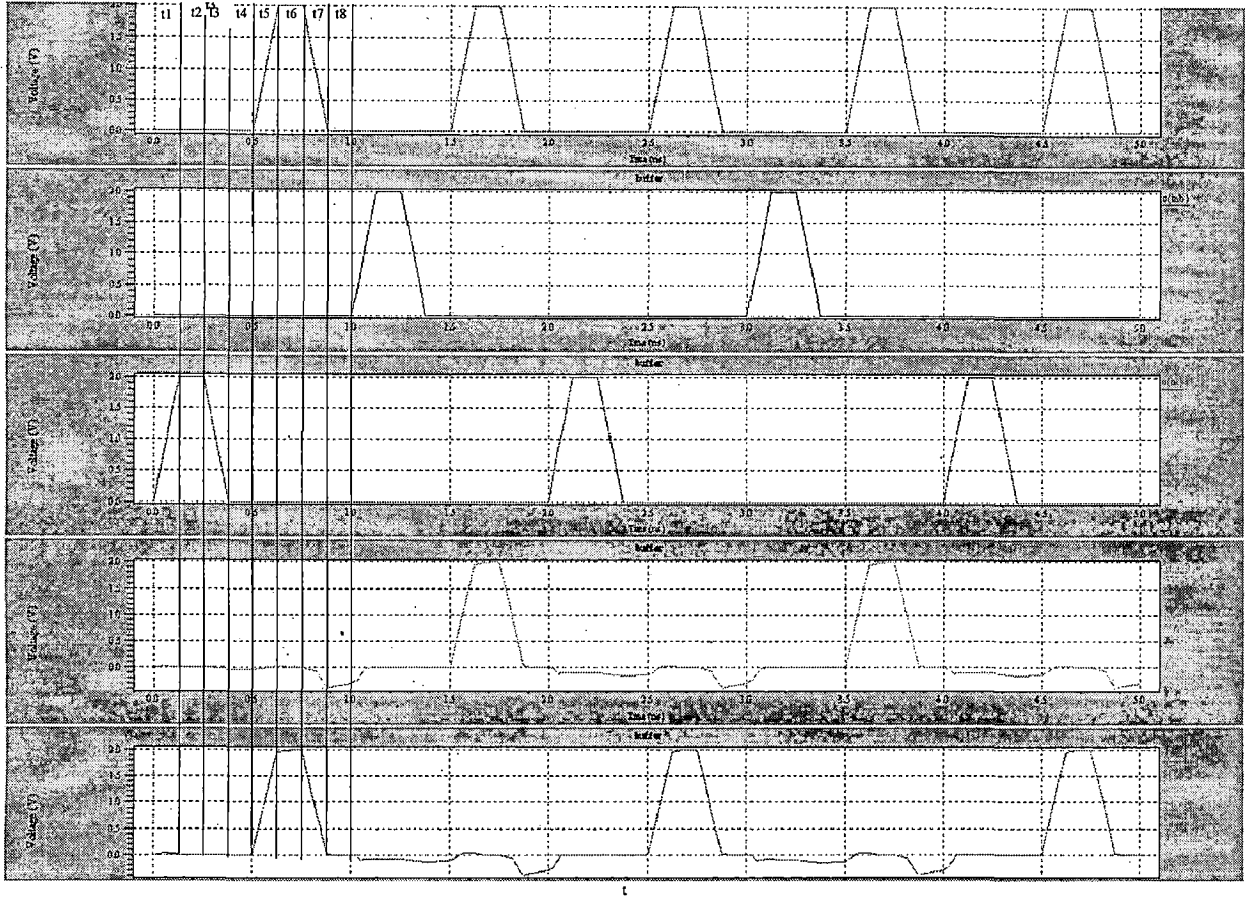


Figure 3.3. Simulation waveform of CPAL Buffer

During T_6 , the voltage of the node OUT is the same as the clock CLK, while the node OUTb is still at ground. During T_7 , as the voltage of the clock falls from V_{DD} to ground, the charge on the node OUT is recovered through N_5 and P_1 . During the time interval $T_5 - T_7$, because N_1 and N_2 are turned off, the node X is in the high-impedance state. Therefore, the voltage of the node X can be bootstrapped to a higher level than $V_{DD} - V_{TN}$ due to the gate-to-channel capacitance of N_5 , [14]. When CLK rises from 0V to V_{DD} , the voltage of the node X is increased by ΔV , which is expressed as

$$\Delta V = \frac{C_G}{C_{D1} + C_{D2} + C_W + C_G} V_{DD} \quad (2)$$

where $C_G = WLC_{OX}$ is the gate-to-channel capacitance of N_5 (or N_6), W and L are the channel width and length of N_5 (or N_6), C_{OX} is the gate-to-channel capacitance per unit area, C_{D1} and C_{D2} is the diffusion capacitance of N_1 and N_2 , respectively, and C_W represents the wiring capacitance.

According to (2), when the channel width of N_5 and N_6 increases, the ΔV will be raised. High voltage ΔV can reduce the adiabatic loss because the turn-on resistances of the NMOS transistors (N_5 and N_6) are reduced [15].

Energy dissipation of the two-phase CPAL circuits includes mainly two terms:

- Full-adiabatic energy loss on output nodes
- Non-adiabatic energy loss on internal nodes,

The energy dissipation per cycle of the internal nodes X (or Y) can be written as

$$E_X = C_X(V_{DD} - V_{TN})V_{TN} + \frac{1}{2}C_X(V_{DD} - V_{TN})^2 \quad (3)$$

where $C_X = C_{D1} + C_{D2} + C_{G7} + WLC_{OX}$ is the capacitance of the node X, and W and L are the channel width and length of N_5 (or N_6). Full-adiabatic energy loss on output nodes can be represented as

$$E_{output} = 2 \left(\frac{RC_L}{T} \right) C_L V_{DD}^2 \quad (4)$$

where C_L is the load capacitance of the CPAL buffer, T is the transition time of the power-clock, and R is the turn-on resistance of the transmission gate. Compared to 2N-2N2P, CPAL dissipates less energy and is insensitive to load capacitance, because the capacitance of the internal nodes is much smaller than that of output nodes. The energy loss of CPAL is much lower than 2N-2N2P at all operation frequencies, because the turn-on resistance of the transmission gates of CPAL is smaller than that of the PMOS transistors of 2N-2N2P. [16]

3.2 FOUR-PHASE CPAL

The four-phase CPAL is more suitable for design combinational circuits, and they use fewer transistors than conventional CMOS transmission gate-based implementations and other adiabatic logic circuits such as 2N-2N2P and PAL-2N [7,8]. The basic structure of the CPAL buffer (inverter) is same as two-phase CPAL. Cascaded CPAL gates are driven by four-phase power-clocks, as shown in Figure3.4. Four-phase clock is shown in Figure3.5

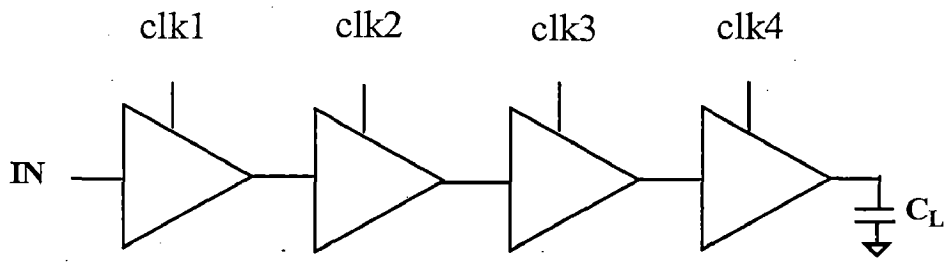


Figure.3.4 4-stage cascaded CPAL buffers.[19]

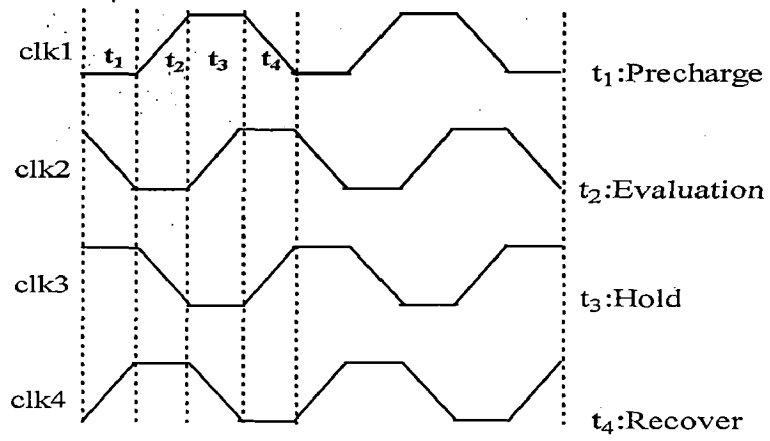


Figure.3.5 four-phase power clock.[19]

ARITHMETIC AND LOGIC CIRCUITS USING CPAL

4.1 TWO-INPUT AND FOUR-INPUT LOGIC GATES

Complementary Pass-transistor Logic (CPL) network employs input signals at both gate and drain terminals. The block diagram of generic CPL circuit is shown in Figure.4.1. Inputs and Outputs are always complementary. Outputs from network provide strong '0's but weak '1's. Load driven circuit provide amplification and buffering as necessary. The CPAL gates can be realized by using complementary pass-transistor logic (CPL) to replace the transistors (N1-N4) of the CPAL buffer. Figure.4.2 shows the two-input AND/NAND gate, OR/NOR gate, XOR/XNOR gate and multiplexer with CPAL circuit topology. Figure.4.3 shows the four-input AND/NAND gate, OR/NOR gate. In Figure.4.2 and Figure.4.3 only the N-logic input blocks are shown and the other transistors (P1, P2, and N5-N8) are omitted for simplicity. All basic gates, such as inverter, AND, OR, multiplexer, and XOR, use the same topology, and only inputs are permuted.

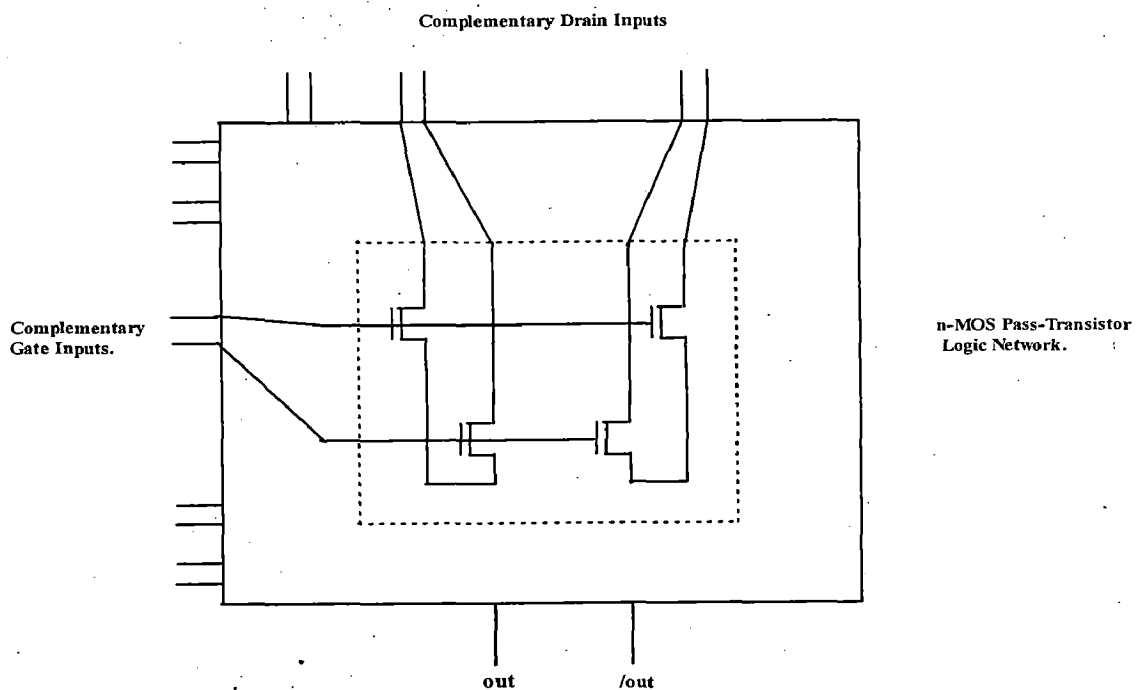


Figure.4.1 Block diagram of generic CPL circuit.

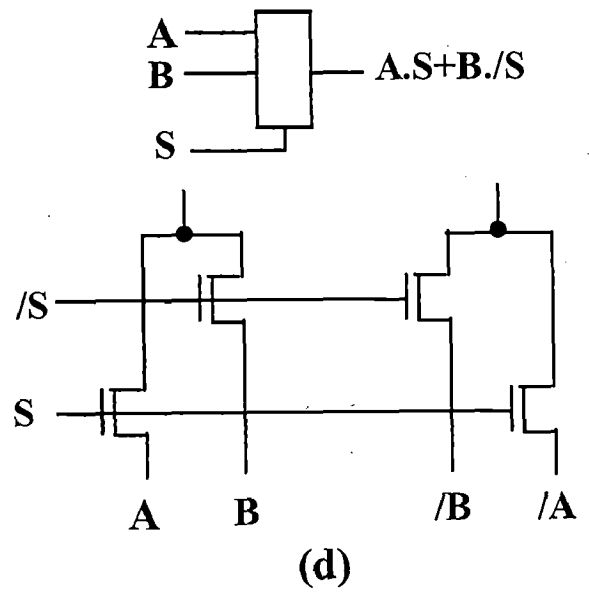
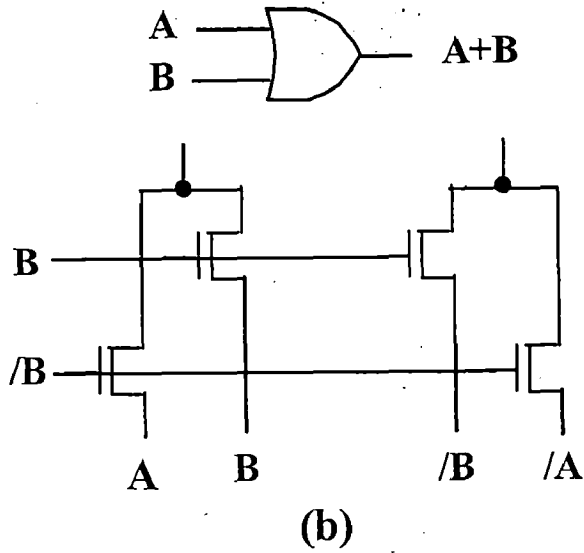
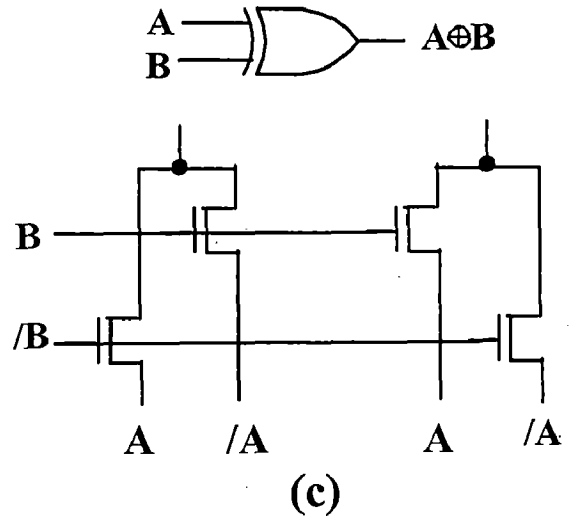
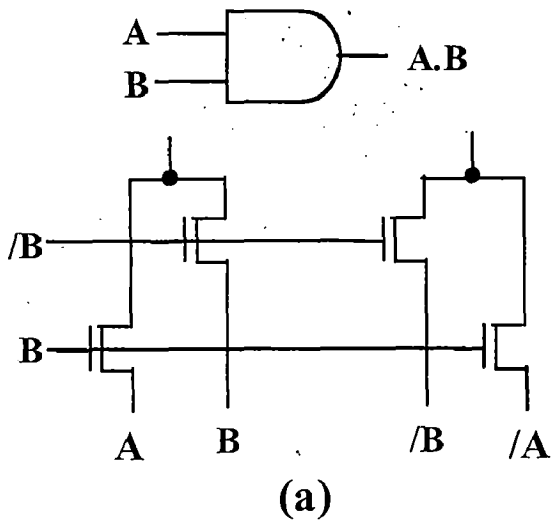
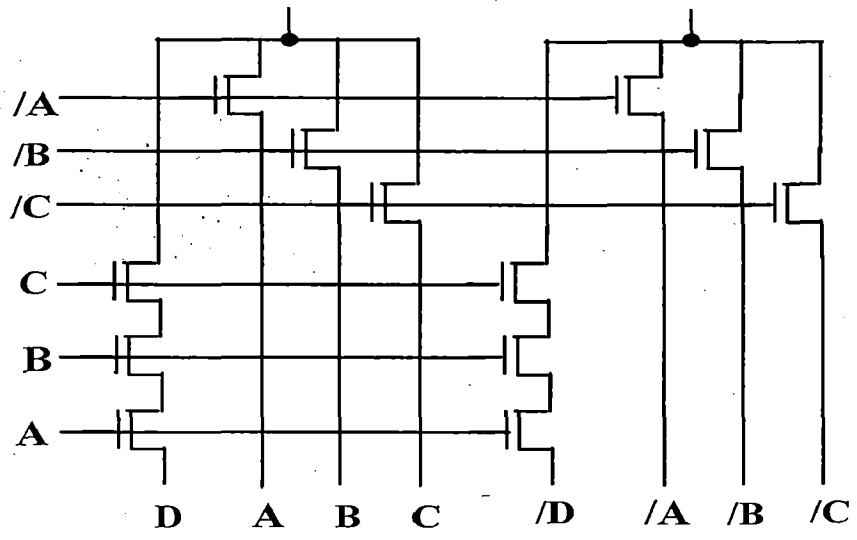
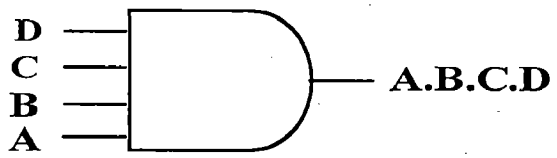
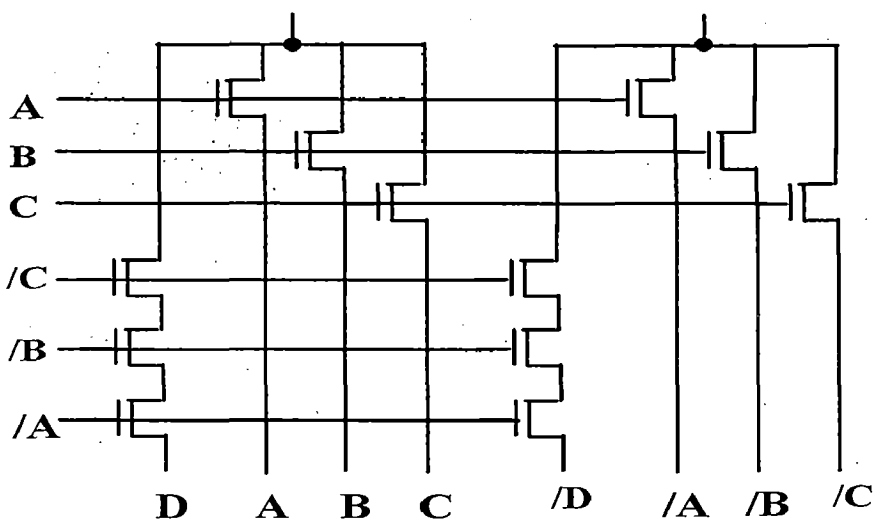
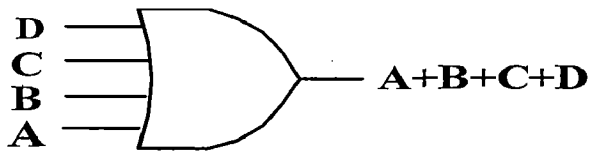


Figure.4.2 Two-input gates using CPAL.[16]



(a)



(b)

Figure.4.3 Four-input gates using CPAL

4.2 FULL ADDER

An adder is an essential element in a digital system, so the design of the adder is chosen to show the usefulness of the CPAL. The binary full adder is a three-input, two-output combinational circuit. The `sum_out` and `carry_out` signals can be found as the following two combinational Boolean functions of the three input variables, A, B and C.

$$\begin{aligned} \text{sum}_{\text{out}} &= A \oplus B \oplus C \\ &= ABC + \overline{A}BC + A\overline{B}C + A\overline{B}\overline{C} \end{aligned}$$

$$\text{carry}_{\text{out}} = AB + AC + BC$$

The gate level realization of these functions is shown in Figure.4.4 [17]. The CPAL full adder can be realized by using complementary pass-transistor logic (CPL) as shown in Figure.4.5 to replace the transistors (N1-N4) of the CPAL buffer.

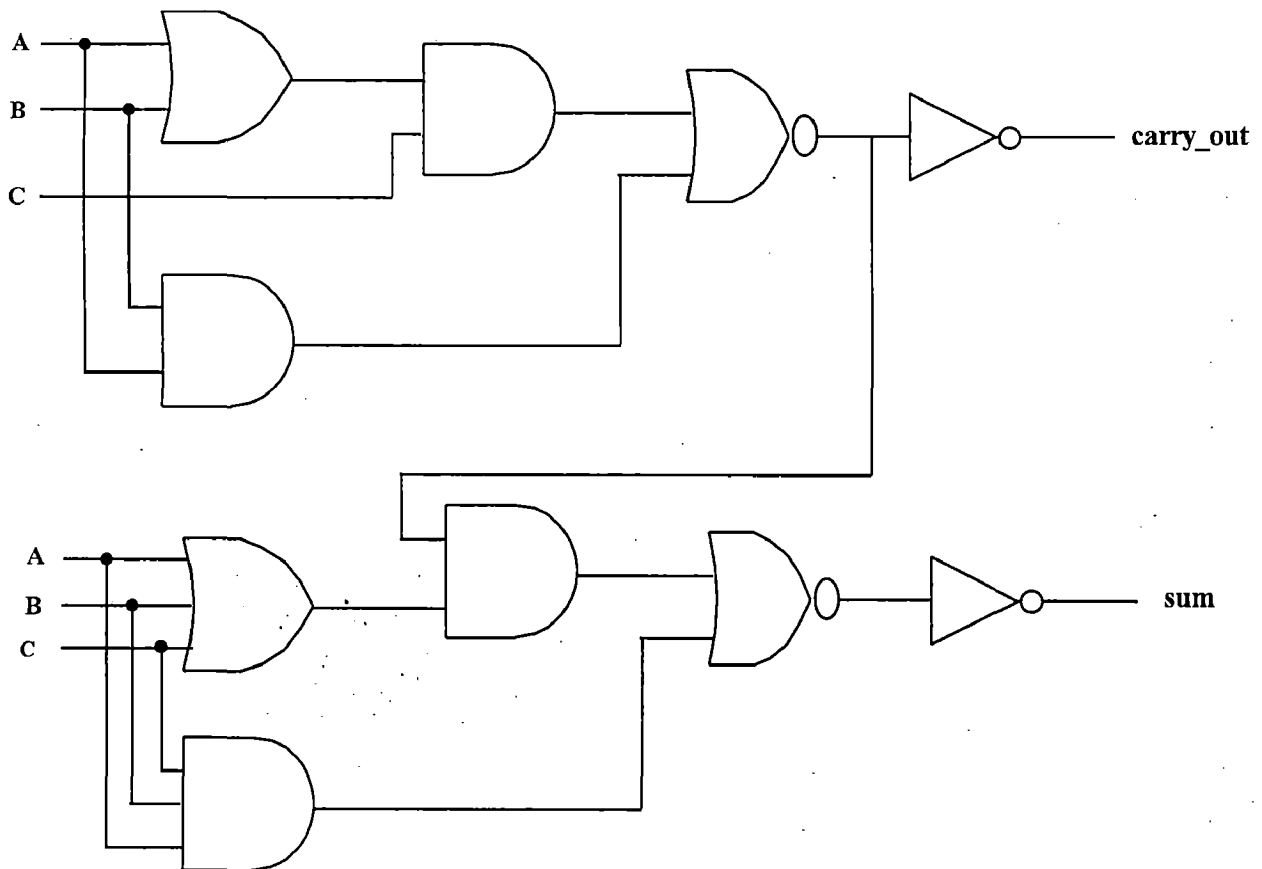


Figure.4.4 CMOS Full adder.[17]

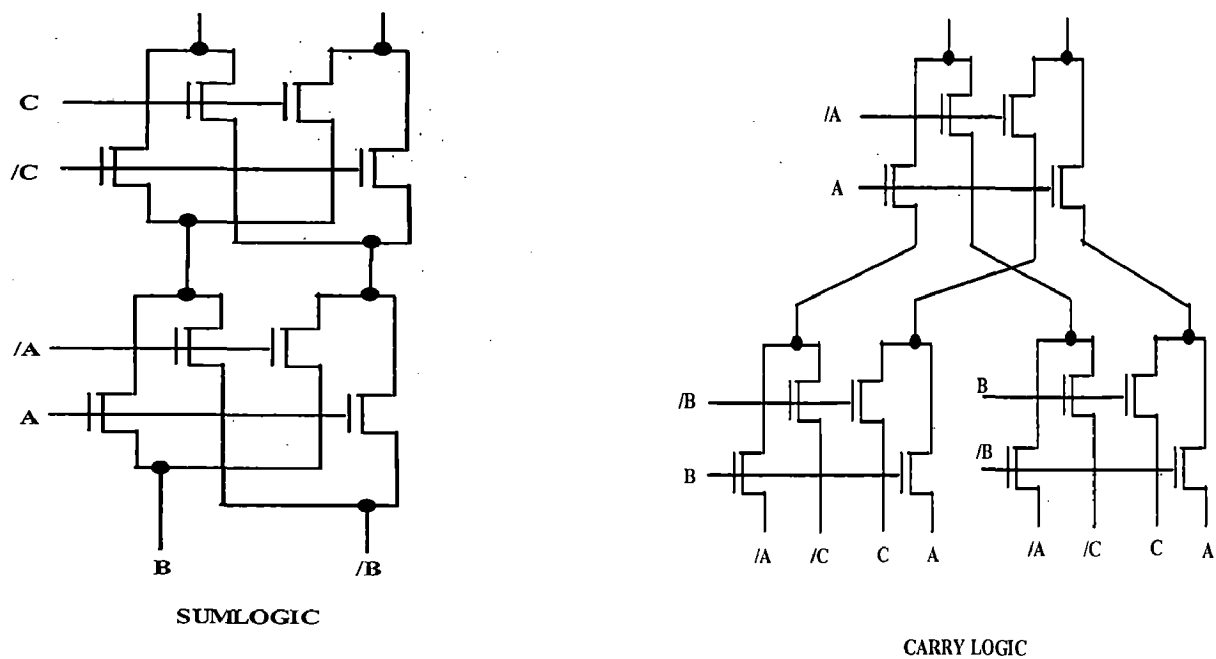


Figure.4.5 Full adder using CPAL. [19]

4.3 CARRY-LOOKAHEAD ADDER

Adiabatic logic computes only one logic level per phase, so it is difficult to implement a complex logic with long chain of stages. In this respect, a Ripple-Carry Adder (RCA) is not practical with the adiabatic method, since it needs many phases to obtain result. 32-bit RCA would need at least 32-phase to execute one addition. Also when designing fast adders, it is essential to get around the rippling effect of the carry that is still present in one form or another in both the carry-bypass and carry-select adders. The carry-lookahead principle offers a possible way to do so [18]. A CLA (Carry-Lookahead Adder) not only reduces logic depth, but also offers a pipelined structure if buffers are added in the circuit. So CLA structure is adopted in the design of the CPAL adder to overcome the drawback of an adiabatic circuit and to design fast adders. Carry-lookahead adder depends on two things:

1. Calculating, for each digit position, whether that position is going to propagate a carry if one comes in from the right.

2. Combining these calculated values so as to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

Carry-lookahead logic uses the concepts of *generating* and *propagating* carries. For each bit in a binary sequence to be added, the carry-lookahead logic will determine whether that bit pair will generate a carry or propagate a carry. This allows the circuit to "pre-process" the two numbers being added to determine the carry ahead of time. Then, when the actual addition is performed, there is no delay from waiting for the ripple carry effect (or time it takes for the carry from the first Full Adder to be passed down to the last Full Adder). The addition of two 1-digit inputs A and B is said to *generate* if the addition will always carry, regardless of whether there is an input carry or not. In the case of binary addition, $A + B$ generates if and only if both A and B are 1. If we write $G(A,B)$ to represent the binary predicate that is true if and only if $A + B$ generates, we have:

$$G(A,B)=A.B$$

The addition of two 1-digit inputs A and B is said to *propagate* if the addition will carry whenever there is an input carry. Note that propagate and generate are defined with respect to a single digit of addition and do not depend on any other digits in the sum. In the case of binary addition, $A + B$ propagates if and only if at least one of A or B is 1. If we write $P(A,B)$ to represent the binary predicate that is true if and only if $A + B$ propagates, we have:

$$P(A,B)=A+B$$

Sometimes a slightly different definition of *propagate* is used. By this definition $A + B$ is said to propagate if the addition will carry whenever there is an input carry, but will not carry if there is no input carry. It turns out that the way in which generate and propagate bits are used by the carry lookahead logic, it doesn't matter which definition is used. In the case of binary addition, this definition is expressed by:

$$P'(A,B)=A\oplus B$$

for a multiple-level carry-lookahead adder, it is simpler to use $P'(A,B)=A\oplus B$. Carry relations for 4-bit CLA:

$$C_j=G_0+P_0.C_0$$

$$C_2 = G_1 + P_1.C_1$$

$$C_3 = G_2 + P_2.C_2$$

$$C_4 = G_3 + P_3.C_3$$

Substituting C_1 into C_2 , then C_2 into C_3 , then C_3 into C_4 yields the expanded equations:

$$C_1 = G_0 + P_0.C_0$$

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

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

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

These expanded relations can be used to implement a 4-bit adder. For every bit the carry and sum outputs are independent of the previous bits. The rippling effect has thus been effectively eliminated, and addition time should be independent of the number of bits. Such a high level module contains some hidden dependencies. When we study the detailed schematics of the adder it becomes obvious that the constant addition time is wishful thinking and that the real delay is at least increasing linearly with the number of bits. Implementing it with simpler gates requires multiple logic levels due to this propagation delay increases. Further more, the fan-out on some of signals tend to grow excessively, slowing down the adder even more. Finally, the area of the implementation grows progressively with no of bits and is only useful for ≤ 4 -bits.

4.3.1 Logarithmic-lookahead adder

In order to built very fast adders it is necessary to organize carry propagation and generation into recursive trees. A more effective implementation is obtained by hierarchically decomposing the carry propagation into subgroups of N bits.

$$C_{0,0} = G_0 + P_0.C_{i,0}$$

$$C_{0,1} = G_1 + P_1.G_0 + P_1.P_0.C_{i,0} = (G_1 + P_1.G_0) + (P_1.P_0).C_{i,0} = G_{1:0} + P_{1:0}.C_{i,0}$$

$$C_{0,2} = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_{i,0} = G_2 + P_2.C_{0,1}$$

$$C_{0,3} = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.C_{i,0}$$

$$= (G_3 + P_3.G_2) + (P_3.P_2).C_{0,1} = G_{3:2} + P_{3:2}.C_{0,1}$$

In the above equations the carry-propagation is decomposed into subgroups of two bits. G_{ij} and P_{ij} denote the generate and propagate functions, respectively, for a group of bits (from bit position i to j). Therefore, we call them block generate propagate signals. G_{ij} equals 1 if the group generates a carry, independent of incoming carry. The block propagate P_{ij} is true if an incoming carry propagates through the complete group. For example, $G_{3:2}$ is equal to 1 when a carry either generated at position 3 or is generated at position 2 and propagated through position 3, or $G_{3:2}=G_3+P_3G_2$. $P_{3:2}$ is true when an incoming carry propagates through both bit positions, and $P_{3:2}=P_3P_2$. Note that the format of the new expression for the carry is equivalent to the original one, except that the generate and propagate signal are replaced with block generate and propagate signals. The notation G_{ij} and P_{ij} generalizes the original carry equations, since $G_i=G_{i:i}$ and $P_i=P_{i:i}$. Another generalization is possible by treating the generate and propagate functions as a pair (G_{ij}, P_{ij}) , rather than considering them as separate functions. A new Boolean operator called the *dot* (\cdot), can be introduced. This operator on the pairs and allows for the combination and manipulation of blocks of bits:

$$(G,P) \cdot (G',P') = (G + PG', PP')$$

Using this operator we can now decompose $(G_{3:2}, P_{3:2}) = (G_3, P_3) \cdot (G_2, P_2)$. The dot operator obeys the associative property, but it is not commutative. By exploring the associative property of the dot operator, a tree can be constructed that effectively computes the carries at all $2^i - 1$ positions (that is, 1, 3, 7, 15 etc.) for $i=1 \dots \log_2(N)$. The crucial advantage is that the computation of the carry at position $2^i - 1$ takes only $\log_2(N)$ steps. In other words, the output carry of an N -bit adder can be computed in $\log_2(N)$ time. This is a major improvement over the previously described adders. For example, for an adder of 64 bits, the propagation delay of a linear adder is proportional to 63. For a square-root select adder, it is reduced to 8, while, for a logarithmic adder, the proportionality constant is 6. The carry at position 15 is computed by combining the results of blocks (0:7) and (8:15). Each of these, in turn, is composed hierarchically. For instance, (0:7) is the composition of (0:3) and (4:7), while (0:3) consists of (0:1) and (2:3), etc. [1].

Computing the carries at just $2^i - 1$ position is obviously not sufficient. It is necessary to derive the carry signal at the intermediate positions is obviously not sufficient. It is necessary to derive the carry signals at the intermediate positions as well. One way to accomplish this is by replicating the tree at every bit position. For instance, the carry at position 6 is computed by

combining the results of blocks (6:3) and (2:0). This complete structure, which frequently is referred to as a *Kogge-Stone tree* is a member of the *radix-2 trees*. *Radix-2* means that the tree is binary. It combines two carry words at a time at each level of hierarchy. The total adder requires 49 complex logic gates each to implement *dot* operator. In addition, 16 logic modules are needed for the generation of the propagate and generate signals at the first level (P_i and G_i), as well as 16 sum generation gates. [1].

4.3.2 8-bit logarithmic-lookahead adder based on Brent-Kung tree

Kogge-stone tree has some interesting properties. First, its interconnect structure is regular, which makes implementation quite easy. However the replication of the carry trees to generate the intermediate carries comes at a large cost in terms of both area and power. A simpler tree structure based on the following relations known as logarithmic-lookahead adder based on Brent-Kung tree or simply Brent-Kung adder.

$$\begin{aligned}
 (C_{0,0}, 0) &= (G_0, P_0) \cdot (C_{i,0}, 0) \\
 (C_{0,1}, 0) &= [(G_1, P_1) \cdot (G_0, P_0)] \cdot (C_{i,0}, 0) = (G_{1:0}, P_{1:0}) \cdot (C_{i,0}, 0) \\
 (C_{0,2}, 0) &= [(G_{2:1}, P_{2:1}) \cdot (G_0, P_0)] \cdot (C_{i,0}, 0) = (G_{2:0}, P_{2:0}) \cdot (C_{i,0}, 0) \\
 (C_{0,3}, 0) &= [(G_{3:2}, P_{3:2}) \cdot (G_{1:0}, P_{1:0})] \cdot (C_{i,0}, 0) = (G_{3:0}, P_{3:0}) \cdot (C_{i,0}, 0) \\
 (C_{0,4}, 0) &= [(G_{4:3}, P_{4:3}) \cdot (G_{2:0}, P_{2:0})] \cdot (C_{i,0}, 0) = (G_{4:0}, P_{4:0}) \cdot (C_{i,0}, 0) \\
 (C_{0,5}, 0) &= [(G_{5:3}, P_{5:3}) \cdot (G_{2:0}, P_{2:0})] \cdot (C_{i,0}, 0) = (G_{5:0}, P_{5:0}) \cdot (C_{i,0}, 0) \\
 (C_{0,6}, 0) &= [(G_{6:3}, P_{6:3}) \cdot (G_{2:0}, P_{2:0})] \cdot (C_{i,0}, 0) = (G_{6:0}, P_{6:0}) \cdot (C_{i,0}, 0) \\
 (C_{0,7}, 0) &= [(G_{7:4}, P_{7:4}) \cdot (G_{3:0}, P_{3:0})] \cdot (C_{i,0}, 0) = (G_{7:0}, P_{7:0}) \cdot (C_{i,0}, 0)
 \end{aligned}$$

By using above relations we can realize the carry bits. The schematic of 8-bit logarithmic-lookahead adder based on Brent-Kung tree or simply 8-bit Brent-Kung adder is shown in Figure.4.7. The triangles are buffers. These are the same as the inverter except that output nodes are swapped. Many buffers are used for maintaining a pipeline. These buffers propagate the correct logic values for addition like latch in the general pipeline structure. Pipeline structure uses all the blocks of a digital system, so it gives high throughput. The rectangular box is the basic cell and it is composed of three gates. We merged two gates-AND and OR gates- in the basic cell into one CPAL gate. The implementation of merged gate and basic cell is shown in

Figure.4.6. This adder can execute 8-bit addition per phase and six stages are needed to compute the result as shown in Figure.4.7. Using buffers increases the transistor count, but helps to ease layout due to regularity. [1]

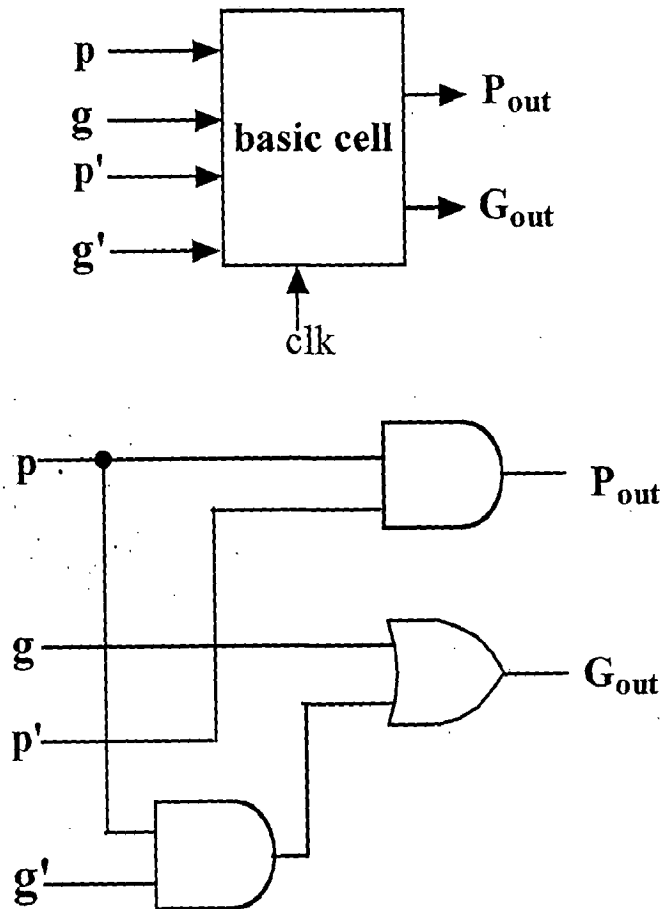


Figure .4.6 P,G signals generation block (dot gate)-[6]

Figure.4.6. This adder can execute 8-bit addition per phase and six stages are needed to compute the result as shown in Figure.4.7. Using buffers increases the transistor count, but helps to ease layout due to regularity. [1]

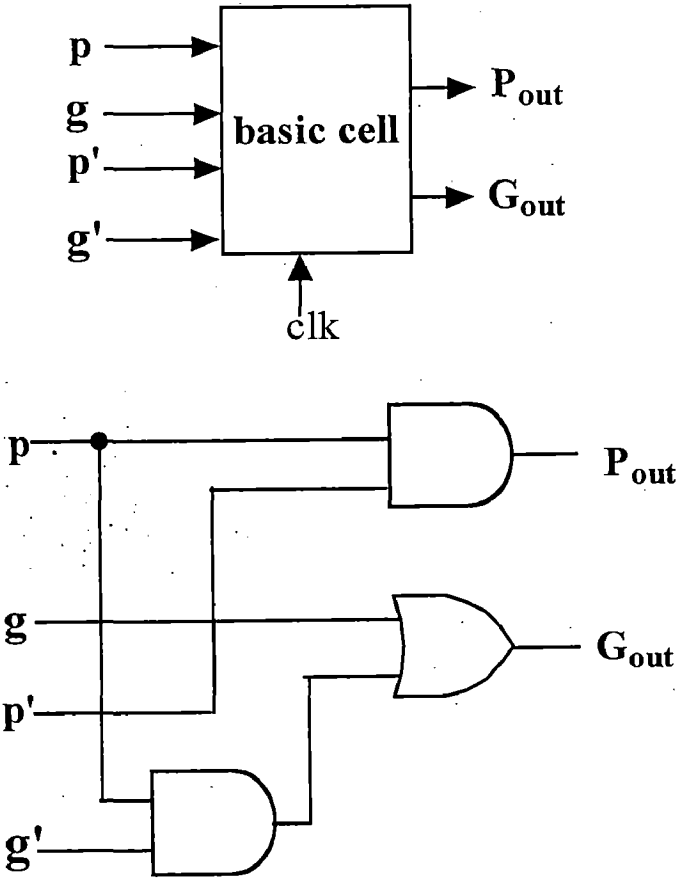


Figure .4.6 P,G signals generation block (dot gate)-[6]

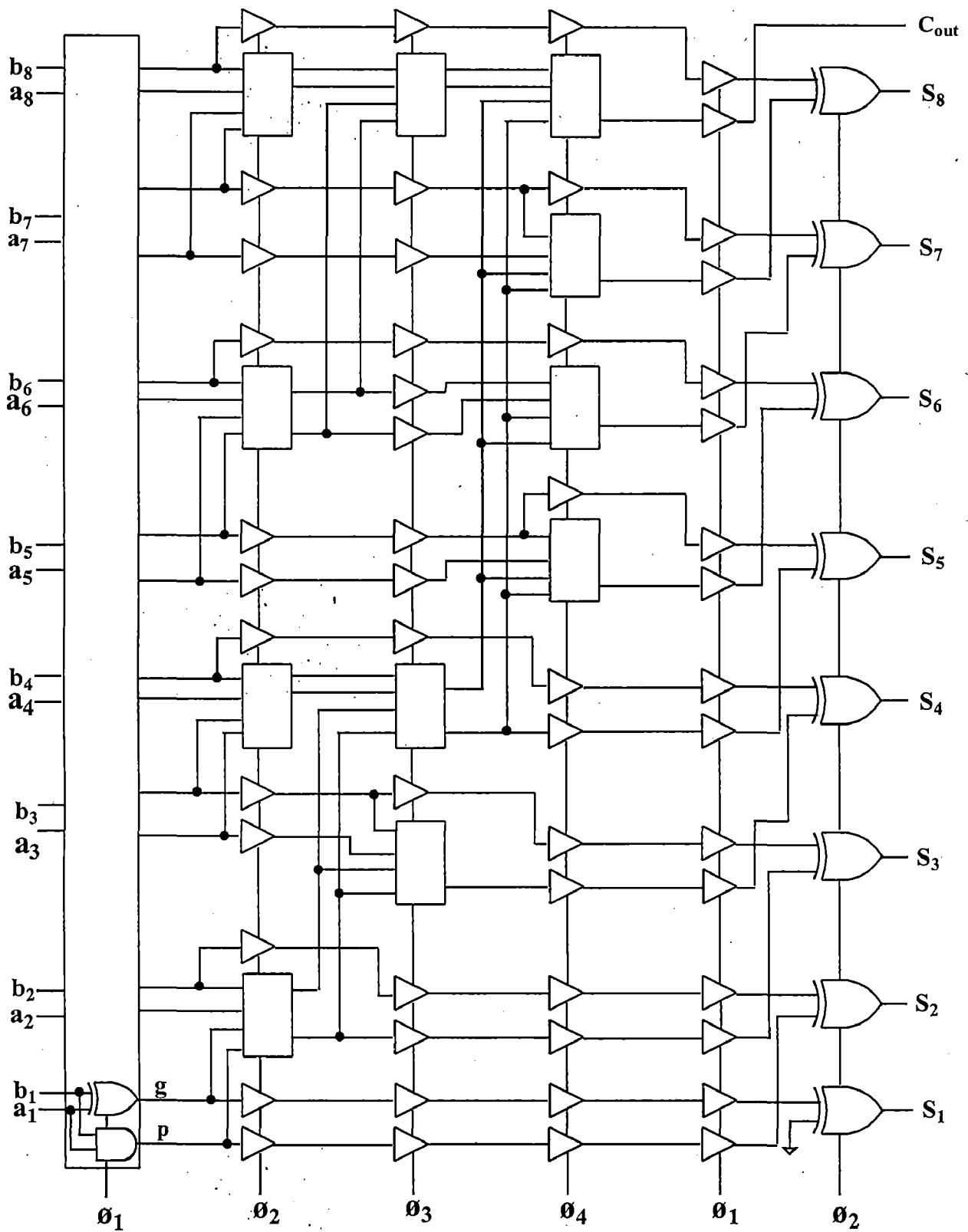


Figure.4.7 8-bit Brent-Kung adder. [1, 6]

4.4 MULTIPLIER

Multiplications are expensive and slow operations. The performance of many computational problems often is dominated by the speed at which a multiplication operation can be executed. This observation has, for instance, prompted the integration of complete multiplication units in state-of-the-art digital signal processors and microprocessors. Multipliers are in effect, complex adder arrays. The analysis of the multiplier gives us some further insight into how to optimize the performance (or the area) of complex circuit topologies. Consider two unsigned binary numbers X and Y that are M and N bits wide, respectively. Then the multiplication operation can be defined as follows:

$$Z = X \times Y = \sum_{K=0}^{M+N-1} Z_K 2^K$$
$$= \left(\sum_{i=0}^{M-1} X_i 2^i \right) \left(\sum_{j=0}^{N-1} Y_j 2^j \right) = \sum_{i=0}^{M-1} \left(\sum_{j=0}^{N-1} X_i Y_j 2^{i+j} \right)$$

The simplest way to perform a multiplication is to use a single two-input adder. For the inputs that are M and N bits wide, the multiplication takes M cycles, using an N -bit adder. This *shift-ana-add* algorithm for multiplication adds together M partial products. Each partial product is generated by multiplying the multiplicand with a bit of multiplier, which is essentially, is an AND operation and shifting the result on the basis of the multiplier bit's position. A faster way to implement multiplication is to resort to an approach similar to manually computing a multiplication. All the partial products are generated at the same time and organized in an array. The resulting structure is called an array multiplier. Block diagram of multiplier is shown in Figure.4.8. A multiplier can be divided into three stages:

1. Partial products generation stage.
2. Partial products addition stage.
3. The final addition stage.



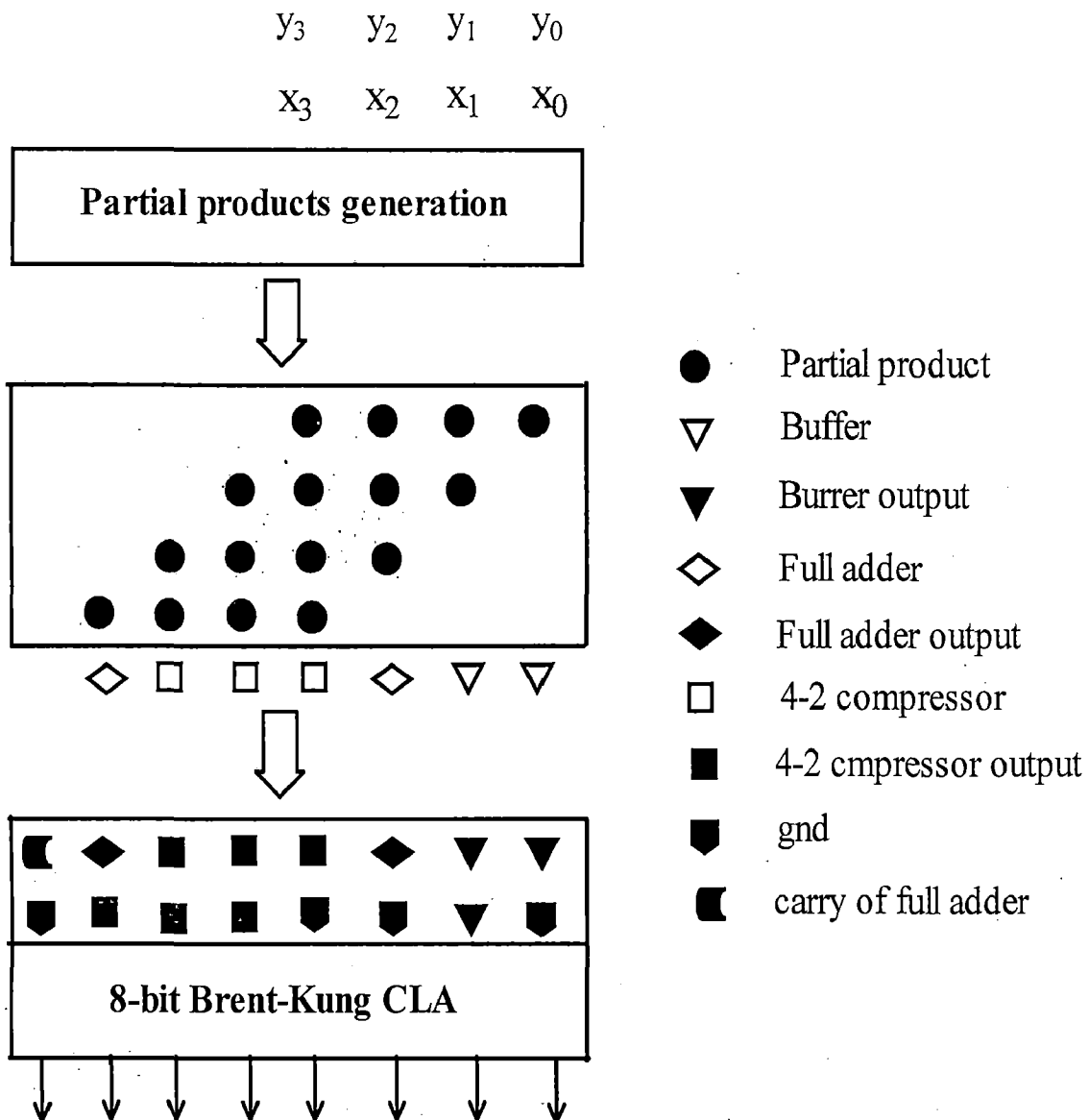


Figure.4.8 Block diagram of multiplier. [19]

4.4.1 PARTIAL PRODUCTS GENERATION STAGE

In this stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial products. The partial-products results from the logical two input AND of multiplicand X with a multiplier bit Y_i as shown in Figure.4.9. Each row in the partial product array is either a

copy of the multiplicand or a row of zeros. Careful optimization of the partial-product generation can lead to some substantial delay and area reduction. [1].

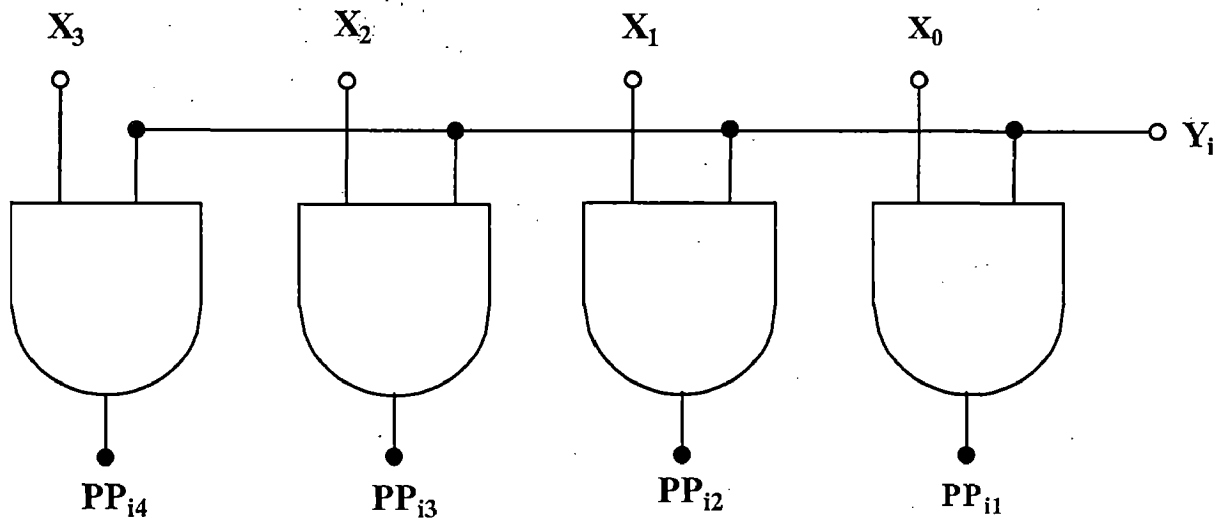


Figure 4.9. Generation of partial products. [1]

4.4.2 PARTIAL PRODUCTS ADDITION STAGE

The second stage is the most important, as it is the most complicated and determines the speed of the overall multiplier. The 4-2 and 5-2 compressors have been widely employed in the high speed multipliers to lower the latency of the partial product accumulation stage [19, 20]. Owing to its regular interconnection, the 4-2 compressor is ideal for the construction of regularly structured Wallace tree [21] with low complexity. In high-speed designs, the Wallace tree construction method is usually used to add the partial products in a tree-like fashion in order to produce two rows of partial products that can be added in the last stage. The Wallace tree is fast since the critical path delay is proportional to the logarithm of the number of bits in the multiplier. There exist a handful of ways to construct the Wallace Tree. The prominent method considers all the bits in each column at a time and compresses them into two bits (a sum and a carry). The Wallace tree is constructed by considering all the bits in each four row at a time and compressing them in an appropriate manner. Thus, compressors form the essential requirement of high speed multipliers. The speed, area and power consumption of the multipliers will be in

direct proportion to the efficiency of the compressors. Thus, in order to satisfy the requirement of small area, low power, high throughput circuitries, a 4-2 compressor is used in second stage of multiplier to compress the partial products. The design details of 4-2 compressor based on full adder mentioned in the section 4.4.2.1

4.4.2.1 4-2COMPRESSOR

The 4-2 compressor structures actually compress five partial products bits into three [12]. The architecture is connected in such a way that four of the inputs are coming from the same bit position of the weight j while one bit is fed from the neighboring position $j-1$ (known as carry-in). The outputs of 4-2 compressor consist of one bit in the position j and two bits in the position $j+1$. This structure is called compressor since it compresses four partial products into

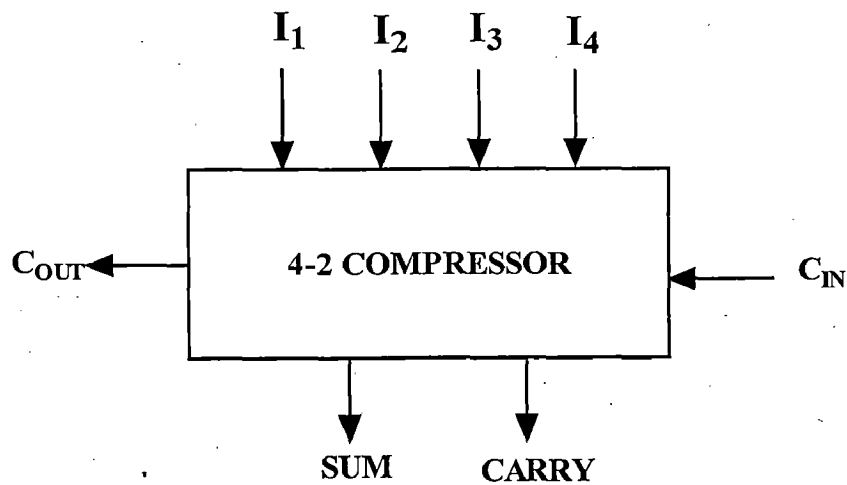


Figure 4.10 Block diagram of 4-2 compressor.[19]

two (while using one bit laterally connected between adjacent 4:2 compressors). Figure.4.10 shows the block diagram of 4-2 compressor. A 4-2 compressor can also be built using 3-2 compressors. It consists of two 3-2 compressors (full adders) in series as shown in Figure.4.11. The output C_{OUT} , being independent of the input C_{IN} accelerates the carry 'save summation of the partial products. A 4-2 compressor has five inputs and three outputs, as shown in Figure.4.10. The four inputs (I_1 , I_2 , I_3 , and I_4), and the output (SUM) have the same weight. The output CARRY is weight one binary bit order higher. The 4-2 compressor receives an input C_{IN} from the preceding module of one binary bit order lower in significance, and produces an output C_{OUT}

to next compressor module of higher significance. To accelerate the carry save summation of the partial products, the output C_{OUT} must be independent of the input C_{IN} . The compressors have large load capacitances, which consist mostly of the wire parasitic capacitor, because they route often over several cells. The compressors are implemented using the CPAL to reduce energy dissipation. The conventional implementation of a 4-2 compressor is composed of two serially connected full adders as shown in Figure.4.11. We can also use CPAL full adders to configure a 4-2 compressor. The summary and carry logic of CPAL full adders are shown in Figure.4.5 [22]

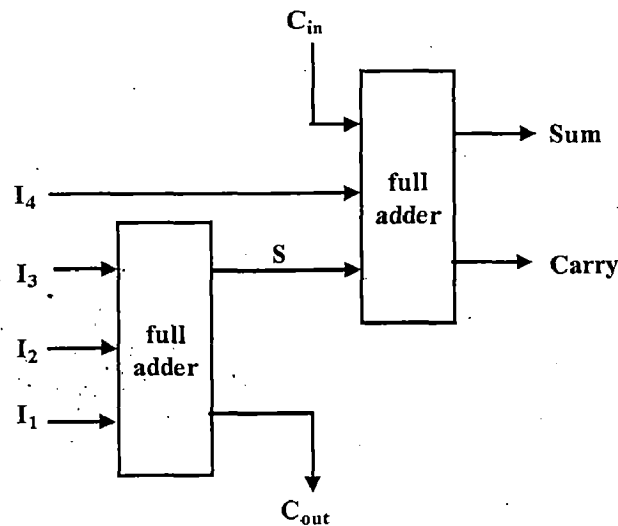


Figure.4.11 4-2 compressor based on full adders.[19]

4.4.3 THE FINAL ADDITION STAGE

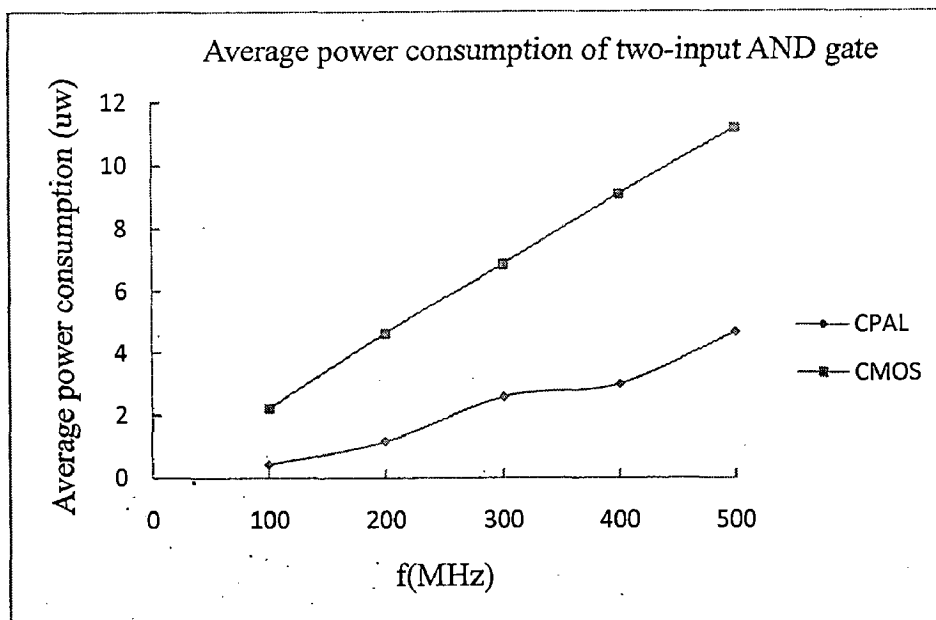
The final step for completing the multiplication is to combine the result in the final adder. Performance of this “vector-merging” operation is of key importance. The choice of the adder style depends on the structure of accumulation array. A carry-lookahead adder is preferable as it yield the smallest possible delay. The last stage of the present multiplier is a 8-bit logarithmic lookahead adder based on Brent-Kung tree or simply 8-bit Brent-Kung adder, which is realized using CPAL. The buffers shown are used for maintaining a pipeline. Block diagram of 8-bit Brent-Kung adder is shown in Figure.4.7. [1]

5.1 TWO-INPUT AND FOUR-INPUT LOGIC GATES

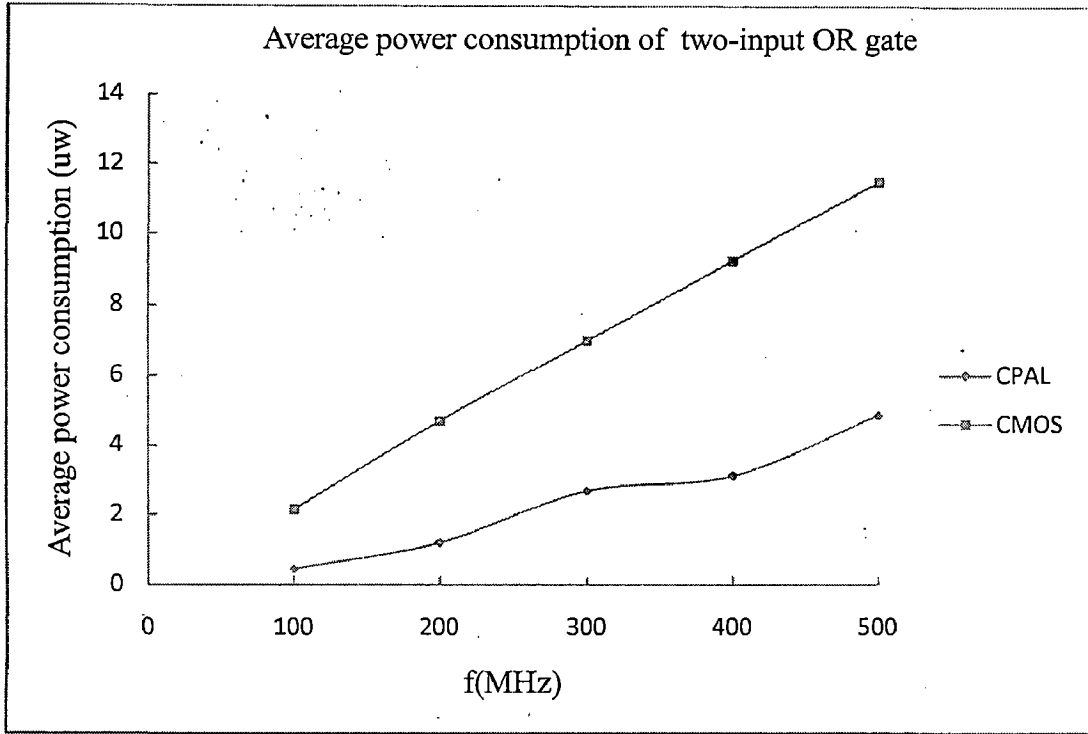
As we have seen in chapter.2 and chapter.3 that CPAL technique has the lesser power consumption than the CMOS logic. Simple two-input and four-input logic gates were simulated using the SPICE for CPAL and CMOS logic to evaluate the average power consumption for the two logics at different frequencies.

The SPICE simulation of two-input and four-input logic gates was carried out using Predictive Technology Model Beta Version 130nm [23]. The simulation was carried up to 500MHz, and the peak voltage of power clock V_{clk} is 1.8V. The aspect ratio of all pMOSFET's is $W/L=6\lambda/2\lambda$ and for nMOSFETS's it is $W/L=3\lambda/2\lambda$ except for N5 and N6 for which it is $W/L=6\lambda/2\lambda$ with $\lambda=65\text{nm}$

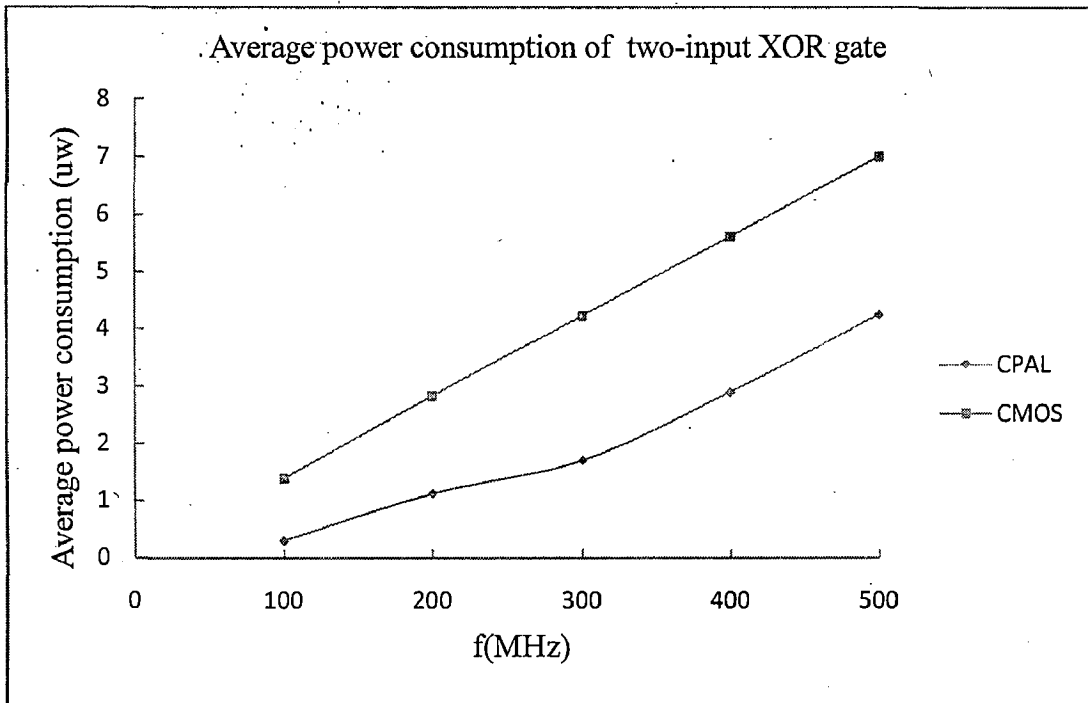
The comparison of average power consumption for two-input and four-input gates using CPAL and CMOS logic is shown in below Figure.5.1 for different frequencies up to 500MHz.



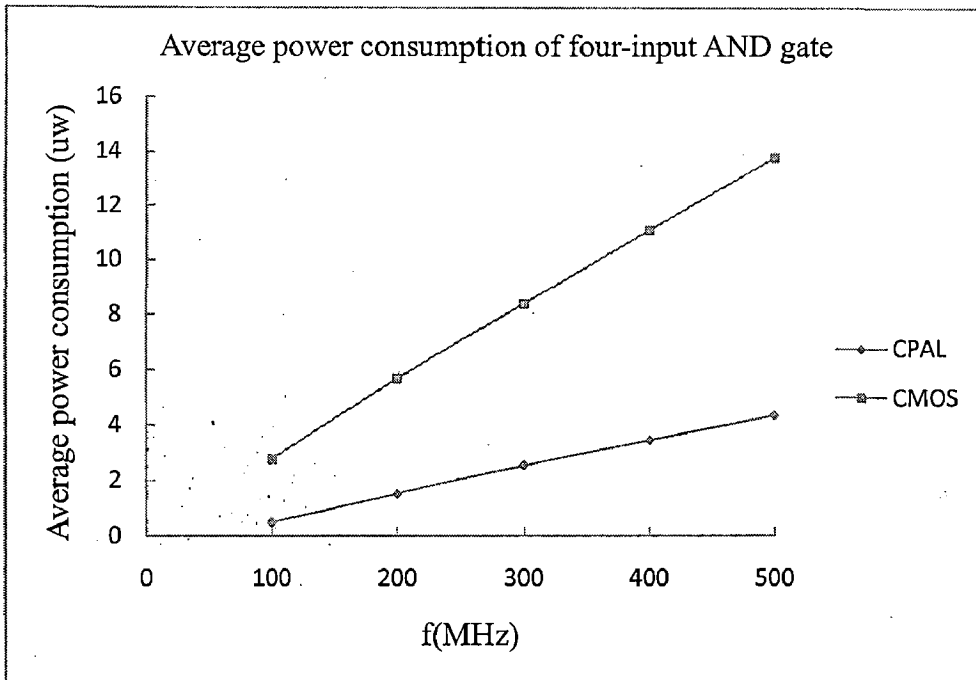
(a)



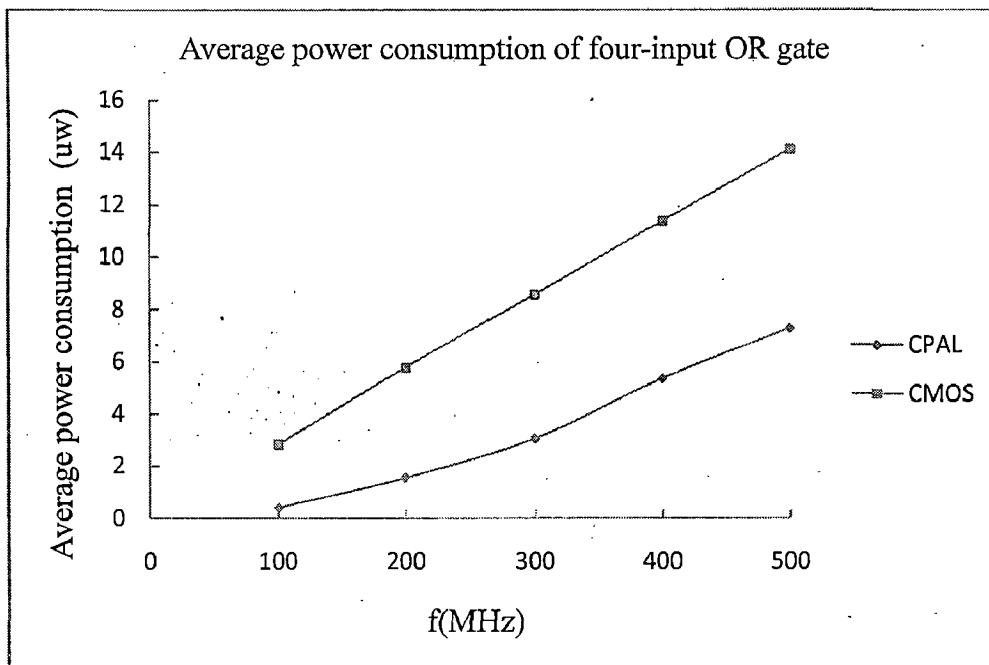
(b)



(c)



(d)



(e)

Figure.5.1 Average power consumption of two-input and four-input logic gates

Therefore from the above Figure.5.1, it is clear that the CPAL has the lesser power consumption than the CMOS logic at all frequencies. The CPAL also has full voltage swings at the out put nodes as these nodes are charged and discharged adiabatically through the transmission gates which are present in load driven circuit. Thus the CPAL is superior to the CMOS logic in terms of average power consumption.

5.2 FULL ADDER AND 4-2 COMPRESSOR

Full adder and 4-2 compressor (which are based on full adder) were implemented using SPICE with CPAL and CMOS. SPICE simulation is carried out using Predictive Technology Model Beta Version 130nm [23]. The design was verified up to 500MHz, and the peak voltage of power clock V_{clk} is 1.8V, also V_{DD} for CMOS is 1.8v. The aspect ratio of all pMOSFET's is $W/L=6\lambda/2\lambda$ and for nMOSFETS's it is $W/L=3\lambda/2\lambda$ except for N5 and N6 for which it is $W/L=6\lambda/2\lambda$ with $\lambda=65\text{nm}$. Figure.5.2 and Figure.5.3 shows the comparison of average power consumption at different frequencies between CPAL and CMOS logics for full adder and 4-2 compressor respectively.

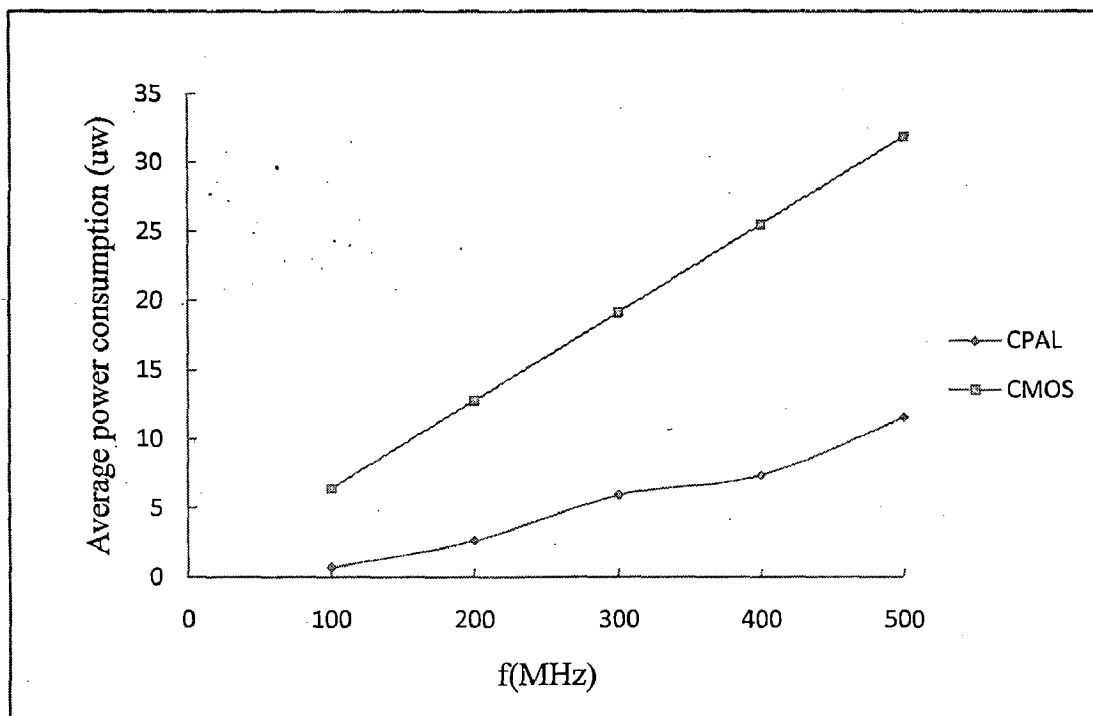


Figure.5.2 Average power consumption of full adder

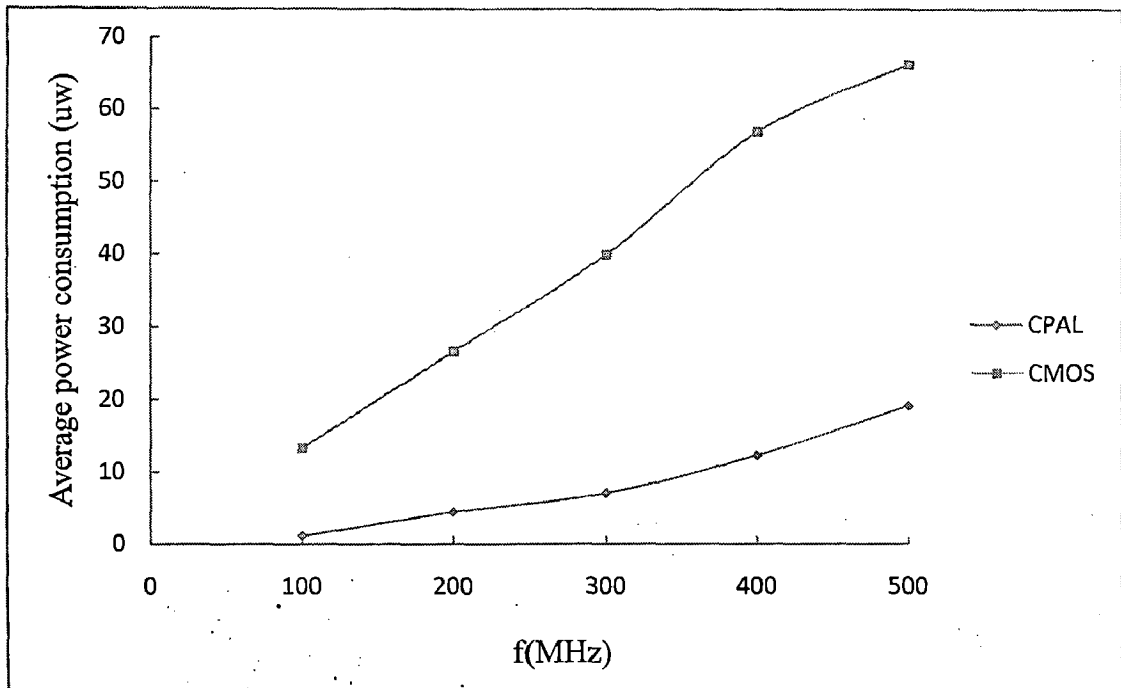


Figure.5.3 Average power consumption of 4-2 compressor

Number of transistors required to implement these modules in both logics are almost equal. So the area required for design in both logics is almost same. From Figure.5.2 and Figure.5.3 it is clear that CPAL shows much decrease in average power consumption than CMOS logic.

5.3 FOUR-STAGE BUFFER CHAIN

The simple buffer (or inverter) circuit of Figure.3.1 can be extend to form the buffer chain of four stages driven by four-phase power-clocks as shown in Figure.3.2. which is one of the main module in carry-lookahead adders for pipeline structure.

The SPICE simulation of four stage buffer chain was carried out using Predictive Technology Model Beta Version 130nm [23]. The simulation was carried up to 500MHz, and the peak voltage of power clock V_{clk} is 1.8V and also V_{DD} is 1.8v for CMOS logic. The aspect ratio of all pMOSFET's is $W/L=6\lambda/2\lambda$ and for nMOSFETS's it is $W/L=3\lambda/2\lambda$ except for N5 and N6 for which it is $W/L=6\lambda/2\lambda$ with $\lambda=65nm$.

Figure.5.3 shows the comparison of average power consumption at different frequencies between CPAL and CMOS logics.

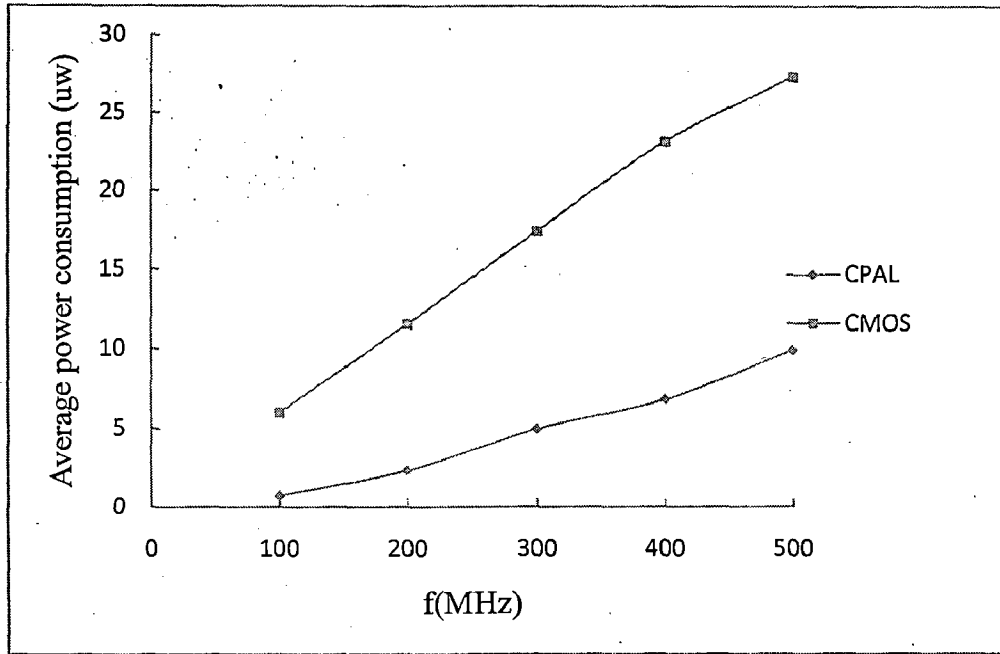


Figure.5.4 Average power consumption of buffer chain

5.4 8-BIT BRENT-KUNG ADDER AND 4-BIT MULTIPLIER

The Digital circuits like adders and multipliers are the most power consuming block in microprocessors. Also multiplier is the most critical element in DSP functional modules. The CPAL has lesser power consumption than the conventional CMOS logic as seen above. So, if we extend this logic to adders and multipliers, most of the power can be saved and even the battery life of the portable electronic devices can be increased.

Hence 8-bit Brent-Kung adder and 4-bit multiplier circuits of Figure.4.7 and Figure.4.8 were implemented using CPAL 130nm using SPICE.

The SPICE simulation of four stage buffer chain was carried out using Predictive Technology Model Beta Version 130nm [23]. The simulation was carried up to 500MHz, and the peak voltage of power clock V_{clk} is 1.8V and also V_{DD} is 1.8v for CMOS logic. The aspect ratio of all pMOSFET's is $W/L=6\lambda/2\lambda$ and for nMOSFETS's it is $W/L=3\lambda/2\lambda$ except for N5 and N6 for which it is $W/L=6\lambda/2\lambda$ with $\lambda=65nm$.

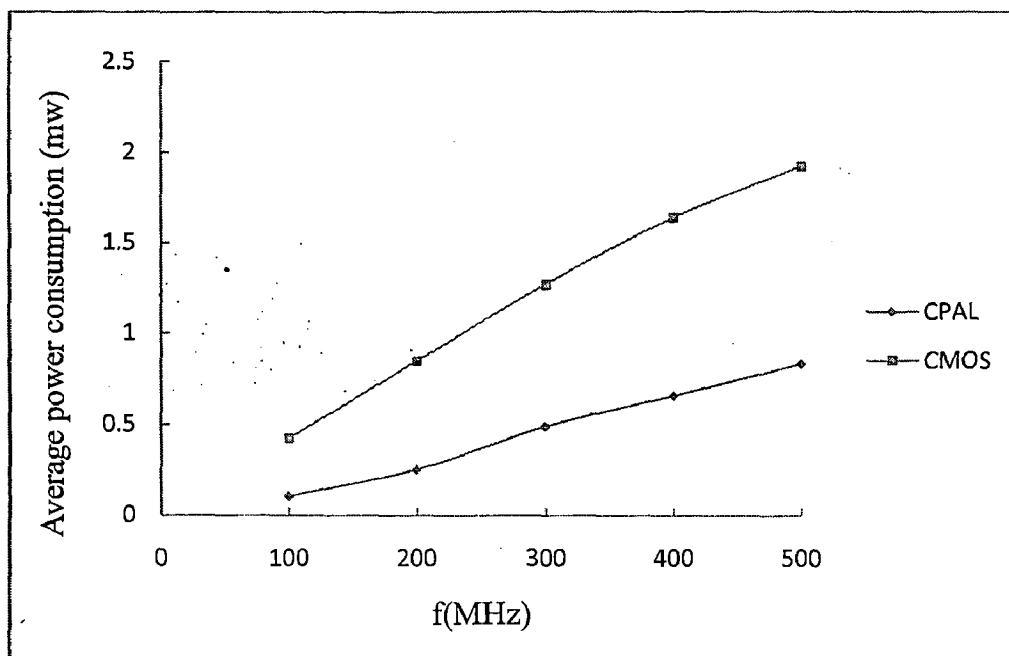
Figure.5.5 and Figure.5.6 shows the comparison of average power consumption at different frequencies between CPAL and CMOS logics for 8-bit Brent-Kung adder and 4-bit

multiplier. The output waveforms of 4-bit multiplier and 8-bit Brent-Kung adder are shown in Figure.5.7 and Figure.5.8 respectively.

The comparison of power consumption versus operation frequency has been made against CPAL and CMOS 4-bit multiplier and 8-bit Brent-Kung adder at 130nm is tabulated in Table.1.

Frequency(MHz)	Average power consumption (mW)			
	8-bit Brent-Kung adder		4-bit multiplier	
	CPAL	CMOS	CPAL	CMOS
100	0.1080	0.424	0.1370	0.4856
200	0.2550	0.847	0.3765	0.9704
300	0.4892	1.270	0.6200	1.4530
400	0.6570	1.640	0.9530	1.8360
500	0.8310	1.920	1.3507	2.2506

Table.1 Power consumption of 8-bit Brent-Kung adder and 4-bit multiplier



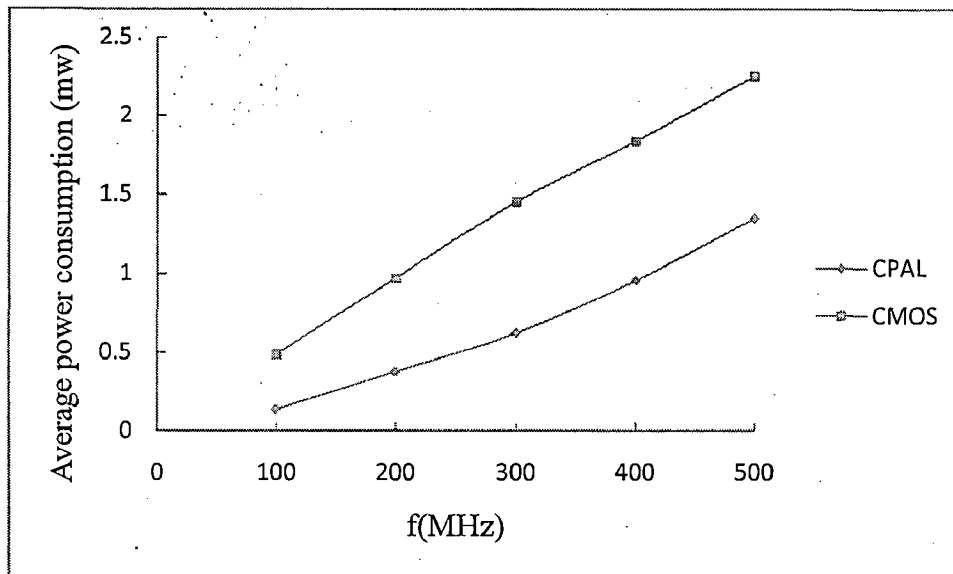


Figure.5.6 Average power consumption of 4-bit multiplier

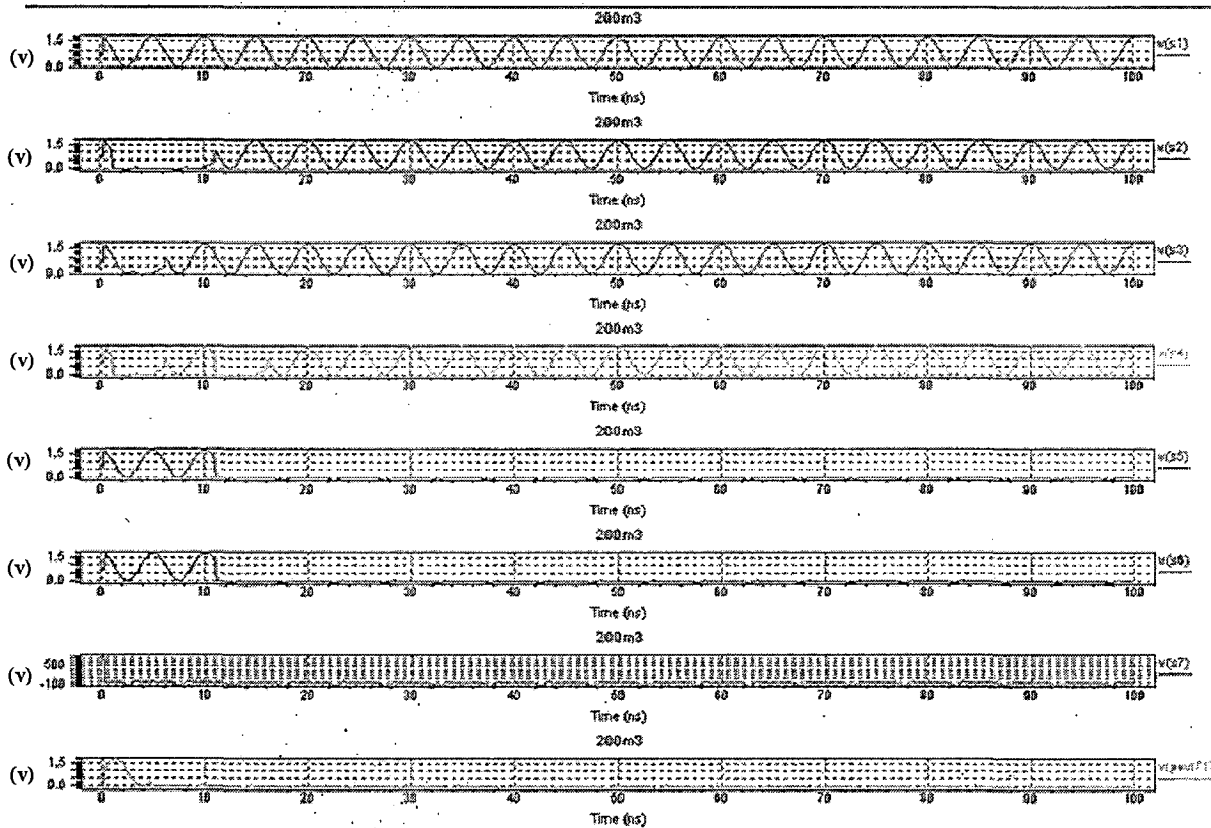


Figure.5.7(a). Output waveform of 4-multiplier using CPAL

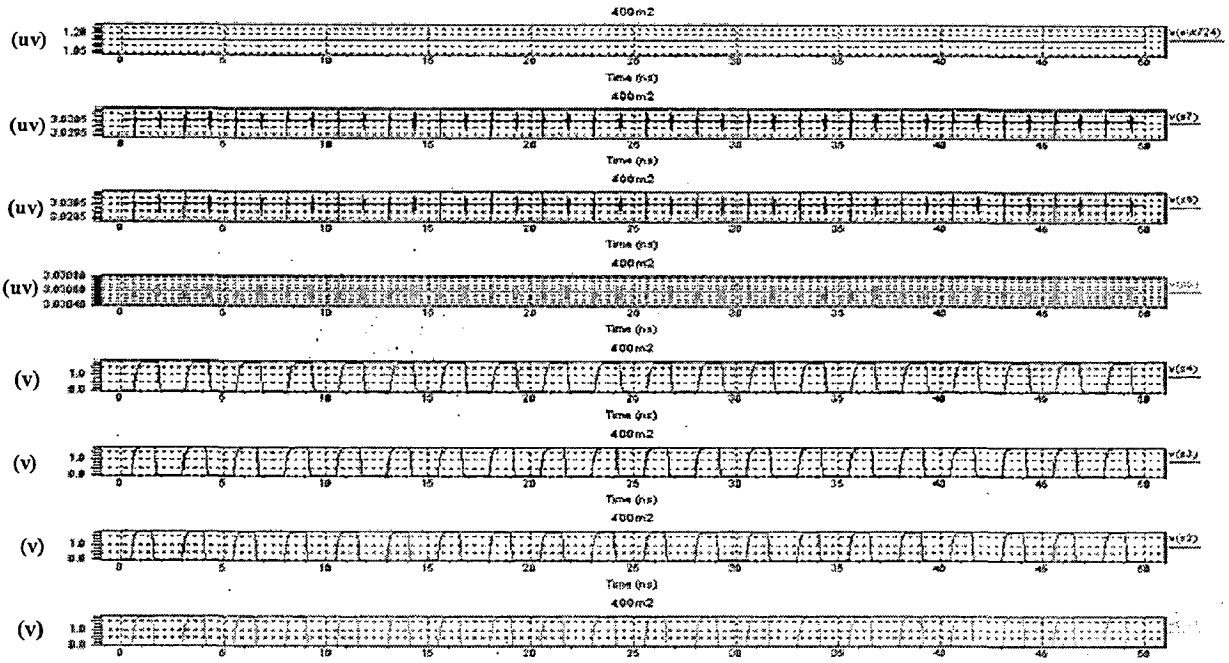


Figure.5.7(b). Output waveform of 4-bit multiplier using CMOS

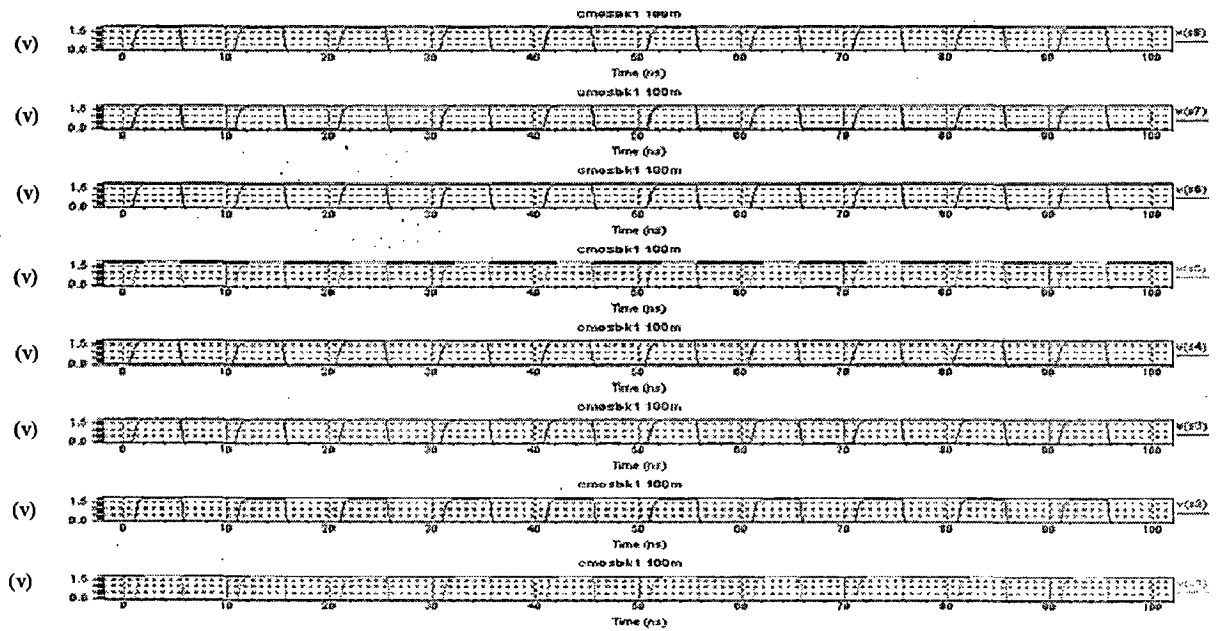


Figure.5.8(a). Output waveform of 8-bit Brent-Kung adder using CMOS

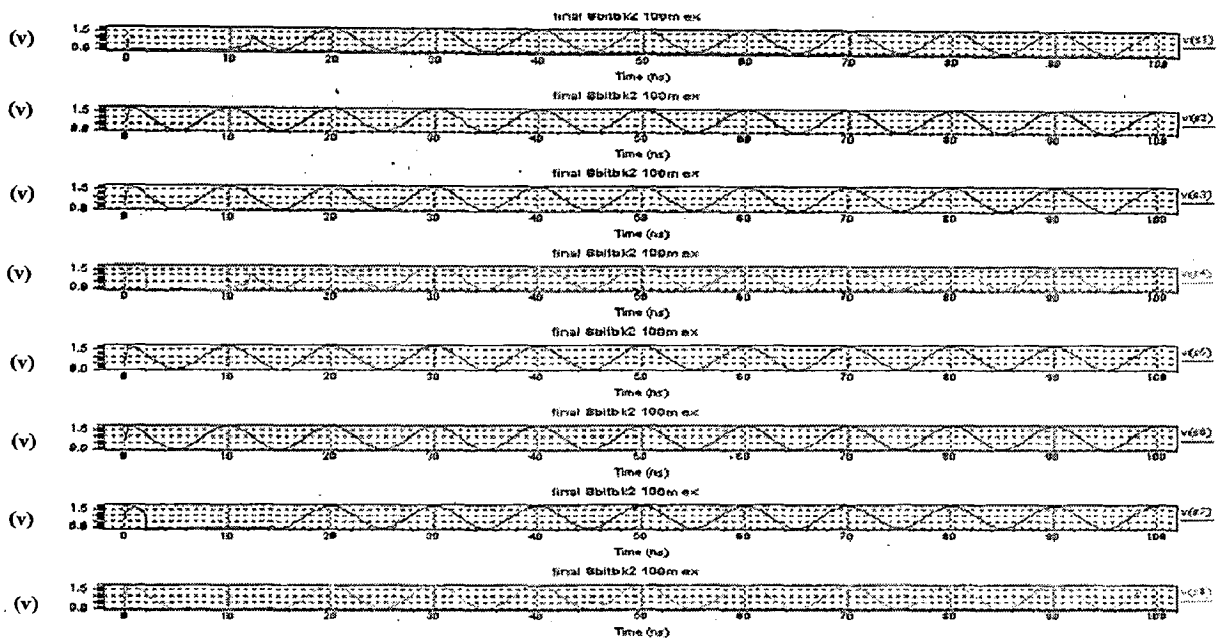


Figure.5.8(b) Output waveform of 8-bit Brent-Kung adder using CPAL

Apart from the power dissipation area of the chip needed for a particular design is also important. The CPAL increase the number of transistors needed for a particular design so, it is also important to consider even the area need for a design. Therefore it is necessary to compare number of transistors required for different arithmetic circuits using CPAL and CMOS logic. Table.2 shows the number of transistors required for different circuits using CPAL and CMOS logic.

Circuit/gate	Transistor count	
	CMOS	CPAL
Two-input gates	6	10
Four-input gates	10	18
Four-stage buffer chain	16	24
Full adder	28	32
4-2 compressor	56	64
8-bit Brent-Kung adder	558	784
4-bit multiplier	734	1014

Table.2 Transistor count for Arithmetic and Logic circuits in CMOS and CPAL design

CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

The CPAL is a promising technique for Low Power Applications as it consumes less power than conventional CMOS logic.

At lower frequencies CPAL has much low power consumption, but as the frequency increases the degree of decrease in power consumption is lower which is obvious. But the overall power consumption is lower than that of CMOS logic even at high frequencies. Also the logic levels for this adiabatic logic are still rail to rail.

All combinational circuits can be implemented by CPAL. The Digital circuits such as adders and multipliers consuming large amount of power can be designed using CPAL offering less power consumption even in the presence of large load capacitance.

The number of MOSFETS needed for CPAL circuits is higher than the CMOS counter parts.

6.2 FUTURE SCOPE

Adiabatic logic uses sinusoidal or trapezoidal clock signal with multiple phase, further study on how to design an efficient power clock generator can be done.

While implementing the Adiabatic Arithmetic and Logic circuits using CPAL, transistor sizes ($W/L=1.5$ and 3) were used. Further study on how to optimize the transistors size depending up on the load capacitance, to obtain still reduction in power consumption.

CPAL can be extended to design sequential circuits and memories.

REFERENCES:

- [1]. Jan M. Rabaey ,Anantha Chandrakasan, Borivoje Nikolic, “Digital Integrated Circuits A Design Perspective,” Second edition, Low Price Edition.
- [2].R. Rogenmoser, H. Kaeslin, and N. Felber, “The impact of transistor sizing on power efficiency in submicron CMOS circuits”, Proc. 22nd European Solid-State Circuits Conf., Neuch[^]atel, Switzerland, pp. 124–127,Sept. 1996.
- [3].C. Piguet, J.-M. Masgonty, S. Cserveny, and E. Dijkstra, “Low-power low-voltage digital CMOS cell design”, in Proc. PATMOS’94, Barcelona, Spain, pp. 132–139,Oct. 1994.
- [4].N. Ohkubo, and M. Suzuki “A 4.4 ns CMOS 54x 54-b multiplier using pass transistor multiplexer”, IEEE J. Solid-State Circuits, vol. 30, pp. 251–257, Mar. 1995.
- [5].S. Kim, and M. C. Papaefkymiou, “True single-phase adiabatic circuitry,” IEEE Trans. on VLSI Systems, vol. 9(1), pp. 52-63, 2001.
- [6].Y. Moon, and D. K. Jeong. , “An efficient charge recovery logic circuit,” IEEE J. of Solid-State Circuits, Vol. 31(4), pp. 514- 521,1996.
- [7].A. Kramer, J. S. Denker, B. Flower, and J. Moroney, “2nd order adiabatic computation with 2N-2P and 2N-2N2P logic circuits,” Proc. of the International Symposium on Low Power Electronics and Design, pp. 191-196,1995.
- [8].F. Liu, and K. T. Lau, “Pass-transistor adiabatic logic with NMOS pull-down configuration,” Electronics Letters, Vol. 34(8), pp. 739-741,1998.
- [9].Warren D. Smith, “Fundamental physical limits on computation,” Technical Report, NECI, May 1995.
- [10]. M. P. Frank, “Common Mistakes in Adiabatic Logic Design and How to Avoid Them,” Proceedings of International Workshop on Methodologies in Low-Power Design, 2003
- [11].A Schlaffer and J.A.Nossek,”is there a connection between adiabatic switching and reversible computing?”,Institute for Network theory and Circuit design,Munich University of Technology, <http://citeseer.nj.nec.com/schlaffer97is.html>.
- [12]. Jianping Hu, Tiefeng Xu, and Yinshui Xia, “Low-Power Adiabatic Sequential Circuits with Complementary pass-Transistor Logic,” Proc. 48th Midwest Symposium on Circuits and Systems, Cincinnati, USA, Vol. II, pp. 1398-1401, Aug. 2005.

- [13]. K. Yano, T. Yamanaka, T. Nishida, and M. Satio, "A 3.8-ns cmos 16-b multiplier using complementary pass-transistor logic," *IEEE J. of Solid-State Circuits*, Vol. 25(2), pp. 388-395, 1990.
- [14]. C. Kim, S. M. Yoo, and S. M. Kang, "Low power adiabatic computing with NMOS energy recovery logic," *Electronics Letters*, Vol. 36(16), pp.1349-1350, 2000.
- [15]. J. P. Hu, W. J. Zhang, and Y. S. Xie, "Complementary Pass- Transistor Adiabatic Logic and Sequential Circuits Using Three-Phase Power Supply," *Proc. 47th Midwest Symp. on Circuits and Systems*, Hiroshima, Japan, Vol. II, pp.201-204, July 2004.
- [16]. Jianping Hu, Tiefeng Xu and Yinshui Xia, "Low-power adiabatic sequential circuits with complementary pass-transistor logic," *IEEE Symp. Circuits and Systems*, Vol. 2, pp. 1398-1401, 2005.
- Sung-Mo Kang, and Yusuf Leblebici, "CMOS Digital Integrated Circuits Analysis and Design," Third edition, Tata McGraw-Hill.
- Joonho Lim, Dong-Gyu Kim, and Soo-Ik Chae, "A 16-bit Carry-Lookahead Adder Using Reversible Energy Recovery Logic for Ultra-Low-Energy Systems," *IEEE J. of Solid-State Circuits*, Vol. 34, pp. 898-903, Jun. 1999.
- Yangbo Wu, Weijiang Zhang, and Jianping Hu, "Adiabatic 4-2 Compressors for Low-Power Multiplier," *Proc. 48th Midwest Symposium on Circuits and Systems*, Cincinnati, USA, Vol. II, pp. 1473-1476, Aug. 2005.
- Xien Ye, Jianping Hu, and Weijiong Tao, "A Low-Power Complementary Pass-Transistor Tree Multiplier Based on Adiabatic 4-2 compressors," *IEEE Proc. of 6th Inter. Conf. on ASIC*, (Beijing, China) p. 1240, 2005.
- V. Oklobdzija, "High-Speed VLSI Arithmetic Units: Adders and Multipliers", in "Design of High-Performance Microprocessor Circuits", Book Chapter, Book edited by A. Chandrakasan, IEEE Press, 2000.
- Y. Kazuo, Y. Toshiaki and N. Takashi, "A 3.8ns CMOS 8x8-b multiplier using complementary pass-transistor logic," *IEEE J. of Solid-State Circuits*, vol. 25(2), pp. 388-395, 1990.
- BPTM models available at <http://www.eas.asu.edu/~ptm/>

LIST OF PUBLICATIONS

1. A. Rajasekhara Reddy and S. Dasgupta, "8-bit Brent-Kung adder design using Complementary Pass-Transistor Adiabatic Logic". International Conference on Microelectronics 2008, Sharjah, UAE (communicated)
2. A. Rajasekhara Reddy and S. Dasgupta, "4-bit multiplier design using Complementary Pass-Transistor Adiabatic Logic", 22nd International Conference on VLSI Design, January 2009, New Delhi, India (communicated)