**CHAPTER-1**

**INTRODUCTION**

**1.1 Motivation**

Integrated circuits and sophisticated electronic circuits are the drivers of modern human civilization. Each and every electronic gadget is made using these integrated circuits. it becomes hard to imagine a civilization without these electronic devices.

The increasing complexity of VLSI circuits poses an additional overhead of increased power consumption. The manual process of optimization may prove to be tedious and hopeless. Software based techniques may be utilized to increase the pace of power optimization. Genetic algorithm is a search algorithm which mimics the process of natural evolution. This heuristics technique helps us find solutions which are reasonably good.

**1.2 Literature Review**

In feb, 2002 Tarek. K darwish and Ahmed M. Shams designed a technique to generate low power cmos full adder circuit. They divided the full adder circuit into several smaller modules. These modules were redesigned to achieve lower power dissipation. Various modules were connected together to form low power cmos full adder. The problem with this method was the overhead involved in the design and analysis of each module [1].

2002, Mohammed Sayed and Wael Badawy, proposed three new full adder circuits and compared thirty one others for power consumption. All the cells had different structures using XOR and XNOR gates. They tried to implement the circuit using .18, .25 and .35u m technologies. An additional buffer is used to generate the input signal, which is an extra overhead and might affect the power consumption of the circuit, but somehow this has not been taken into account. [4]

Chip Hong Chang, Mingyan Zhang and Jiangmin gu proposed a new hybrid style full adder circuit. This full adder circuit has been developed on .18u m technology specially for complex arithmetic circuits. The proposed hybrid full adder not only exhibits full swing logic but also high drivability. The circuit suffers from a threshold voltage drop problem. [8]

Amir Amirabadi and Ali Afzali Kusha tried to optimized power and delay of full adder circuit by using simulated annealing. A cost function of power and delay is computed using HSPICE and simulated annealing was implemented using MATLAB. The techniques helps in reducing the delay without allowing the power to increase significantly. Out of numerous simulated circuits it may be possible that some circuit may result in incorrect function, these circuits may be attributed a large cost. [10]

Uming ko and Poras T. Balsara have implemented a 32-bit carry look ahead adder on .6u m cmos technology. The same circuit has been implemented using technique such as double pass transistor logic (DPL), complementary pass transistor logic (CPL) and single rail domino. The DPL was found to be the most efficient. The number of transistors used for DPL and rail domino is quite high, CPL uses fewer transistors but the partial swings lead to a lot of power wastage. [13]

**1.3 Objective of the Project**

A lot of work has been done to optimize power of the digital circuits. The most common methodology followed by the researchers is to sub divide the circuit in to smaller modules, then change the structure of the modules and the finally reconnect various modules to form new circuits having the same logic output. CMOS transistors may be replaced by CPL, pass gates etc to form modules with lower power dissipation. Unlike all these approaches we have tried to optimize the power by choosing the correct and optimized values of parameters that affect the working of the circuit.

We very well know that the dynamic power of a circuit depends on the supply voltage, load capacitance and switching frequency. We tried to optimize these values and thus the dynamic power by making use of the genetic algorithm. Several other heuristic search techniques have been implemented to optimize the power of the circuit. However the work done by using genetic algorithm is relatively rare. We have devoted our effort to explore this relatively unexplored field.

**CHAPTER-2**

**OPTIMIZATION**

**2.1 Introduction**

Optimization is the process of making something better. An engineer or scientistconjures up a new idea and optimization improves on that idea. Optimization consists in trying variations on an initial concept and using the information gained to improve on the idea. A computer is the perfect tool for optimization as long as the idea or variable influencing the idea can be input in electronic format. Feed the computer some data and out comes the solution Optimization is the math tool that we rely on to get these answers. A simple example reveals many shortfalls of the typical minimum seekers. Since the local optimizers of the past are limited, people have turned to more global methods based upon biological processes.

**2.2 Finding the Best solution**

The terminology “best” solution implies that there is more than one solution and the solutions are not of equal value. The definition of best is relative to the problem at hand, its method of solution, and the tolerances allowed. Thus the optimal solution depends on the person formulating the problem. Education, opinions, bribes, and amount of sleep are factors influencing the definition of best. Some problems have exact answers or roots, and best has a specific definition. Other problems have various minimum or maximum solutions known as optimal points or extreme, and best may be a relative definition.

**2.3 What Is Optimization?**

Our lives confront us with many opportunities for optimization. What time do we get up in the morning so that we maximize the amount of sleep yet still make it to work on time? What is the best route to work? Which project do we tackle first? When designing something, we shorten the length of this or reduce the weight of that, as we want to minimize the cost or maximize the appeal of a product. Optimization is the process of adjusting the inputs to or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or result. The input consists of variables; the process or function is known as the cost function, objective

function, or fitness function; and the output is the cost or fitness. If the process is an experiment, then the variables are physical inputs to the experiment. For most of the examples, we define the output from the process or function as the cost. Since cost is something to be minimized, optimization becomes minimization. Life is interesting due to the many decisions and seemingly random events that take place. Quantum theory suggests there are an infinite number of dimensions, and each dimension corresponds to a decision made.

Life is also highly nonlinear, so chaos plays an important role too. A small perturbation in the initial condition may result in a very different and unpredictable solution. These theories suggest a high degree of complexity encountered when studying nature or designing products. Science developed simple models to represent certain limited aspects of nature. Most of these simple (and usually linear) models have been optimized. In the future, scientists and engineers must tackle the unsolvable problems of the past, and optimization is a primary tool needed in the intellectual toolbox



Figure 2.1Diagram of a function or process that is to be optimized. Optimization varies the input to achieve a desired output.

**2.4 Categories of Optimization**

The given divides optimization algorithms into six categories None of these six views or their branches are necessarily mutually exclusive. For instance, a dynamic optimization problem could be either constrained or unconstrained. In addition some of the variables



Fig 2.2 Six categories of optimization algorithms

may be discrete and others continuous. Let’s begin at the top left of Figure 1.2 and work our way around clockwise.

**2.4.1 Trial-and-error optimization**

It refers to the process of adjusting variables that affect the output without knowing much about the process that produces the output. Various mathematical manipulations of the function lead to the optimal solution. Theoreticians love this theoretical approach.

**2.4.2 Single variable and multiple variable optimizations**

If there is only one variable, the optimization is one-dimensional. A problem having more than one variable requires multidimensional optimization. Optimization becomes increasingly difficult as the number of dimensions increases. Many multidimensional optimization approaches generalize to a series of one-dimensional approaches.

**2.4.3 Static and Dynamic optimization**

Dynamic optimization means that the output is a function of time, while static means that the output is independent of time. When living in the suburbs of Boston, there were several ways to drive back and forth to work. What was the best route? From a distance point of view, the problem is static, and the solution can be found using a map or the odometer of a car. The shortest route isn’t necessarily the fastest route. Finding the fastest route is a dynamic problem whose solution depends on the time of day, the weather, accidents, and so on. The static problem is difficult to solve for the best solution, but the added dimension of time increases the challenge of solving the dynamic problem.

**2.4.4 Discrete or Continuous variables Optimization**

Optimization can also be distinguished by either discrete or continuous variables. Discrete variables have only a finite number of possible values, whereas continuous variables have an infinite number of possible values. Discrete variable optimization is also known as combinatorial optimization, because the optimum solution consists of a certain combination of variables from the finite pool of all possible variables.

**2.4.5 Constrained and Unconstrained Optimization**

Variables often have limits or constraints. Constrained optimization incorporates variable equalities and inequalities into the cost function. Unconstrained optimization allows the variables to take any value. A constrained variable often converts into an unconstrained variable through a transformation of variables. When constrained optimization formulates variables in terms of linear equations and linear constraints, it is called a linear program. When the cost equations or constraints are nonlinear, the problem becomes a nonlinear programming problem.

**2.4.6 Random and Minimum seeking Optimization**

Some algorithms try to minimize the cost by starting from an initial set of variable values. These minimum seekers easily get stuck in local minima but tend to be fast. They are the traditional optimization algorithms and are generally based on calculus methods. On the other hand, random methods use some probabilistic calculations to find variable sets. They tend to be slower but have greater success at finding the global minimum.[16]

**CHAPTER-3**

**FULL ADDER CIRCUIT**

**3.1 What is a Full Adder ?**

A 1‐bit adder takes two 1‐bit inputs and adds them together. To make it a full adder, it also needs to consider a carry in and carry out flag. Thus a 1‐bit full adder takes three 1‐bit inputs and contains two 1‐bit outputs. The first two inputs are the two bits that are to be added together, respectively A and B. The third input is a carry in flag. This flag specifies whether or not a previous addition has occurred which contained a carry out. The first output is the 1‐bit result of the addition. The second output is the carry out flag which specifies if the result of the addition was larger than the 1‐bit result.

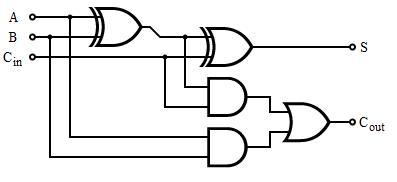


Fig 3.1 ( Full adder circuit using logic gates )

**3.2 How does a full adder work?**

The Boolean expression for a full adder are given as under:



The following figure shows the logic table for a full adder:

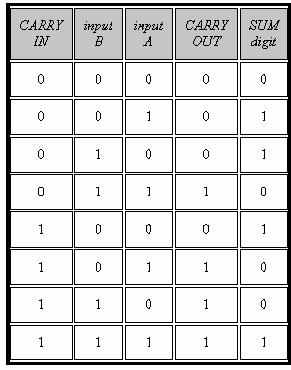


Fig 3.2 (Truth table of Full Adder )

**3.3 CMOS Full Adder**

For translating the gate-level design into a transistor-level circuit description, we note that both the sum..out and the carryout functions are represented by nested ANDOR-NOR structures in Fig 3.3. Each such combined structure (complex logic gate) can be realized in CMOS as follows: the AND terms are implemented by *series-connected* nMOS transistors, and the OR terms are implemented by *parallel-connected* nMOS transistors. The input variables are applied to the gates of the nMOS (and the complementary pMOS) transistors. Thus, the nMOS net may consist of nested series-parallel ,connections of nMOS transistors between the output node and the ground. Once the nMOS part of a complex CMOS logic gate is realized, the corresponding pMOS net, which is connected between the output node and the power supply, is obtained as the *dual* *network* of the nMOS net. [3]

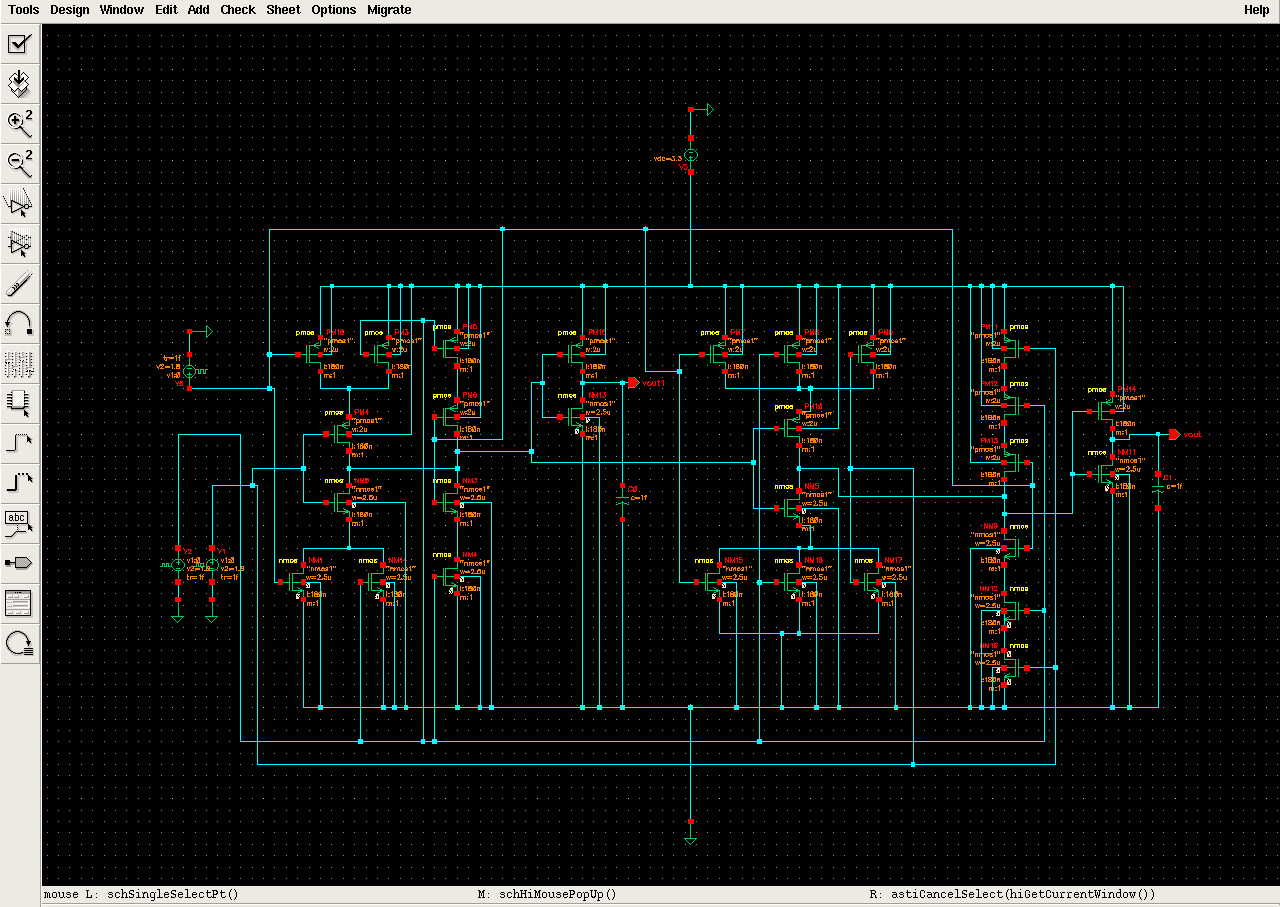


Fig 3.3 (1- bit CMOS Full Adder implemented in Cadance Virtuso-4 )

Fig 3.3 showsa basic CMOS full adder that was implemented using the cadence tool. The Full adder circuit utilized 28 transistors. Fourteen of them are NMOS and the rest of them PMOS so as to complete the CMOS inverter structure. Our Full adder circuit is shown in the figure3.3. This circuit is working at a supply voltage of 3.3V. The three inputs of circuit have been given a pulsating input. In order compensate the overall capacitance, Sum and Carry outputs have been connected with load capacitors. The logic table of a full adder is shown fig 3.2 .The fig 3.4 depicts the input and output wave when the load capacitance in negligible (1f F), The input A has a pulse width of 5ns and a time period of 10ns, The input B has a pulse width of 10ns and a time period of 20ns, The input C has a pulse width of 25ns and a time period of 40ns. The pulsating input given to A, B & C cover almost all the input combinations. The results prove that the full adder is working as it should.

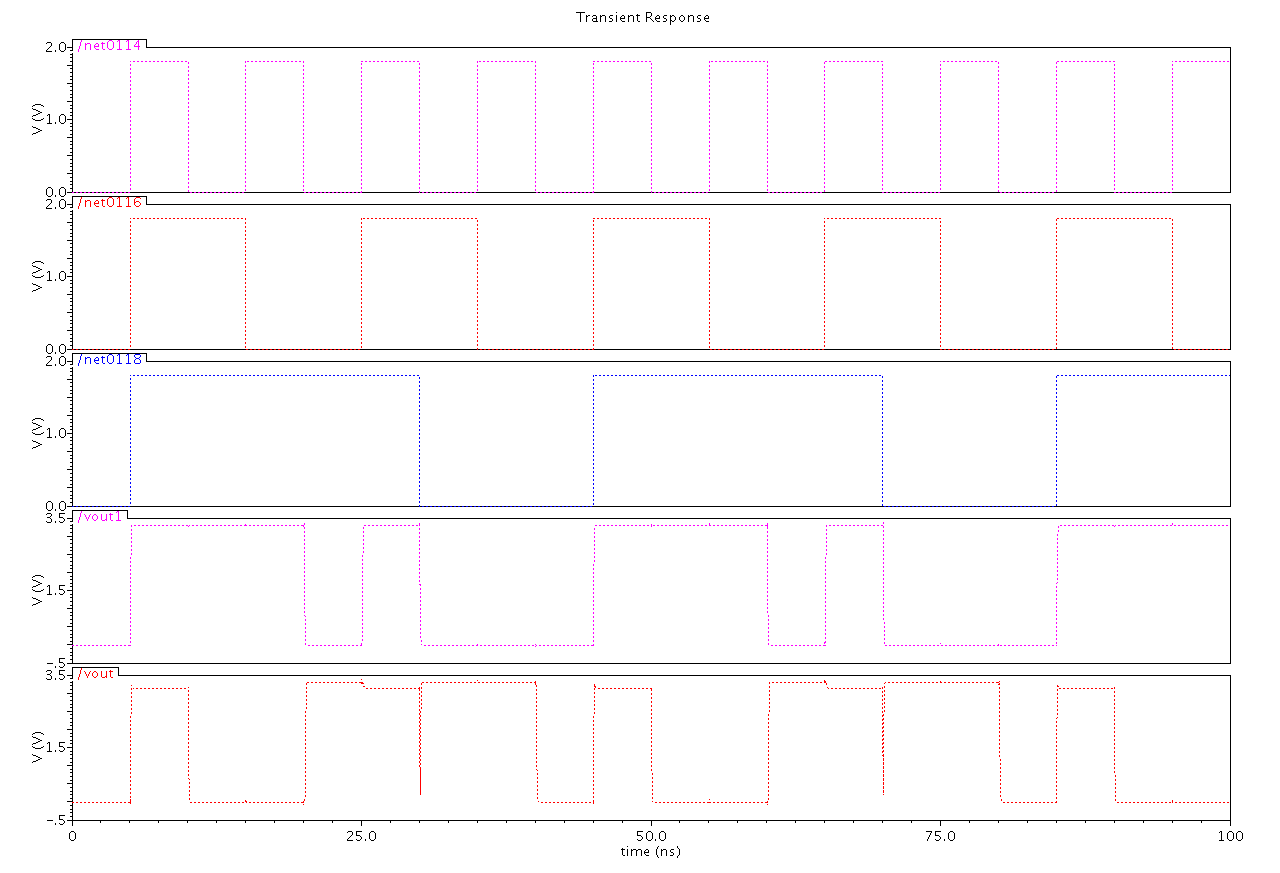
****

Fig 3.4 (Waveform of CMOS Full Adder)

**CHAPTER- 4**

**GENETIC ALGORITHM**

**4.1 Introduction to GA**

Genetic algorithm was developed in the year 1975 at University of Milligan by Prof. John Holland. Genetic algorithm and its variants are basically computational procedures which try to ape the natural phenomenon of evolution. Evolution is the phenomenon by which species may refine themselves over a number of generations in order to adapt to environment. Evolution was first studied by Darwin. He analyzed that only the fittest of every generation would survive, eventually adapting to the environment. Two parents reproduce to give new off springs. These offspring have chromosomes from both parents i.e have characteristics of both the parents. This is the general way that Genetic algorithm and its variants work to solve problems of adaptive systems and optimum solutions. Since Genetic algorithm is heuristic in nature they do not give us ideal solutions yet the solution given by them are reasonably good.

In Genetic algorithm each individual is assumed to have a fitness function associated with it, in order for an individual to be eligible for the next generation it must poses some minimum fitness value. For each generation individuals are selected from the existing population and crossed among each other to form the new and refined set of population.

Here we can categorize Genetic algorithm in to two types:

1. Simple Genetic algorithm.
2. Steady state Genetic algorithm.

Simple Genetic algorithm is the one in which new generation may completely replace the old individuals whereas in Steady State Genetic algorithm the new population may consist of a combination of old and new individuals. Decision for which one to use may differ from application to application. In either of the cases fitness value increases from one generation to another.



Fig 4.1 (Explanation of Genetic Algorithm )

**4.2 Two Basic Type of Genetic Algorithms**

**4.2.1 Simple Genetic algorithm**

The simple genetic algorithm utilizes three evolutionary operators viz: Selection, Crossover and Mutation. Each chromosome is an encoding of a solution to the problem at hand and each individual has an associated fitness which depends on the application. The fitness of the population increases with every evolving generation by selecting two individuals, crossing the parents and mutating the characteristics. In Genetic algorithm selection is done probabilistically but more towards highly fit individuals. Distinct generations are evolved and the process of selection crossover and mutation are repeated until all entries in a new generation are filled[2].

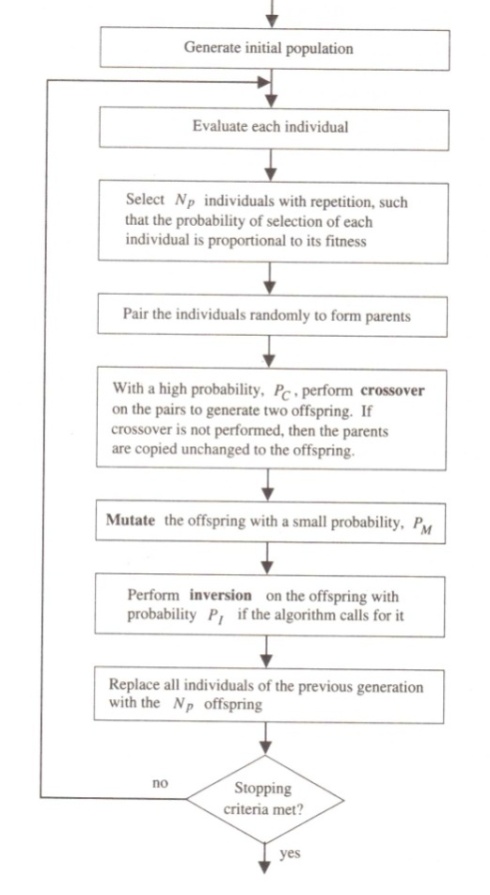


Fig 4.2 ( Flow chart of Simple Genetic Algorithm )

**4.2.2 Steady State Genetic Algorithm**

In a Genetic algorithm having overlapping generations only a fraction of the individuals are replaced in each generation. In each generation, two different individuals are selected as parents according to their fitness. Crossover is performed with a high probability to form crossover. The offspring are mutated with a low probability and may be inverted if necessary. A duplicate check may be performed to discard the duplicates if. The offspring are then evaluated and survive only if they are better than the worst member of the current population. In steady state genetic algorithm the generation gap is minimal, Since only two offspring are produced in each generation [2].

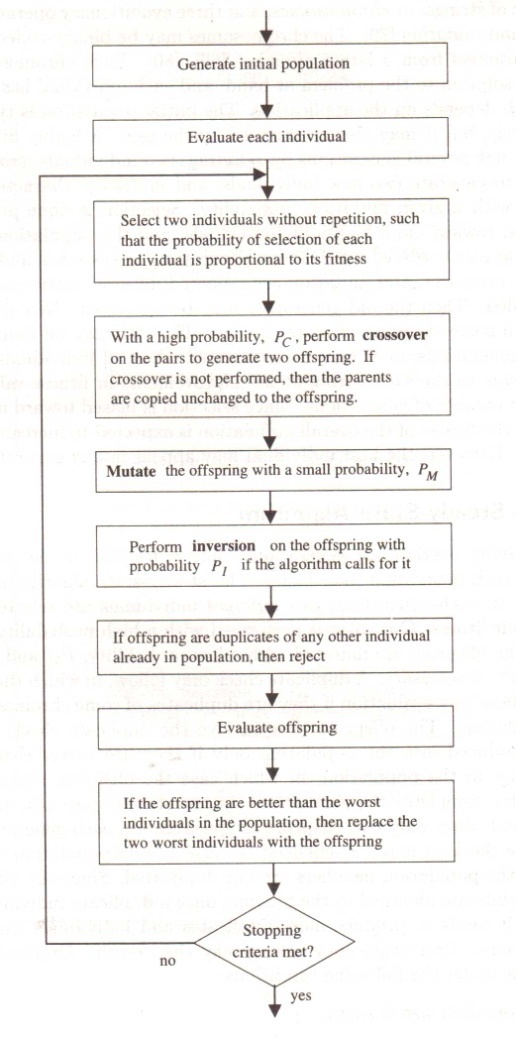


Fig 4.3 ( Flow chart of Steady State Genetic Algorithm )

**4.3 Basic Terminology of Genetic Algorithm**

**4.3.1 Chromosomes**

Central to all genetic algorithms is the concept of the chromosome. The chromosome contains all information necessary to describe an individual. In nature, chromosomes are composed of DNA. In a computer, a long binary or character string is used. Chromosomes are composed of genes for the various characteristics to be optimized and can be any length depending on the number of parameters to be optimized.

**4.3.2 Encoding**

Encoding defines the way genes are stored in the chromosome and translated to actual problem parameters. A possible encoding scheme for a hypothetical circuit using a 16 bit binary chromosome is shown below:



**4.3.3 Fitness**

Fitness is a single numerical quantity describing how well an individual meets predefined design objectives and constraints. Fitness can be computed based on the outputs of multiple analyses using a weighted sum.

**4.3.4 Crossover**

Crossover is a method of exchanging genetic material between two parents to produce offspring. The operation is simple, a cross-over point is chosen at random and the genetic information to the left of the cross-over point in parent A is combined with the genetic information to the right of the cross-over point in parent B to produce new offspring. A second offspring can be created by using the information from the right of parent A and the left of parent B. The operation is illustrated with the example below:



**4.3.5 Mutation**

Mutation in Genetic algorithm is some what similar to the genetic mutation.It is used to maintain genetic diversity between subsequent generations. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution or may be different only by a single bit.[17]

Mutation may be of the following types:

* Bit String mutation.
* Flip-Bit Mutation.
* Uniform Mutation.
* Non – Uniform Mutation.
* Gaussian Mutation.

**CHAPTER- 5**

**POWER DISSIPATION**

**5.1 Sources of Power Dissipation in CMOS**

**5.1.1 Static Power**

Taking in to consideration the figure given below [6]

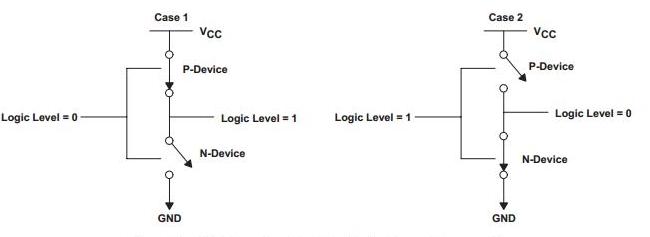


Fig 5.1 ( Explanation of Static Power )

We can see that if the input is at logic 0, the n-MOS device is OFF, and the p-MOS device is ON (Case 1). The output is logic 1. Similarly, when the input is at logic 1, the associated n-MOS device is biased ON and the p-MOS device is

OFF. The output is logic 0. Note that one of the transistors is always OFF. Since there is no current path between Vcc and GND hence, static power consumption is theoretically zero. However, there is a small amount of static power consumption due to reverse-bias leakage between diffused regions and the substrate which results in static power dissipation. The current flowing between Vcc and GND is known as the leakage current and can be described by the following equation [6]:

Ilkg =is(eqv/kt - 1)

Where

is = reverse saturation current

V = diode voltage

k = Boltzmann’s constant (1.38 × 10–23 J/K)

q = electronic charge (1.602 × 10–19 C)

T = temperature

The total static power can be calculated as under

Static power = ∑ (Leakage Current) x (Supply Voltage)

**5.1.2 Dynamic power**

There are two types of power dissipation in cmos circuits: dynamic and static. Dynamic power dissipation is caused by switching activities of the circuits. A higher operating frequency leads to more frequent switching activities in the circuits and results in increased power dissipation. The most significant source of dynamic power dissipation in cmos circuits is the charging and discharging of capacitance.

Static power dissipation is related to logic states of the circuits rather than switching activities. In cmos logic, leakage current is the only source of static power dissipation**.**

Fig 5.2 shows the equivalent circuit of charging and discharging output capacitance of a cmos logic gate.



Fig 5.2 ( Equivalent circuit of charging & discharging )

We use a switch that flips up and down to model the charging and discharging cycles. Where V is an ideal constant voltage source and Rc (Rd) is the resistance of a charging (discharging) circuitry. According to laws of physics, the voltage vc(t) and the current ic(t) of a capacitance cL at time t are given by

ic(t) = cLdvc(t)/dt

during the charging cycle from time to to t1, the enegy Es drawn from the voltage source is

Es =  dt

Initially capacitor contains no charge and the voltage across its terminals is zero , i.e., vc(t0) = 0. Assume that the capacitor is fully charged at the end of the charging cycle, we have vc(t1) = V. So from above two equations we have

Es = CL V = CL V

Es= CL V2

Part of the electrical energy Es drawn from the voltage source is stored in the capacitance and the rest is dissipated as heat energy in the resistor Rc . The energy Ecap stored in the capacitor at the end of the charging cycle is

Ecap  =

Ecap = CL

Ecap = CL

Ecap = 1/2 CLV2

The energy Ec dissipated at Rc during charging is therefore

Ec = Es – Ecap = 1/2 CLV2

Now consider the discharging cycle from t1 to t2 , we assume that the capacitor is fully discharged, i.e., vc(t1) = V and vc(t2) = 0. The energy Ed  dissipated in the discharge resistor Rd is

Ed = -

Ed = - CL

Ed = 1/2CLV2

Ed  is exactly equal to the energy stored in the capacitance at the beginning of the discharging cycle. If we charge and discharge the capacitance at the frequency of f cycles per seconds, the power dissipation of the system is

P = Esf = CLV2f

Above equation is the most important power dissipation equation in digital VLSI chip design. It relates power dissipation to a few quantities that are readily observable and measurable in VLSI circuits. In general , the total power should be summed over each capacitance Ci in a circuit yielding

**P = ∑I Ci Vi2 f**

Where Vi is the voltage swing across the capacitor Ci switching at frequency fi . For CMOS circuits, V is typically the same for all capacitance Ci.

To calculate the dynamic power we use the formula for constraint is

Function [c, ceq] = simple\_constraint(x) R=1.1; r=0.23; Vdd=3.3;

c = [(Vdd\*(1-(exp(-((x(2))/(R\*x(1)))))))-3.3;

Vdd\*(exp(-((x(2))/(r\*x(1)))))+0];

ceq = [ ];

where x(1) is capacitance and x(2) is frequency

Fitness function used to calculate the power from genetic algorithm is

Function y = simple\_fitness(x)

y = 10.89 \* (x(1)/ x(2));

**CHAPTER – 6**

**TOOLS USED**

**6.1 Cadence**

Cadence is an Electronic Design Automation (EDA) environment that allows integrating in a single framework different applications and tools (both proprietary and from other vendors), allowing to support all the stages of IC design and verification from a single environment. These tools are completely general, supporting different fabrication technologies. When a particular technology is selected, a set of configuration and technology-related files are employed for customizing the Cadence environment. The Fig 6.1 shows the design flow in Cadence development tool. We first make a schematic of the circuit according to the specifications, then the circuit is simulated to check the correctness of working then the layout of the circuit is made and verified. Finally the layout is simulated again to check for last moment errors and the design is then sent for fabrication.[11]

##### **6.1.1 Features of Cadence Design Environment**

* Reduced learning curve with a simulator-independent environment
* Maximum efficiency in the script-driven mode
* Accelerated debug process using a variety of built-in analog analysis tools
* Facilitated design correction via easy comparison of pre- and post-parasitic extracted designs
* Quick detection of circuit problems via a clear visualization cockpit

****

Fig 6.1( Design flow in Cadence Design Environment )

**6.1.2 Cadence Virtuoso**

Cadence Virtuoso is integrated design platform targeted for both digital and analog applications. It includes multiple physical verification tools, such as Design Rules Check (DRC), Layout vs. Schematic (LVS), Electrical Rules Check (ERC), Layout Parasitic Extraction (LPE) and Abstract Generation. In order to invoke Virtuoso, additional files are required. New technology file (tech65n.tf) with process dependent layout information is used to create new library. The graphical display template (display.drf) is used to display the different layer color and stipple pattern.[12]

**6.2 MATLAB**

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. For a million engineers and scientists in industry and academia, MATLAB is the language of technical computing. [7]

**6.2.1 Key Features of MATLAB :**

* High-level language for technical computing
* Development environment for managing code, files, and data
* Interactive tools for iterative exploration, design, and problem solving
* Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
* 2-D and 3-D graphics functions for visualizing data
* Tools for building custom graphical user interfaces
* Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java™, COM, and Microsoft® Excel®

## 6.2.2 Developing Algorithms using MATLAB

MATLAB provides a high-level language and development tools that let you quickly develop and analyze your algorithms and applications. The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution. With the MATLAB language, you can program and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for ‘for’ loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code. At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, [object-oriented programming](http://www.mathworks.in/products/matlab/object_oriented_programming.html) (OOP), and debugging features.

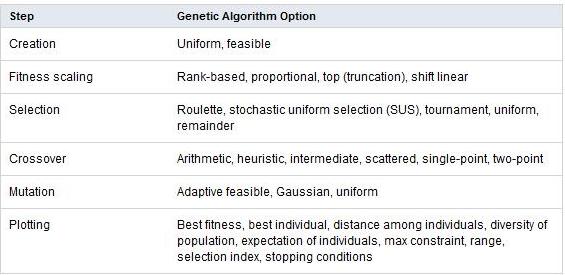
# 6.2.3 Global Optimization Toolbox

The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution. With the MATLAB language, you can program and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for ‘for’ loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, [object-oriented programming](http://www.mathworks.in/products/matlab/object_oriented_programming.html) (OOP), and debugging features.

## 6.2.4 Genetic Algorithm Solver

The [genetic algorithm](http://www.mathworks.in/discovery/genetic-algorithm.html) solves optimization problems by mimicking the principles of biological evolution, repeatedly modifying a population of individual points using rules modeled on gene combinations in biological reproduction. Due to its random nature, the genetic algorithm improves your chances of finding a global solution. It enables the user to solve unconstrained, bound-constrained, and general optimization problems, and it does not require the functions to be differentiable or continuous.[7]



Global Optimization Toolbox also lets you specify:

* Population size
* Number of elite children
* Crossover fraction
* Migration among subpopulations (using ring topology)
* Bounds, linear, and nonlinear constraints for an optimization problem

The figure 6.2 shows empty genetic algorithm toolbox:

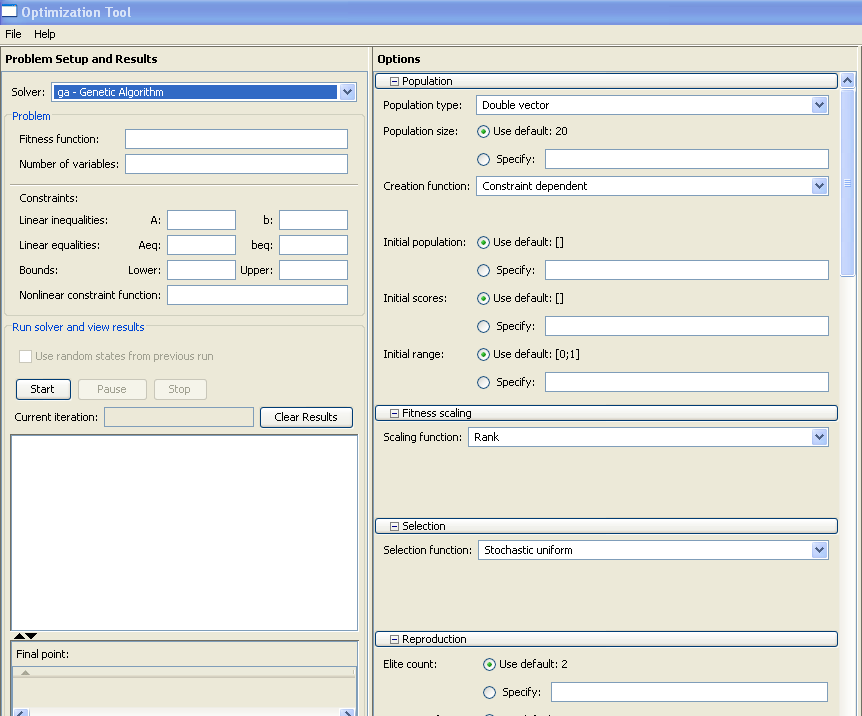


Fig 6.2 ( MATLAB Genetic toolbox )

The user can customize these algorithm options by providing user-defined functions and represent the problem in a variety of data formats, for example by defining variables that are integers, mixed integers, categorical, or complex.

The user can base the stopping criteria for the algorithm on time, stalling, fitness limit, or number of generations. And you can vectorize your fitness function to improve execution speed or execute the objective and constraint functions in parallel (using Parallel Computing Toolbox).

**CHAPTER- 7**

**POWER OPTIMIZATION USING GENETIC ALGORITHM**

**7.1 Introduction to Work**

Every aspect of our life may have a margin for optimization. Every situation that confront us may not be ideal and may require some changes. For example In the morning we try to maximize our time to sleep, In the market we try to maximize our purchase with the minimum amount of money. There is a scope of optimization in every daily life problem so as to maximize the results by utilizing our limited resources. Optimization may be applied manually so as to solve to daily life problems but we may face problem in solving complex technical issues manually. Automated techniques such as Ant colony, Simulated annealing, Lagrangian multiplication, Bacterial foraging algorithm have already being developed and are being implemented to solve complex optimization problems.

Genetic algorithm is one such technique that is based on the natural process of evolution. Darwin studied that a species may adapt itself one step closer to the environment with every upcoming generation. Only the fittest member of a particular generation will survive and reproduce to pass their characteristics to the next generation. While using genetic algorithm to solve complex search problems an initial population of chromosomes may be considered, each chromosome has a different fitness value. Only a few having the required fitness value are selected to act as parents. Two parents may cross among themselves to produce an offspring. All the offspring together form the new population.

Here the genetic algorithm is being used to optimize the dynamic power dissipation in VLSI circuits. We very well know that dynamic power dissipation dependent on three factors viz. load capacitance, frequency and supply voltage. We expect the genetic algorithm to find us the minimum value of every parameter for which the output is well within the acceptable limits. Here we confronted by the challenge of reducing the power dissipation at the cost of lower switching activity which can be considered as a decrease in the frequency of operation. Moreover when we try to reduce the supply voltage below a particular value it may harm the working of our logic circuit.

The circuit under consideration is a simple one bit cmos full adder. It has three input (A, B & Cin ) and two output sum and carry\_out. In the three inputs two inputs A & B are the input variables (addend bits) and the third input Cin is the carry\_in. It performs binary addition between three inputs and generates corresponding sum and carry. Here we have used transistor level one bit full adder. The transistor level design of the CMOS full adder circuit. The circuit contains a total of 14 nMOS and 14 pMOS transistors, together with the two CMOS inverters which are used to generate the outputs.[3]

**7.2 Procedure Followed**

* Circuit schematic was implemented using cadence virtuoso. The full adder circuit has 28 transistors of which 14 are pmos and 14 are nmos including two cmos inverter. By default the value of length and width are 2u m and 180n m.
* Deep analysis of the characteristics of the circuit was done under different values of capacitance. A range of maximum and minimum values of capacitance were found for which the output voltage levels are well within the acceptable range. While finding the acceptable range for capacitance, the pulse width and time period of inputs A, B and C were 5ns and 10ns, 10ns and 20ns & 25ns and 4ons respectively.
* After finding the suitable range for the capacitance, load capacitance was fixed to 1f F. now the characteristics were analyzed for different values of frequencies. This time a suitable range of values was found for frequencies.
* The genetic tool box of MATLAB was used to implement the genetic algorithm. The formula for dynamic power was used as the fitness function. The equation for charging and discharging was used as a constraint. The upper and lower values of capacitance and frequency were used to define the maximum and minimum values of c and f, which are the variables in our constraint equation. The final results were thus obtained.

**CHAPTER - 8**

**SIMULATIONS RESULTS**

**8.1 One Bit CMOS Full Adder**

The schematic of CMOS full adder was implemented using Cadence design environment: Fig 8.1 showsa basic CMOS full adder that was implemented using the cadence. The Full adder circuit utilized 28 transistors. Fourteen of them are NMOS and the rest of them PMOS so as to complete the CMOS inverter structure. Our Full adder circuit is shown in the adjoining figure. This circuit is working at a supply voltage of 3.3V. The three inputs of circuit have been given a pulsating input. In order compensate the overall capacitance, Sum and Carry outputs have been connected with load capacitors. The logic table of a full adder is shown Fig 3.2

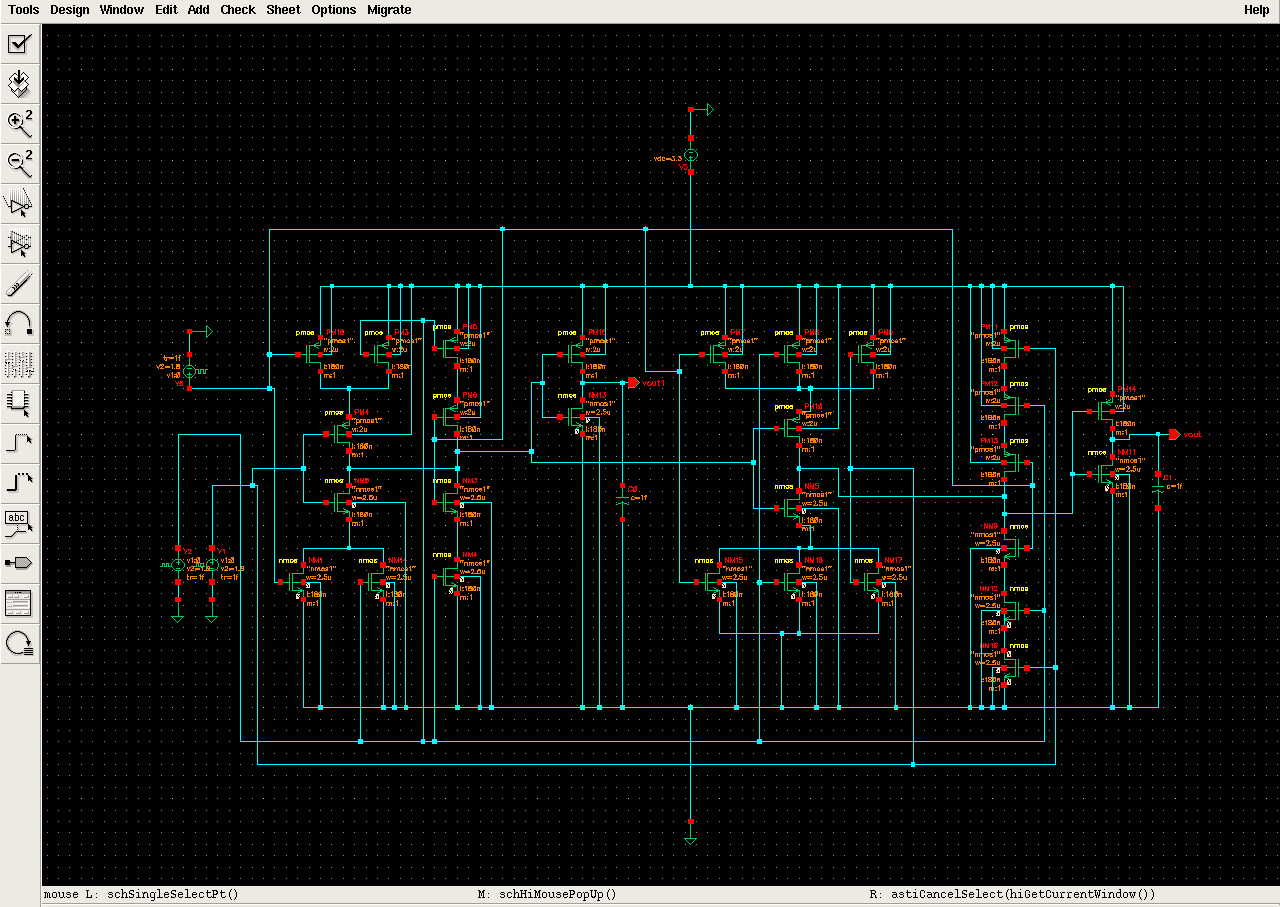


Fig 8.1 (1- bit CMOS Full Adder implemented in Cadance Design Environment )

The following wave form depicts the input and output wave when the load capacitance in negligible (1f F), The input A has a pulse width of 5ns and a time period of 10ns, The input B has a pulse width of 10ns and a time period of 20ns, The input C has a pulse width of 25ns and a time period of 40ns. The pulsating inputs given to A, B & C cover almost all input combinations. The results prove that the full adder is working as it should.

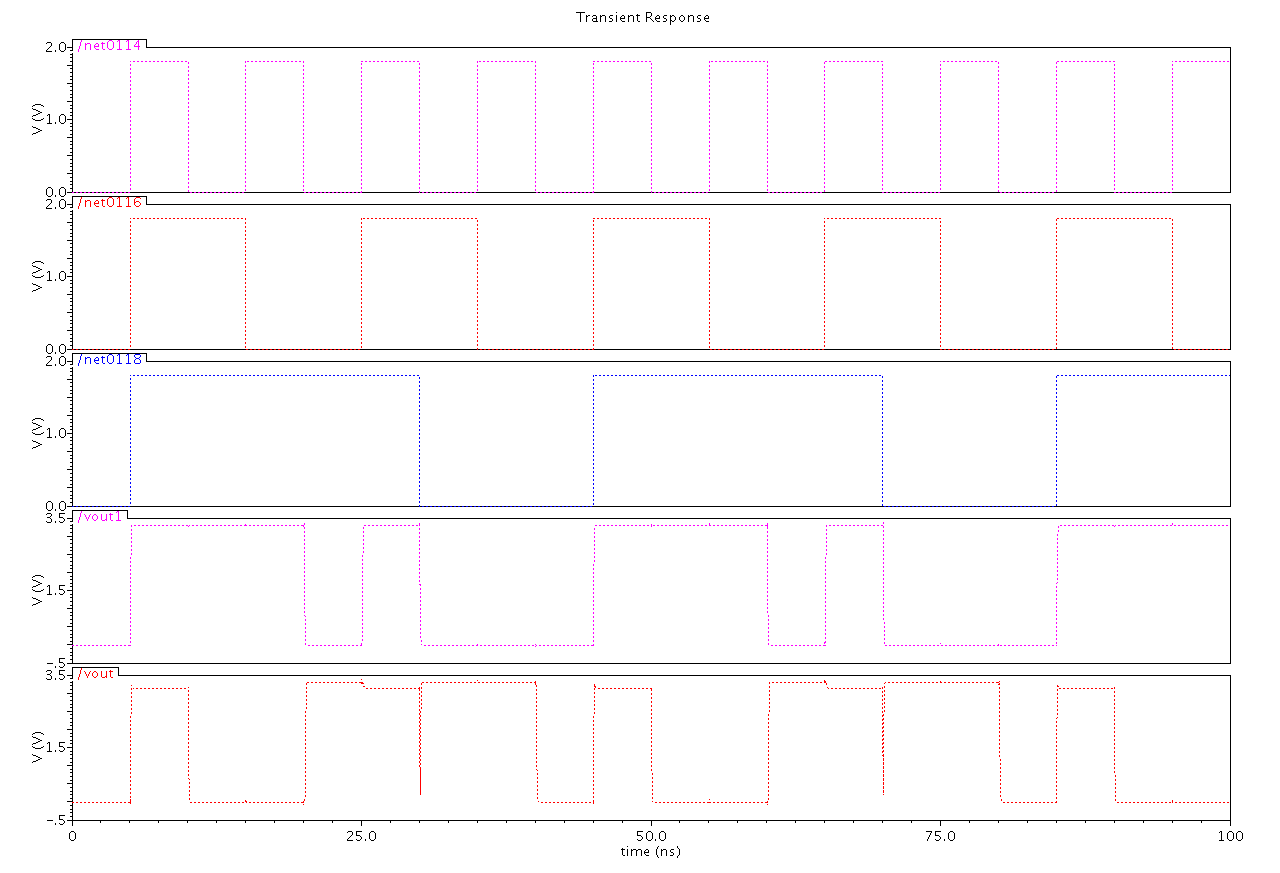
****

Fig 8.2 (Waveform of CMOS Full Adder)

**8.2 Output Waveform for carry, when capacitance = 1pF**

After validating the working of the full adder circuit, we now check the waveforms when the circuit has a load capacitance of one pico farad. We can see that charging and discharging of the capacitor has started affecting the output wave forms. The maximum output of the carry wave form is 3.237 v which is almost close to the supply voltage corresponding to logic ‘1’ of 3.3 volt.

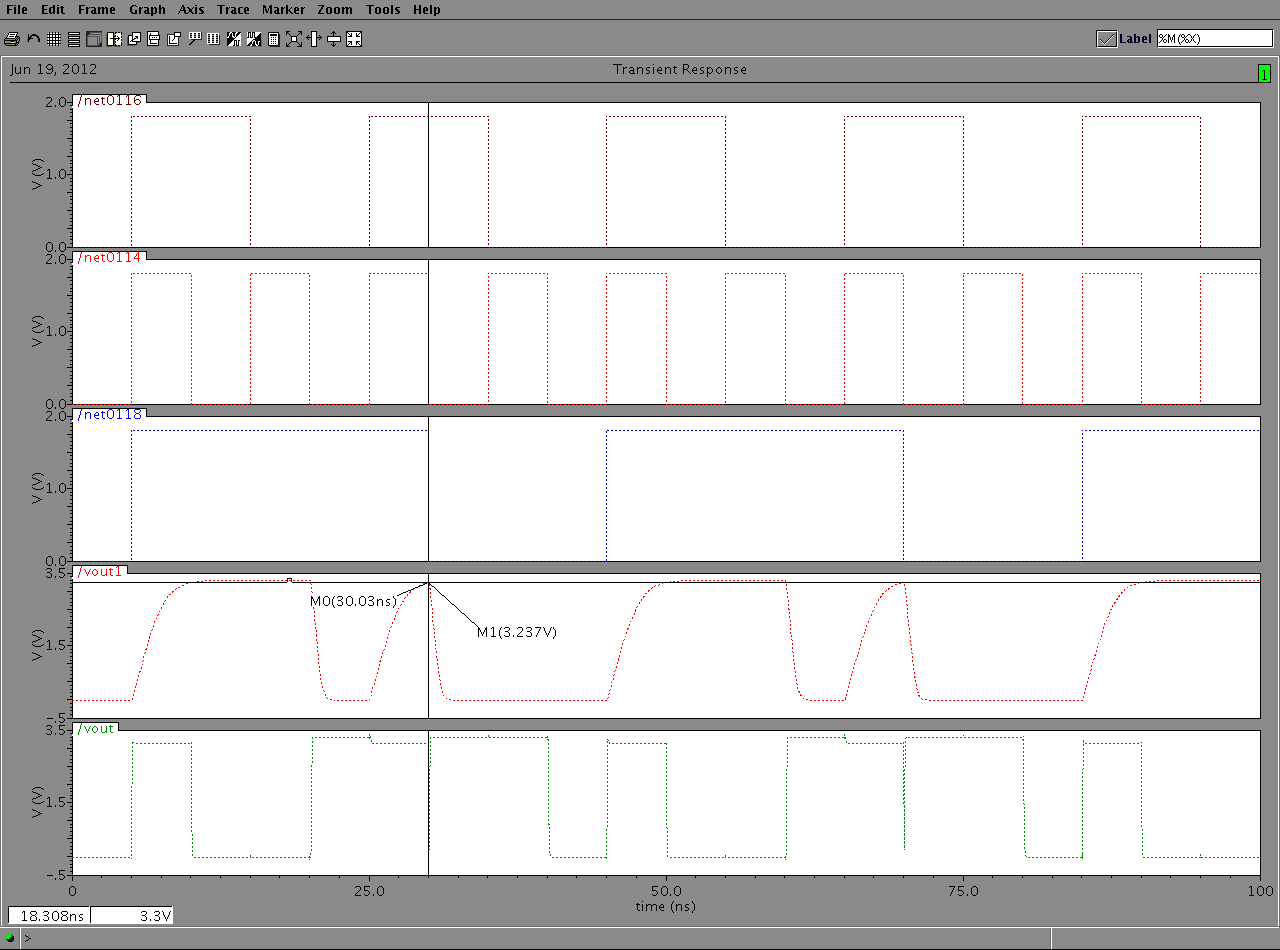


Fig 8.3 (Carry, c = 1pF)

We can see that the output completely obeys the logical working of full adder. The given figure depicts three inputs viz. A, B & C which are named as net 0116, net o114 & net 0118 respectively sum and carry are named as Vout and Vout1. In the output waveform fig 8.3 we have used horizontal and vertical marker lines to show the level of the output at particular point

**8.3 Output waveform for sum, when capacitance = 1pF**

The figure 8.4 shows the output waveforms when a 1 pico farad capacitor is connected at the output of the sum. Discharging and charging of the capacitor has started showing effect the maximum value of the sum output is 3.201v. some of the logic 1’s have started deteriorating and have a value 2.928v which is some what lesser than 3.3v.

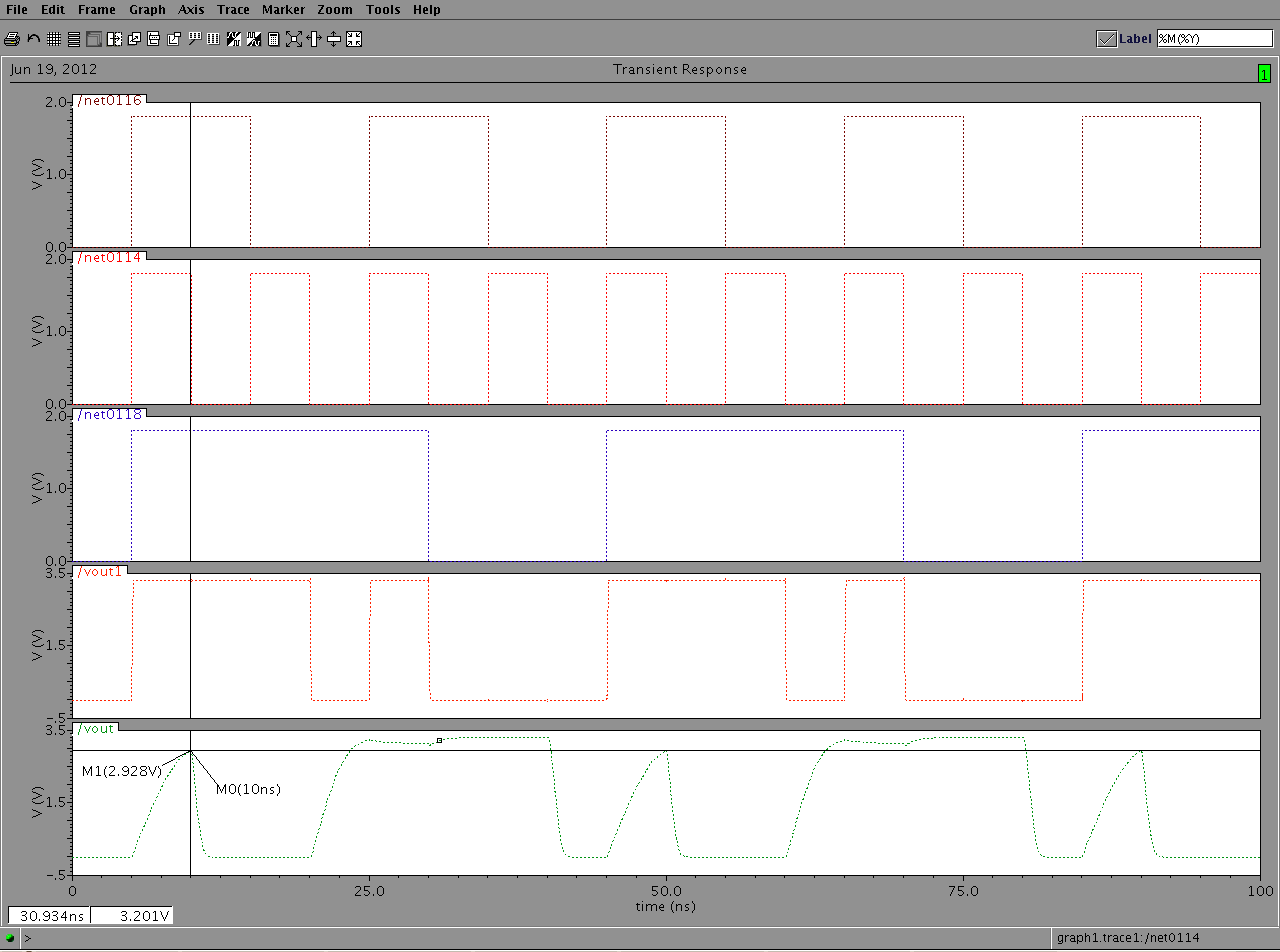


Fig 8.4 (Sum, c = 1pF )

The figure 8.4 depicts three inputs viz. A, B & C. In the output waveform we have used horizontal and vertical marker lines to show the level of the output at particular point. Different waveforms have been shown by different colors which is the property of the tool that we are using for our work.

**8.4 Output waveform for carry, when capacitance = 3.5 pF**

As we increase the capacitance the output voltage level starts decreasing. We assume that the outputs with a voltage level of more than 1.65v (50% level of supply voltage) are correct and can be considered as logic ‘1’. Whereas the effect on logic ‘0’ is almost negligible and remains unchanged.

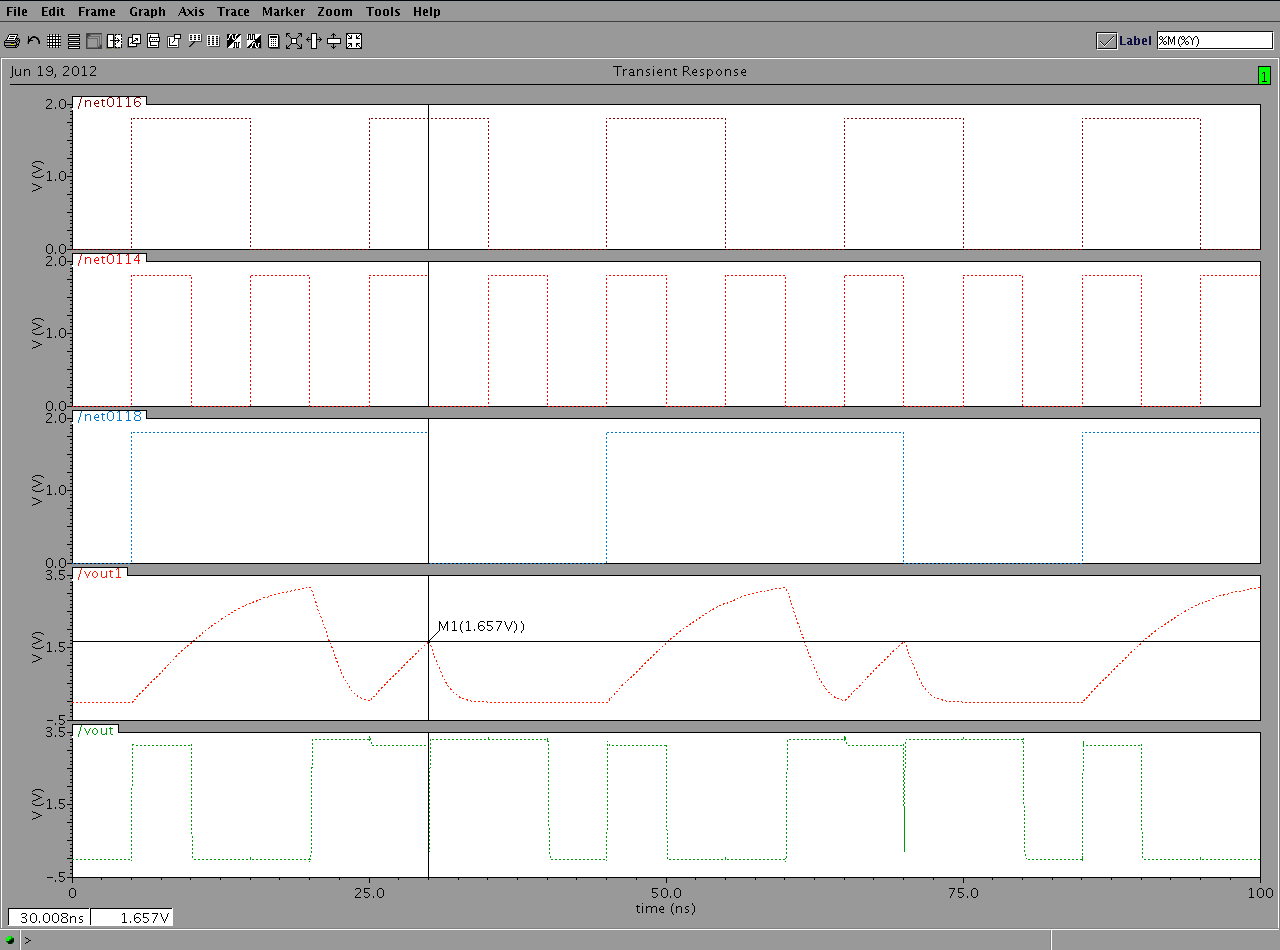


Fig 8.5 (Carry, c = 3.5 pF )

Here we see that a logic ’1’ on the carry is at the lowest acceptable level that is 1.65v. we can conclude that the output voltage at carry is well within the acceptable limits when the value of capacitance is between 1pf to 2.3 pf.

**8.5 Output waveform for sum, when capacitance = 2.3 pF**

As we increase the capacitance the output voltage level starts decreasing. We assume that the outputs with a voltage level of more than 1.65v (50% level of supply voltage) are correct and can be considered as logic ‘1’. Whereas the effects on logic ‘0’ is almost negligible and remains unchanged.

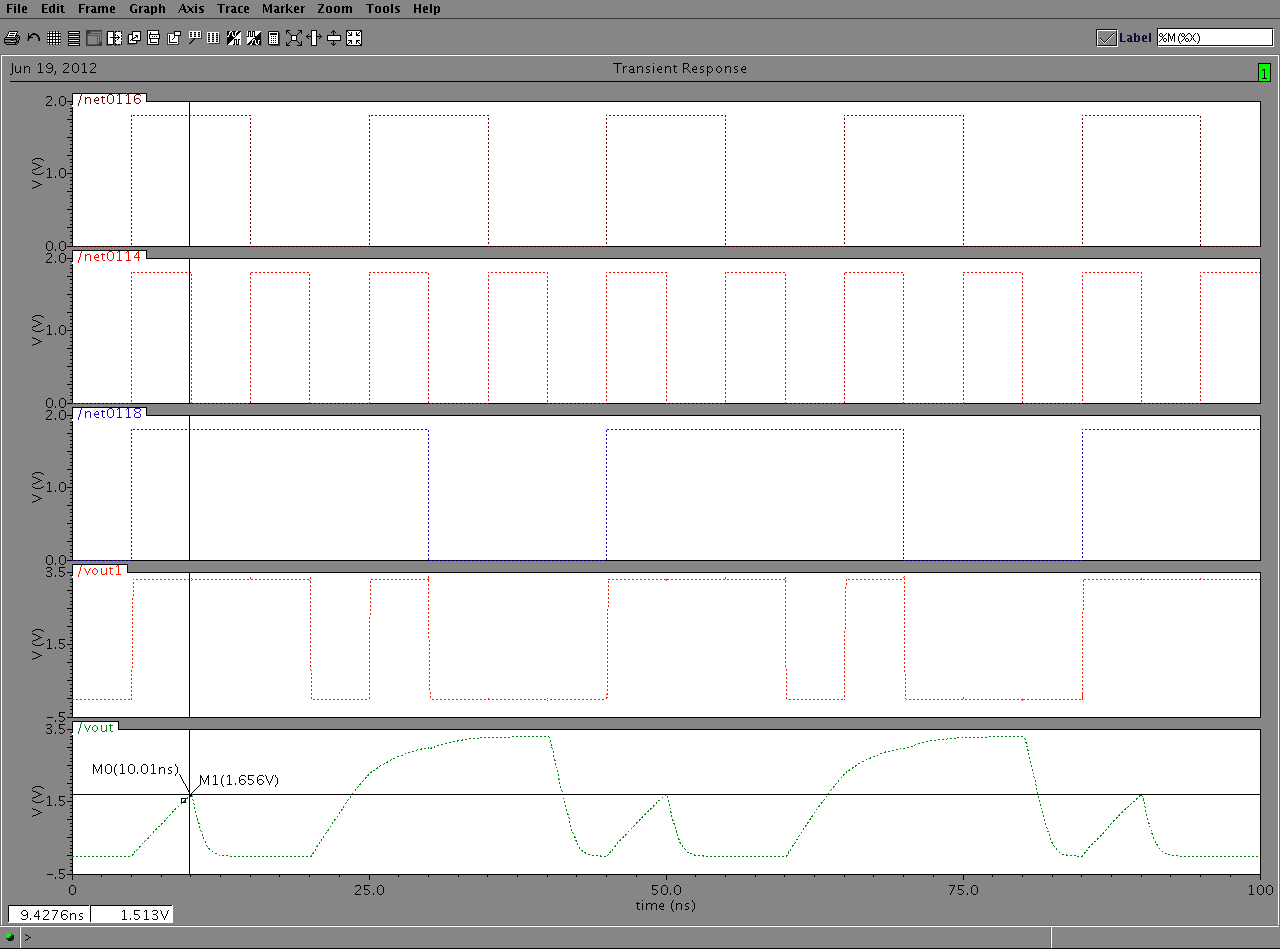


Fig 8.6 (Sum, c = 2.3 pF )

Here we see that logic ’1’ on the sum is at the lowest acceptable level that is 1.65v. we can conclude that the output voltage at sum is well within the acceptable limits when the value of capacitance is between 1pf to 2.3 pf.

**8.6 Parametric analysis of carry**

The figure 8.7 shows the parametric analysis of capacitor for carry . Different waveforms corresponds to outputs at different value of capacitance. As we increase the load capacitance the output levels start decreasing. The analysis has been done for the capacitance values starting from 1pf and going up to 3.5 p. This range corresponds to the capacitance values for which the logic level ’1’Can be easily recognize as logic level one (value should be above 1.65 v.

**8.7 Parametric analysis of sum**

The figure 8.8 shows the parametric analysis of capacitor for sum. Different waveforms corresponds to outputs at different value of capacitance. As we increase the load capacitance the output levels start decreasing. The analysis has been done for the capacitance values starting from 1pf and going up to 2.3 pf. This range corresponds to the capacitance values for which the logic level ’1’ can be easily recognized as logic level one (value should be above 1.65 v).

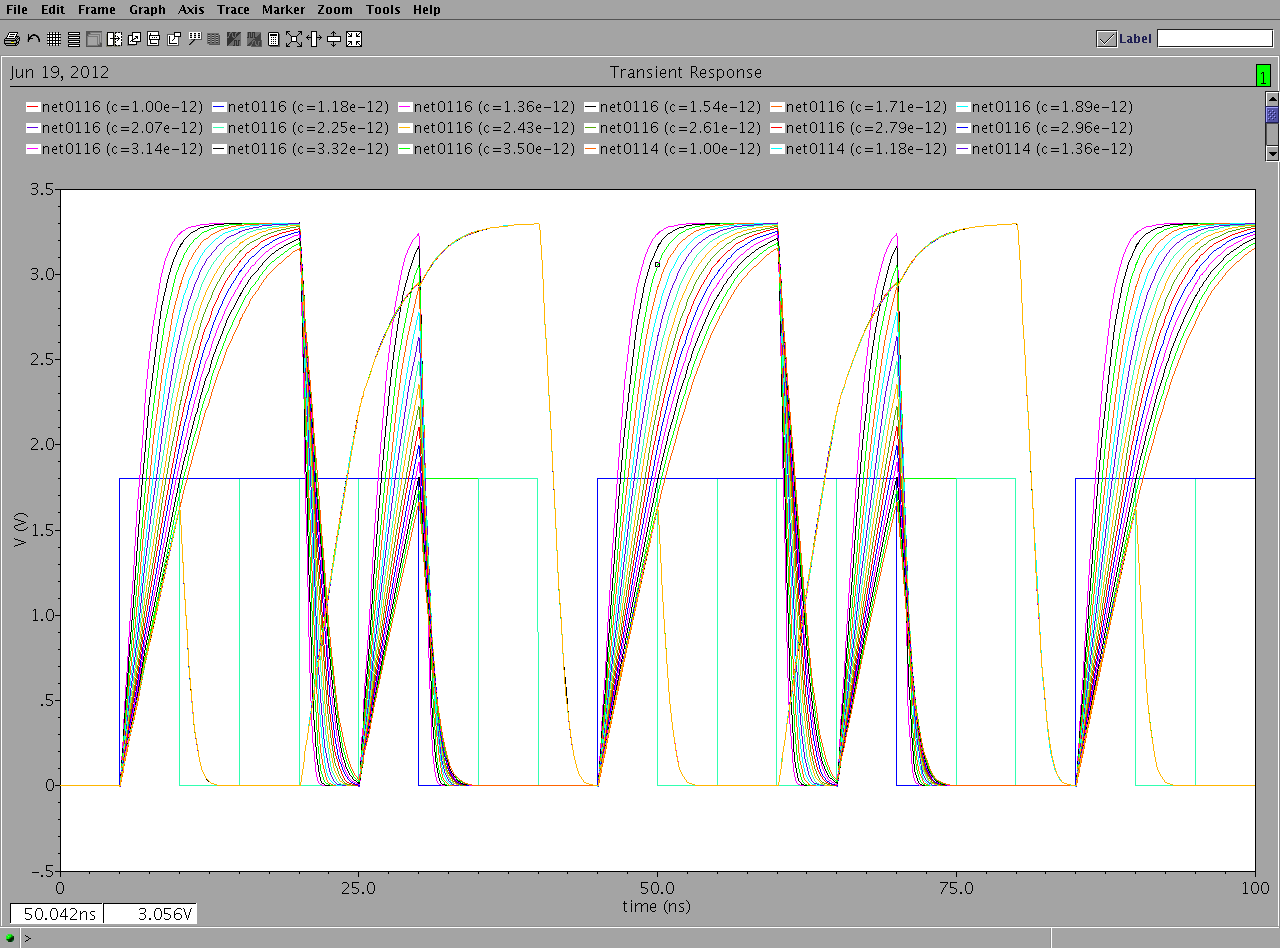


Fig 8.7 (Parametric Analysis of Carry )

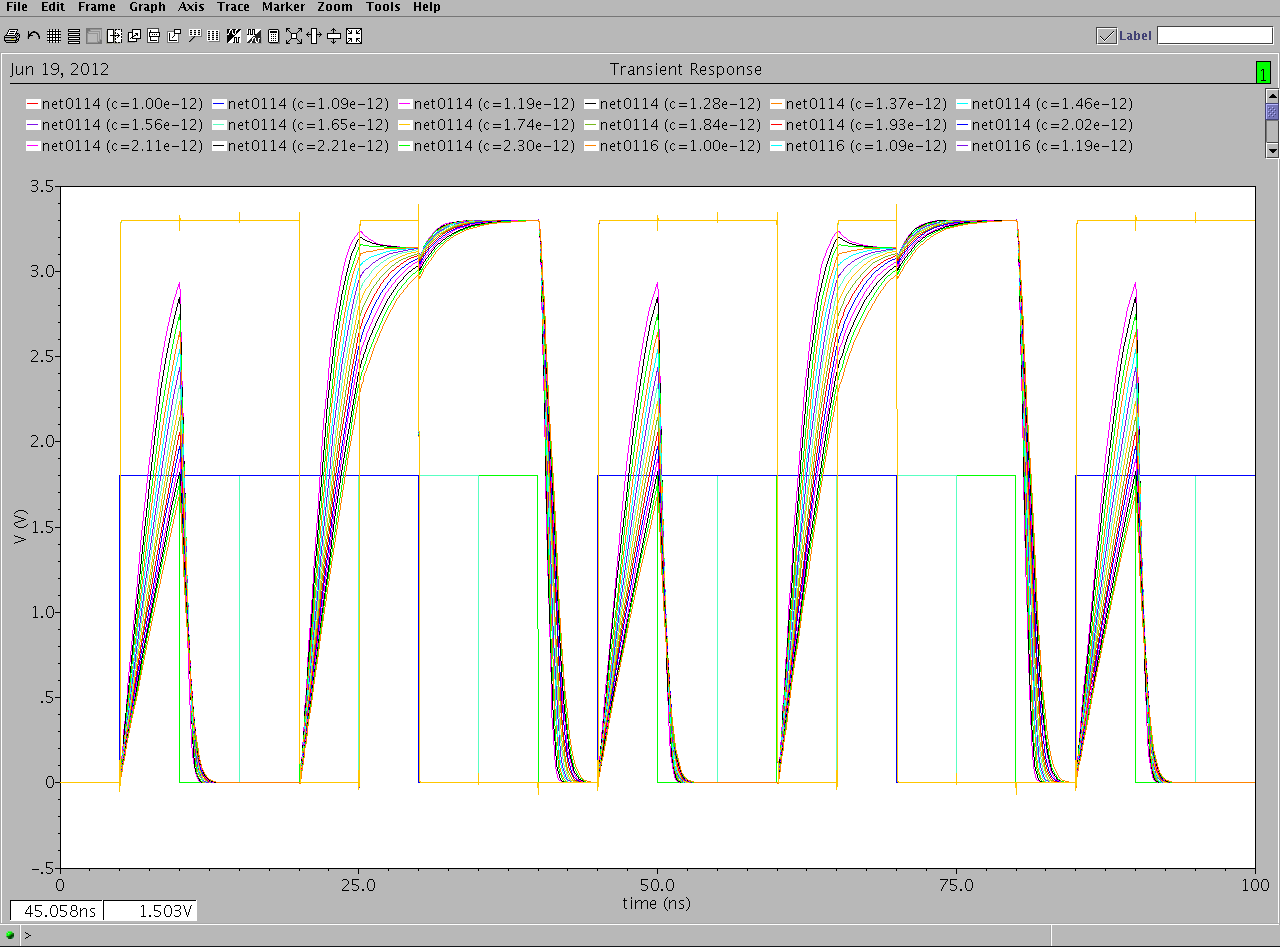


Fig 8.8 (Parametric Analysis of Carry )

**8.8 Output waveform for capacitance above 2.3 pF for sum & 3.5 pF for carry**

Fig 8.9 and Fig 8.10 show that when we increase the value of capacitance above 2.3 pF & 3.5 pF for sum and carry respectively the amplitude of the output wave decreases further and falls down the 1.65 volt level. These pulses are not good enough to be considered as logic ‘1’ (since they are below the 50% level). If we further increase the capacitance the voltage level of these pulses will decrease further and the full adder circuit may stop working according to the correct output levels.

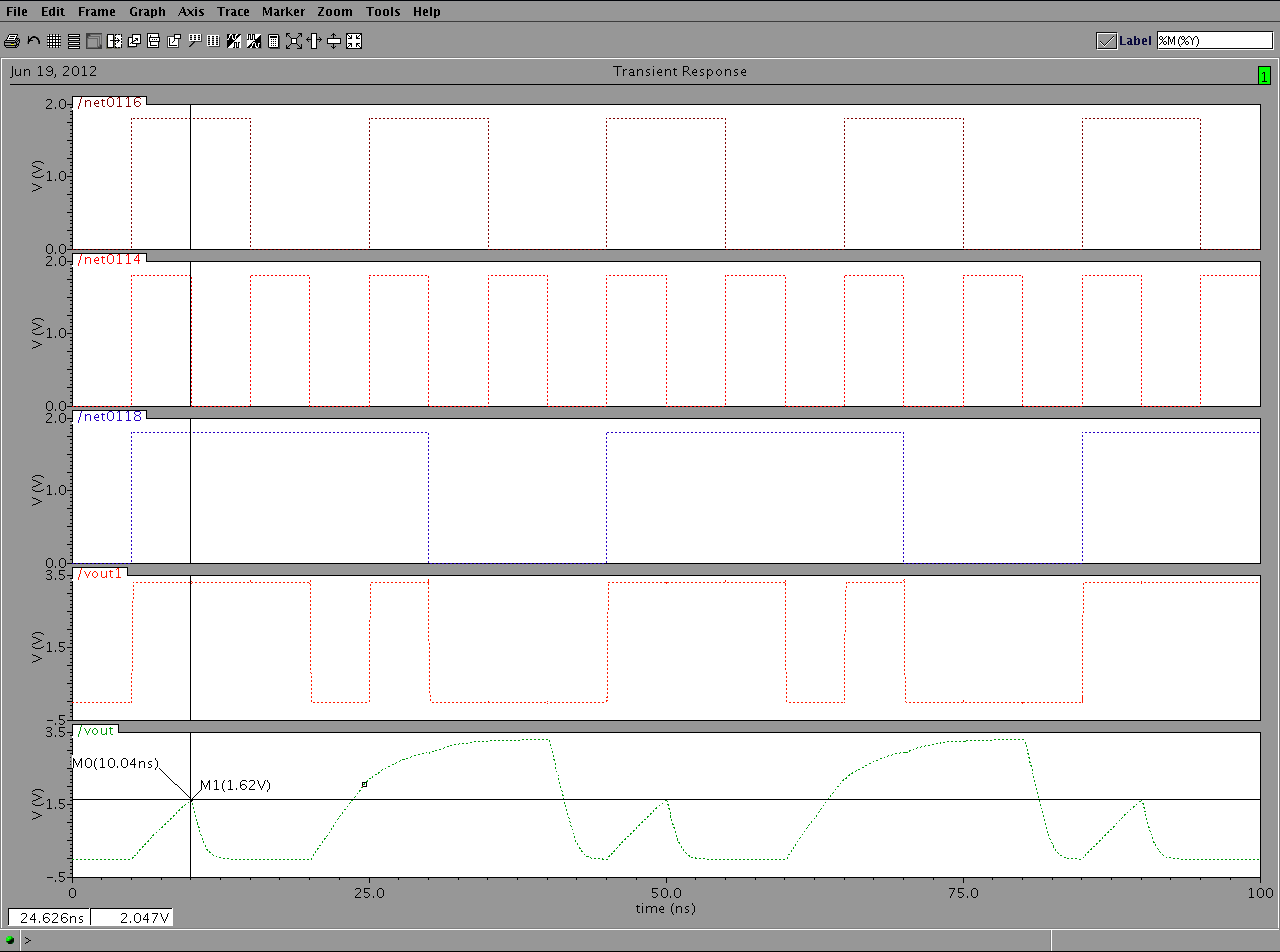


Fig 8.9 ( Sum, c > 2.3 pF )

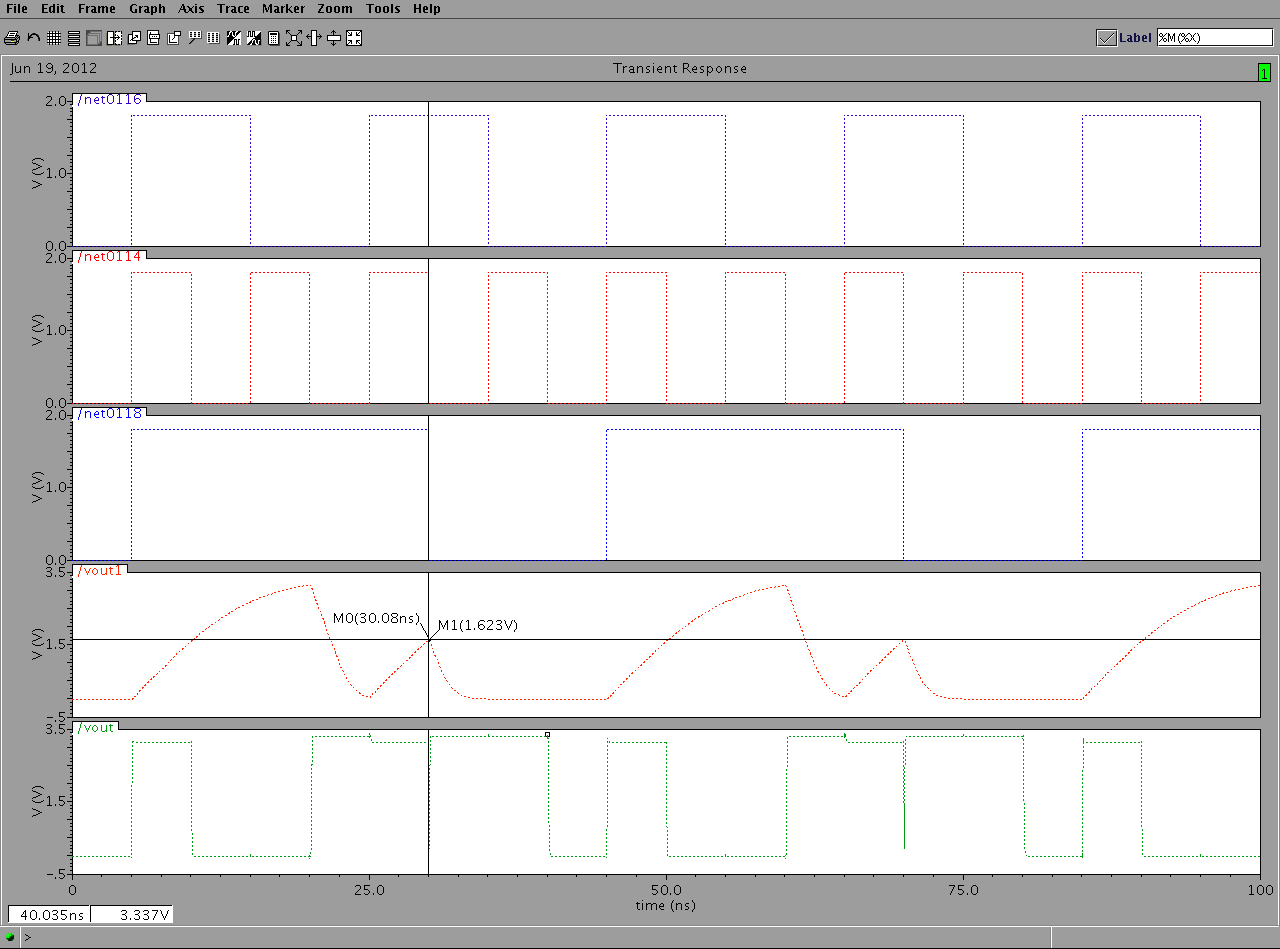


Fig 8.10 ( Carry, c > 3.5 pF )

**8.9 Output waveforms when sum and carry have the same capacitance**

Figure 8.11 to 8.13 show the results when the load capacitance at the output of sum and carry are varied simultaneously, the figure 8.11 shows the situation when the load capacitance for sum and carry is 1pF. We can see that the output is well above the 50% voltage level and almost touches the 3.3v. The figure 8.12 shows the case when the capacitance at the output of sum and carry is 2.355pF which is the above limits of the capacitor. At this time the output is just above the half value of the required voltage. The figure 8.13 shows the parametric analysis of capacitance when the value of capacitance varied from 1pF to 2.355pF.

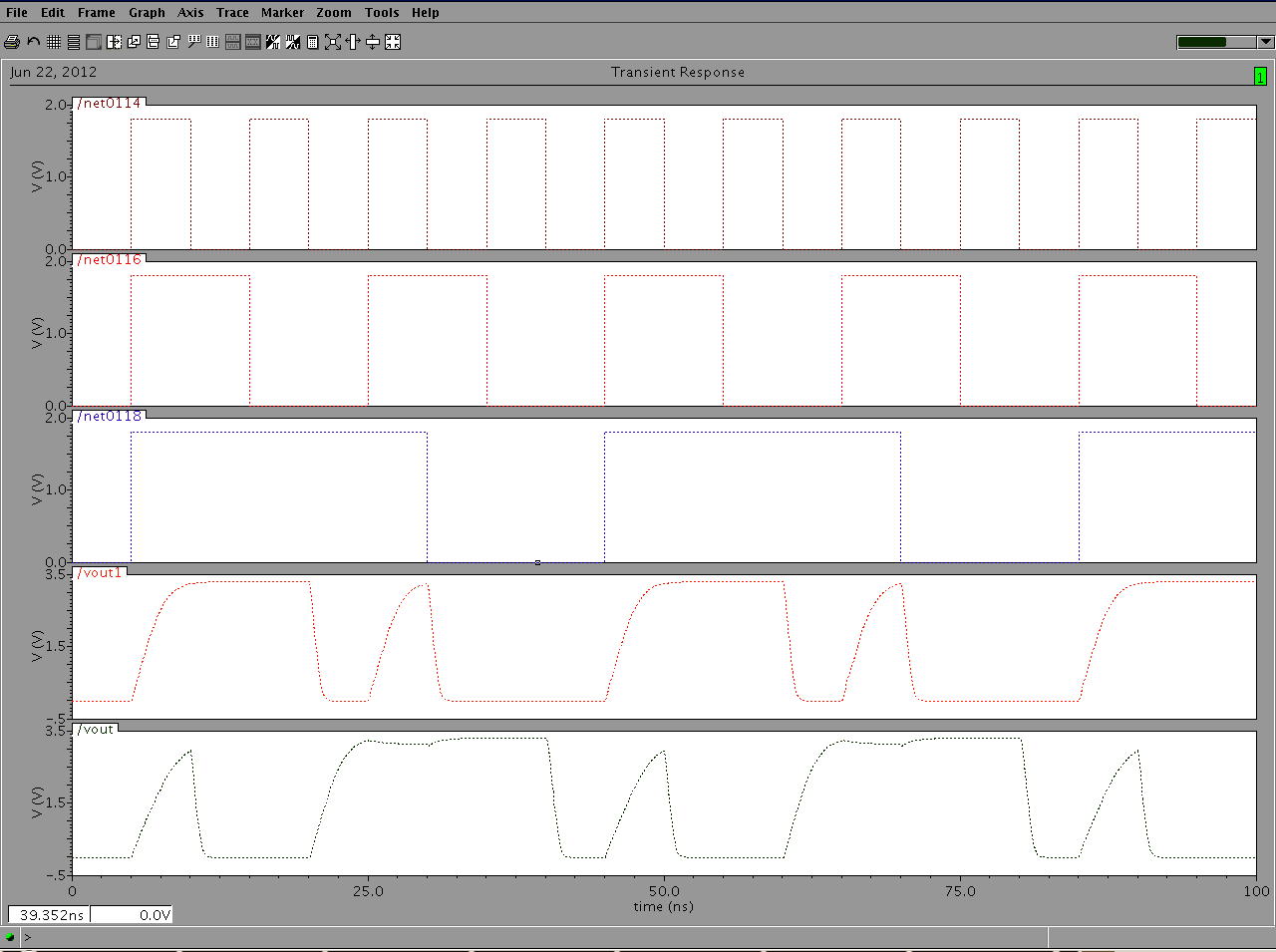


Fig 8.11 ( Sum and Carry when c =1 pF )

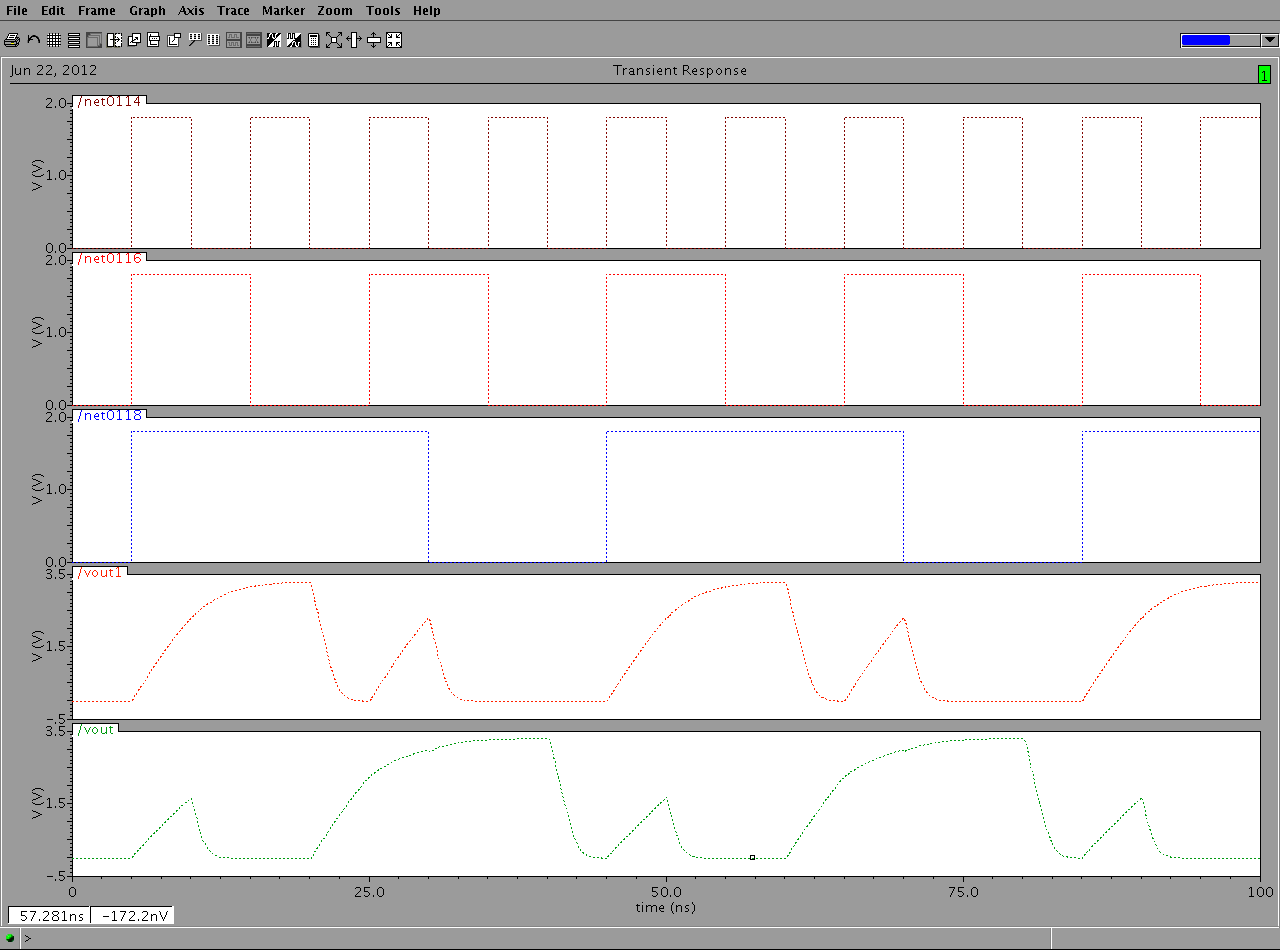


Fig 8.12 ( Sum and Carry when c =2.355 pF )

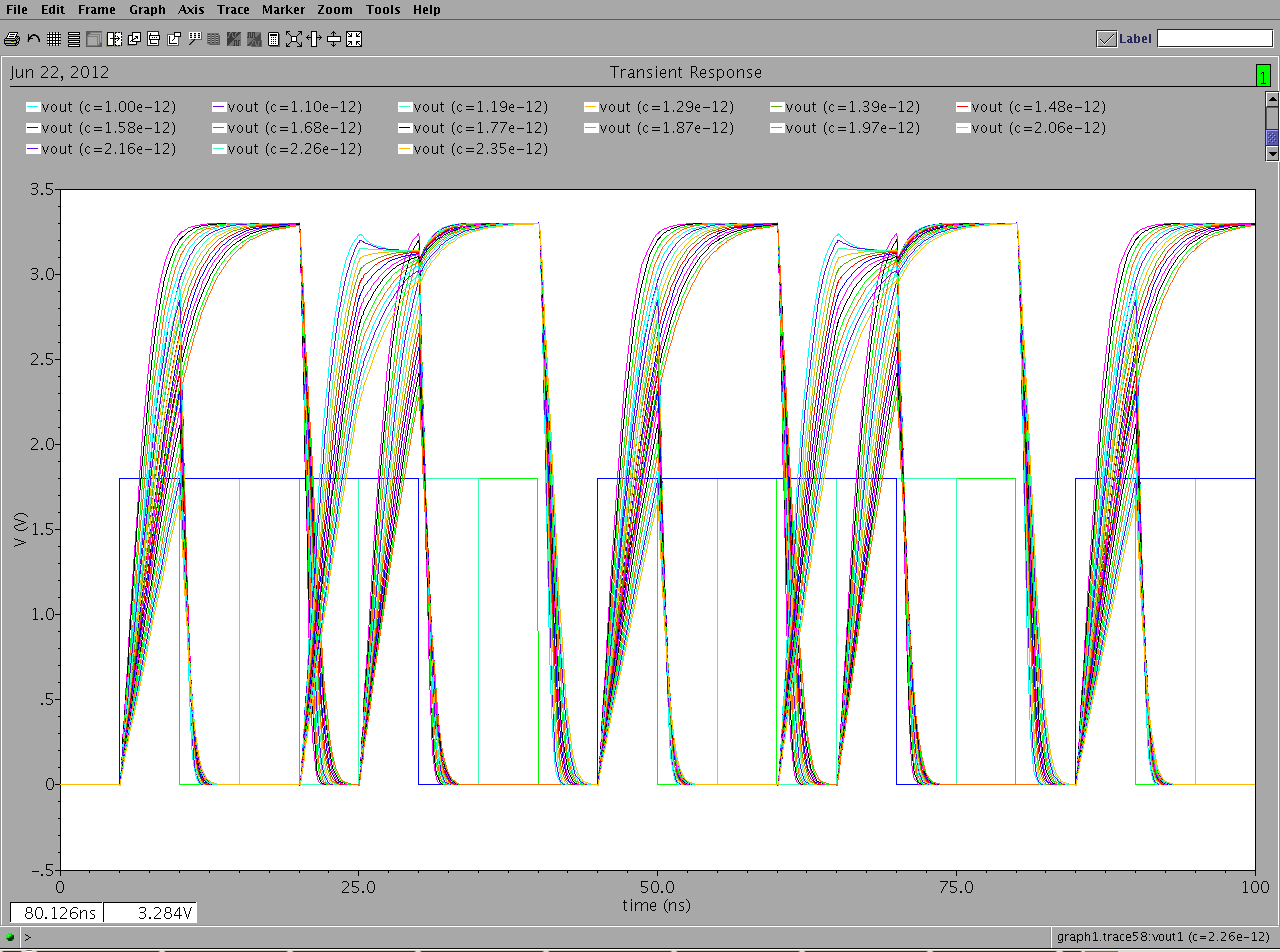


Fig 8.13 ( Parametric Analysis of Sum and carry )

**8.10 Output Waveform when we vary the Frequency**

In an effort to increase the operational speed we increase the frequency of the circuit or in other words reduce the time period. Each period may be considered to consist of charge cycle and discharge cycle. The output voltage level also decreases as the time period decreases the minimum acceptable time period would be such that for which the output voltage level is above the 50% voltage level of the supply.

There is a direct relation between the power dissipated in the circuit and the frequency of the input signal. It is for this very reason that there is a tradeoff between the two parameters. Here we have tried to maximize the frequency and simultaneously maintained the correct working of the circuit. For the upper bound we have maximized the frequency by reducing the overall time period but not the pulse width (we have minimized the distance between the two consecutive pulses). For the lower bound we have reduced both the time period as well as the pulse width. Pulse width is reduced with a consideration that the output voltage level remain just above the 50% mark of the supply voltage.

Fig 8.14 shows the lower bound of the frequency, here the time period is the largest. The capacitor charges to its full swing and the output voltage almost touches 3.3v level. Though this is our lower bound of frequency, we have tried to maximize the frequency by reducing the time between subsequent pulses. Here pulse width = 6.954 ns & time period = 9.354.

Fig 8.15 shows the upper bound of the frequency, here the time period is minimum. The pulse width is such that the maximum voltage for carry is just above 1.65v. Here pulse width = 2.1485 ns & time period = 3.5689.

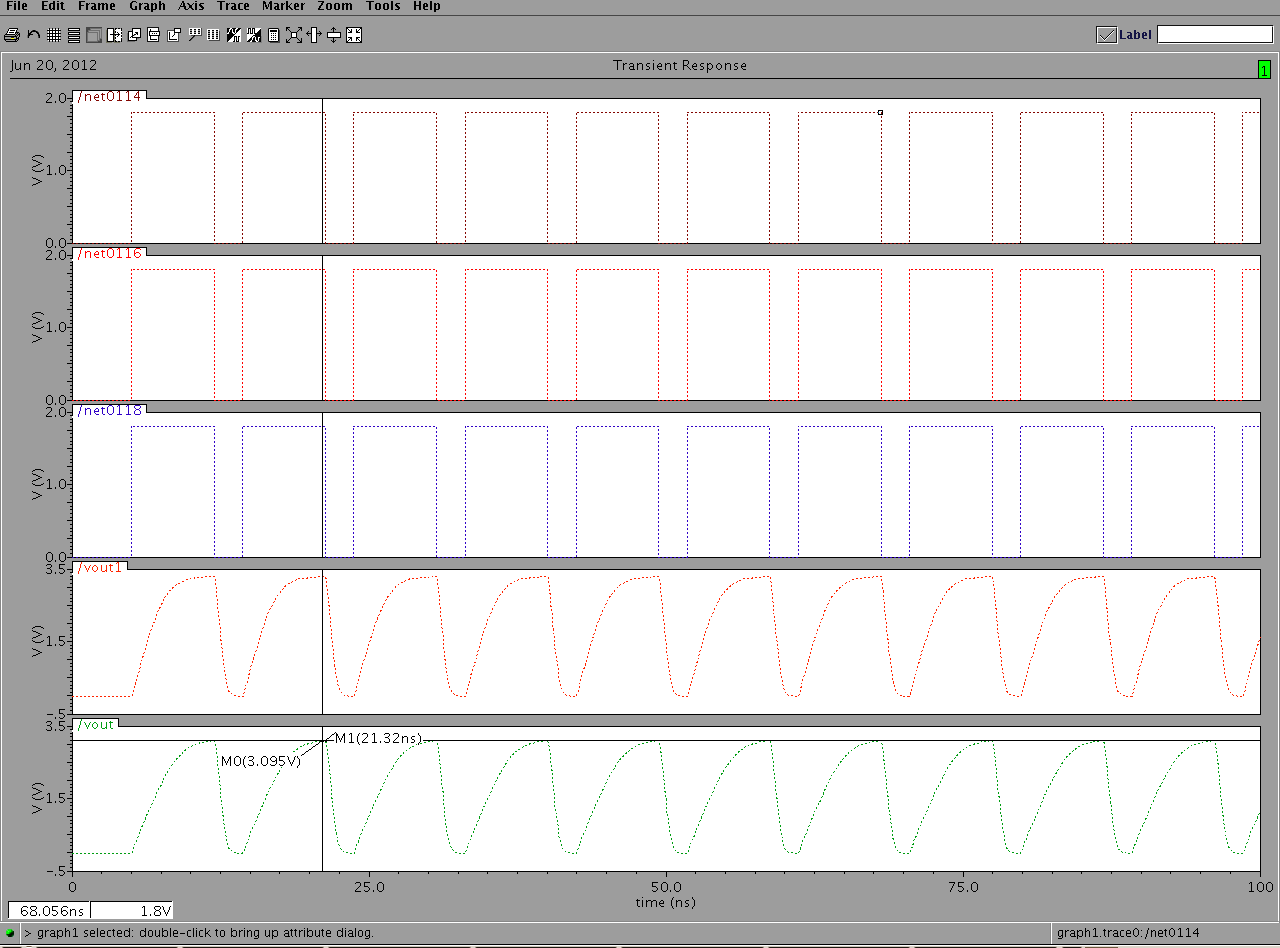
****

Fig 8.14 ( Waveform representing the lower bound for frequency )

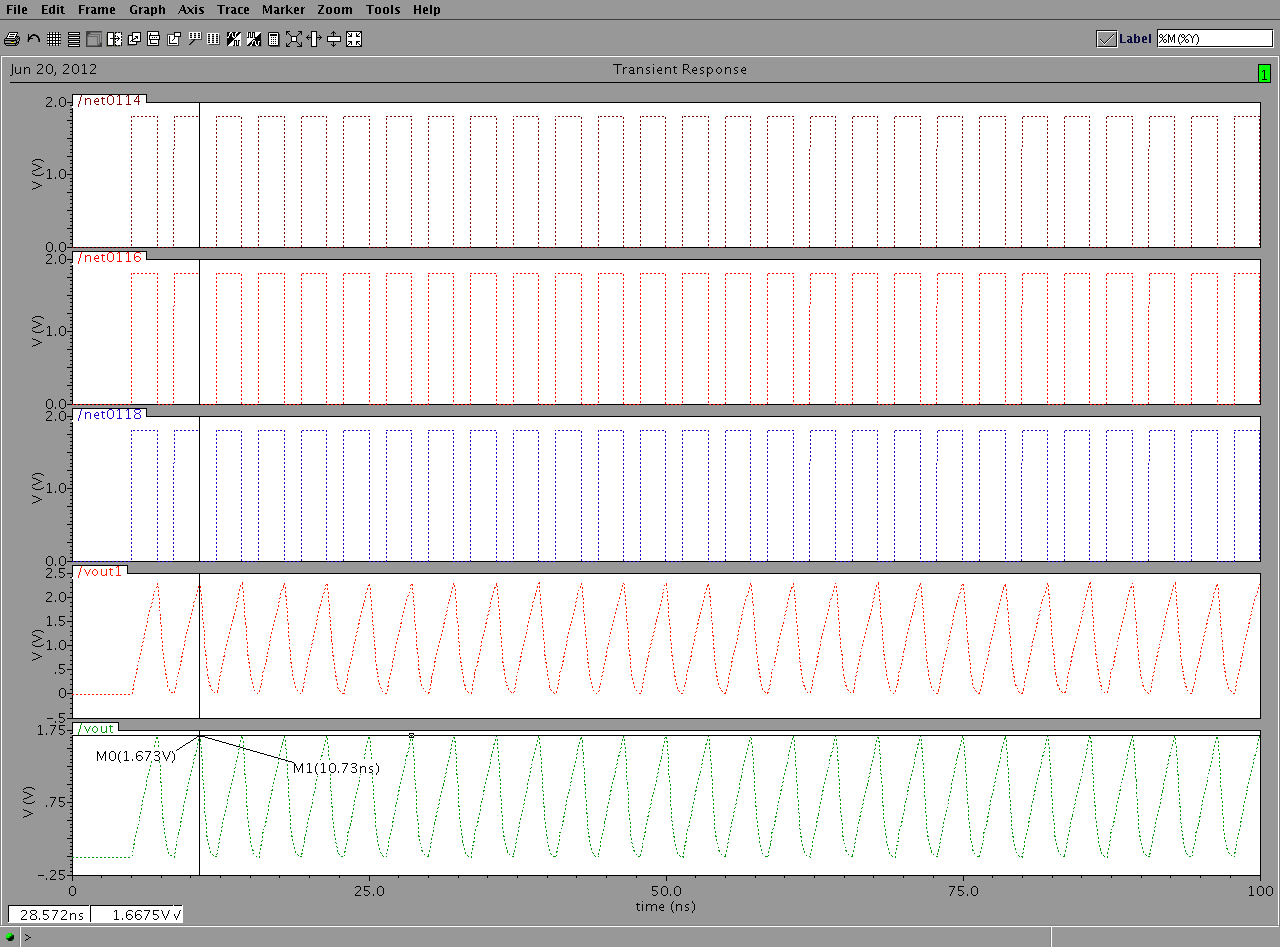
****

Fig 8.15 ( Waveform representing the upper bound for frequency )

**CHAPTER – 9**

**FINAL RESULTS**

**9.1 Results from Genetic Algorithm**

In order to reduce the power usage of our electronic equipments many researchers have tried to optimize power by using numerous techniques. Some have tried to alter the circuit structure and others have tried to change the characteristics of the components.

Here we have utilized the genetic algorithm to find the most suitable set of values for the parameters those which affect the dynamic power. The values of capacitance and frequency thus found adhere to satisfactory outputs in terms of voltage level speed and logical correctness of a full adder circuit.

The figure 9.1 shows the snapshot of the genetic algorithm toolbox. Here the formula for dynamic power is used as the fitness function. The relation between output voltage level, resistance, capacitance and charging and discharging time is used as a constraint. After applying the genetic algorithm we get the optimized value of the power to be 1.164244222m W, the corresponding capacitance and frequency are found to be 1p F and 106.90 MHz.

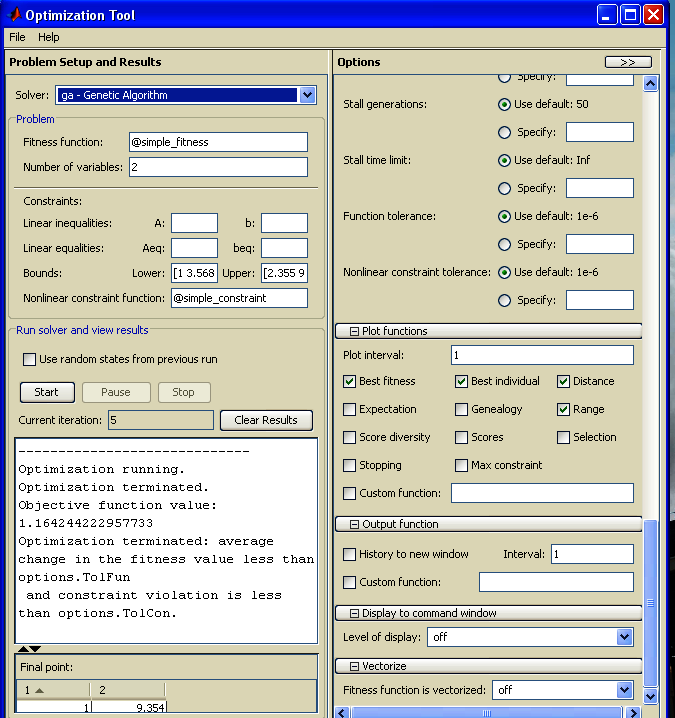


Fig 9.1 ( Optimized Values of Power using Genetic Algorithm )

The figure 9.2 & figure 9.3 shows the execution of genetic algorithm for each iterations. The blue spots correspond to the results for each iteration. After iterating for five cycles the genetic algorithm is able to approach to a credible result. This result is shown in the main genetic algorithm.

****

Fig 9.2 (Execution of Genetic Algorithm )

****

Fig 9.3 (Execution of Genetic Algorithm- Multi Object)

**9.2 Comparison between results**

Here we have utilized the genetic algorithm to find the most suitable set of values for the parameters which affect the dynamic power. The values of capacitance and frequency thus found adhere to satisfactory outputs in terms of voltage level, speed and logical correctness of a full adder circuit.Figure 9.2 and 9.3 show the result obtained by running genetic algorithm and shows the optimized values of the power for the corresponding variables.

The optimized value of the dynamic power as calculated by genetic algorithm is 1.1642m W for the permissible value of variables, while the power calculated manually for the extreme values of variable is 7.180mW. The value of dynamic power found out by genetic algorithm is not only lesser than the manually calculated power but also takes care that the frequency is high and the output level voltages are sufficient to be accepted as logic ‘1’ and logic ‘0’.

**CHAPTER – 10**

**CONCLUSION AND FUTURE SCOPE**

**10.1 Conclusion**

The role of electronic gadgets in our daily lives has been increasing for the past decades and will continue to grow at higher pace in future. Optimization of power at all levels of design becomes important. We not only need to reduce the overall power dissipation but the power should also be reduced at the circuit level. We have made use of the genetic algorithm to search optimized parameters for cmos transistors, These optimized values of parameters enable to reduce the power dissipation to its maximum extent. Genetic algorithm is a powerful heuristic search algorithm which gives reasonably good results.

After applying genetic algorithm the optimized values obtained for capacitance and time period are 1p F and 9.354ns respectively. This means the value of frequency is 106.90 MHz. The optimized value for power is 1.16424422 mW. Here the operating frequency is quite high and the power dissipated is low. When we run the genetic algorithm in multi objective mode the optimized values obtained for capacitance and time period are 1p F and 8.865ns respectively. The optimized value for power is 1.228468 mW

**10.2 Future Scope-Power**

Here we utilized the genetic algorithm to find us the most suitable values of parameters for CMOS transistors, we may further try to optimize the power by altering the structure of the circuit. We may test the circuit with different technologies such as CPL, DPL, pass transistors etc.

Some other heuristics methods such as simulated annealing, Bacterial foraging, Particle swarm optimization etc may be implemented and their results may be compared to obtain better results. Different algorithms may give better results under different conditions.

We have basically focused our efforts towards the betterment of Dynamic power, Static power is still untouched, similar method may be utilized to minimize the static power.

The supply voltage here is fixed at 3.3 volt, this supply may be further scale down to achieve ultra low power circuit.

We may reduce the number of transistors and rearrange the interconnections such that the logical output remains the same. When the number of transistor reduces, the corresponding power reduces by the same factor.

REFERENCES

[1] Ahmed M Shams, Tarek K Darwish and Magdy A Bayoumi, Performance Analysis of Low Power 1-Bit CMOS Full Adder Cells, IEEE, Feb 2002.

[2] Pinaki Mazumder and Elizabeth M. Rudnick, The Complete guide to VLSI design with genetic algorithms, Pearson Education, 2006.

[3] Sung-Mo Kang, Yusuf Leblebici, CMOS Digital Integrated Circuit Analysis and Design, 2003

[4] Mohammed Sayed and Wael Badawy, Performance Analysis of Single Bit Full Adder Cells Using .18, .25 & .35 um CMOS Technologies.

[5] Deepchand Patel ME, WCE, Sangli, Prvinkumar G. Parate ME, WCE, Sangli, ASIC Implementation of 1 Bit Full Adder, First International Conference on Emerging Trends in Engineering and Technology

[6] CMOS power consumption and Cpd calculation, Texas Instruments, SCAA03B, 1997.

[7] Official website of Math Works.

[8] Chip-Hong Chang, Jiangmin Gu, A Review of .18um Full Adder Performances for tree Structured Arithmetic Circuits, IEEE 2002

[9] Anindya Ghosh, DEBAPRIYO GHOSH. Optimization Of Static Power, Leakage Power And Delay Of Full Adder Circuit Using Dual Threshold MOSFET Based Design And T-Spice Simulation.

[10] Amir Amirabadi, Yousef Mortazavi, Ali Afzali-Kusha, Optimizing Low Power High Speed Full Adders With Simulated Annealing, IEEE 2004

[11] Cadance Manual by Antonio J. Lopez Martin, Klipsch School of Electrical and Computer Engineering.

[12] Official website of Cadence.

[13] Uming Ko Poras T Balsara & Wai Lee, Low Power Design Tehniques For High Performance CMOS Adders

[14] M. Basarab And M. Ivoilov, Application Of Genetic Algorithms For Solving Problems Of VLSI Blocks Optimum Placement By The R-Functions Method, Msmw'07 Symposiuim Proceedings. Kharkov, Ukraine, June 23-30, 2007.

[15] Jens Lienig, Physical Design of VLSI Circuits and Application of Genetic Algorithm, Springer 1997.

[16] Randy L. Haupt and Suel Haupt, 2nd addition , practical genetic algorithms, 2004

[17] Kenneth V. Noren and John E. Ross, analog circuit design using genetic algorithm, University of IDAHO

[18] Arvind kumar and Anil Kumar goyal, study of various full adder using EDA tanner tool.IJCST-2012