Abstract— Electronic Systems are necessity of everyday lives. It's an integral part in financial networks, communication systems, power plants systems and personal computers solutions. Electronic systems network is increasingly based on complex and hybrid VLSI (Very Large Scale Integration) integrated circuits. The basic electronic design and automation is concerned mainly with design and production of VLSI systems. Physical design constraints and problems are combinatorial in origin and consist of large problem designs. While there is a shrink in technology, reduction in power consumption and over all power efficiency management on chip are main challenges for any size below 100nm due to increased complexity and area reduction. For many designs, optimization of power is important as timing due to the need to reduce package cost and extended battery life. Darwin carefully observed that, as variations are introduced in concerning population with each new generation of architecture, the least efficient individuals tend to be replaced in the competition of basic necessities of advanced designs. This principle leads to evolution in species origin and functioning. The purpose of Genetic Algorithms (GA) is to search for an optimal-solution to a problem statement. As GA’s are heuristic functioning blocks that can function as optimizers, it is not guaranteed if they will find the optimum, but in any case are able to find justified solutions for a huge resolution of problems. This survey paper aims at a study on Efficient Algorithms for VLSI Physical design including some parameters of power on chip parameters, observe and highlight the traits of the superior contributions.

Keywords— Chromosomes, Gate, Genetic algorithm, Multi level genetic placement algorithm, Mutation, Offspring.

I. INTRODUCTION

VLSI chip contains more than 100 million transistors. Electronic Design Automation (EDA) systems are able to simplify the extremely complex design process of VLSI chips by not exposing the low level circuit theory and device physics to the designer. This encapsulation allows the designer to concentrate on the functionality of the circuit and ways to optimize it. The typical figures of merit for VLSI systems are concerned with maximizing reliability and circuit speed while minimizing the size of the physical package, power consumption, etc.

In creating a VLSI system, a designer travels through six major steps as shown in figure 1.

Specification: It produces a functional specification of the system under design.

Logic Design: This transforms the functional specification into a logical representation, typically via Boolean expressions.

Circuit Design: Represents the logic representation as a circuit using components from an available library of modules.

Physical Design: Translates the circuit into a physical package representation. This representation is specified through a set of mask descriptions, which define how the individual layers of the integrated circuit are to be produced.

Fabrication: Uses the physical package representation to fabricate an actual integrated circuit.

Testing: Identifies the presence of manufacturing errors that prevent the integrated circuit from implementing the functional specification. The [1] Genetic Algorithms are evolution based computational models based on Charles Darwin’s research of natural evolution which is based on concept of survival of the fittest [2]. The concept of natural selection is used in order to explain the main reason as of how species have been able to adapt to changing environments and how, consequently, species with similar adaptively may have evolved. All genetic algorithms work on a population or a collection of several alternative solutions to the given problem. Each individual in the population is called a string or chromosome, in analogy to chromosomes in natural systems. The population size determines the amount of information stored by the GA. The GA population is evolved over a number of generations. All information required for the creation of phenotype and behavior patterns of living organism is contained in its chromosomes. GAs adopt two basic processes from evolution: inheritance, or the passing of features from one generation to the next, and competition, or survival of the fittest, which results in weeding out the bad features from individuals in the population. Main objective of GA is to find an optimal solution to a problem, hence GA’s are modelled as function optimization tools, and are not guaranteed to solve for the most suitable solutions, but are able to find acceptable solutions for a wide range of problems [3].
Requirements for lower power consumption continue to increase significantly as components become small battery-powered, smaller as they are and require more complex functionality. In earlier development major concerns for VLSI designers were design area, performance and cost. The idea of reducing power consumption may differ from application to application. In category of micro-powered battery and portable applications such as mobile phones, main aim is to keep battery lifetime low. At processing architecture of designs below 100 nm technologies, power use due to leakage is active with switching activity as the main power management concern. The basic reduced power design techniques, like clock gating for minimizing dynamic power or use of multiple voltage thresholds (multi-Vt) in order to reduce leakage current, are very well-established and are supported by many existing tools [4].

A. Power Dissipation Basics

In a circuit three components are responsible for power dissipation: dynamic power, short-circuit power and static power. Out of these, dynamic power or switching power is primarily power dissipated when charging or discharging capacitors and is described below [5, 6]:

\[ P_{\text{dyn}} = C_L V_{dd}^2 \alpha f \]  

Where CL: Load Capacitance, a function of fan-out, wirelength, and transistor size, Vdd: Supply Voltage, which has been dropping with successive process nodes, \( \alpha \): Activity Factor, meaning how often, on average, the wires switch, f: Clock Frequency, which is increasing at each successive process node. Static power or leakage power is a function of the supply voltage (Vdd), the switching threshold (Vt), and transistor sizes (figure1). As process nodes shrink, leakage becomes a more significant source of energy use, consuming at least 30% of total power [3]. Crowbar currents, caused when both the PMOS and NMOS devices are simultaneously on, also contribute to the leakage power dissipation [4].

B. Low Power Design Space

From the above section it is revealed that there are three degrees of freedom in the VLSI design space: Voltage, Physical Capacitance and data activity. Optimizing for more power is an attempt to reduce these factors one or more. This section briefly describes about their importance in power optimization process.

C. Voltage

Due to its relationship to power system, voltage reduction gives one of the most effective and efficient means of minimizing power consumed. Without the use of any special circuits and technologies, a factor of two reductions in supply voltage gives a level of four reductions in power consumption. Unfortunately, there is speed penalty for supply voltage reduction and delays drastically increase as Vdd approaches to the threshold voltage Vt of the device. The approach to reduce the supply voltage without loss in throughput is to modify the threshold voltage of the devices. Reducing the total voltage allows the supply voltages to be reduced without loss in speed. The limit of reduction to which the Vt can go is set by the requirement to set adequate noise margins and control the increase in the sub-threshold leakage current [6, 7, 8].

D. Capacitance

The power usage depends linearly on the capacitance being switched. In advancement to operating at low voltages, minimizing capacitances offer another technique for minimizing power consumption. The capacitance is to be kept at minimum using reduced logic, small devices, lesser and shorter wires [7, 8]. As with voltage, however, we are not free to optimize capacitances independently, for example reducing device sizes reduces physical capacitance, but it also reduces the current drive of the transistor making the circuit operate more slowly.

E. Switching Activity

There are two components to switching activity: which determines the average periodicity of data arrivals and \( E_{(sw)} \) which determines how many transitions each arrival will generate [9]. \( E_{(sw)} \) is reduced by selecting proper algorithms architecture optimization, by proper choice of logic topology and by logic level optimization which results in less power [10]. The data activity \( E_{(sw)} \) are combined with the physical capacitance \( C \) to obtained switch capacitance \( C_{sw} = C \cdot E_{(sw)} \), which describes the average capacitance charge during each data period1/Fclk which determines the power consumed by CMOS circuit [11].

Figure 1. Power Dissipation in CMOS
F. Circuit-Design Techniques:

After selecting technology, the focus is on design techniques to optimize power. One has to start by selecting the appropriate logic gate from the standard cell library. Each gate in a standard cell library uses the smallest transistors and has multiple versions with different drive strengths, sizes, delays, multiple-threshold voltage and power consumption. Because the main parameter for controlling active power is the power-supply voltage, cell designers typically design and characterize the gates to operate at voltages as much as 30% lower than the power-supply voltage [12]. Lowering the power-supply voltage produces smaller currents, resulting in more delay. However, this slowdown is acceptable if the design is not pushing the edges of a given technology. Increasing the threshold voltage reduces the leakage current in the device. Leakage power is also controlled by designing logic gates with multiple-threshold-voltage devices [13], including standard high and low threshold voltage devices. Figure 4 shows the variation of gate delay Vs leakage power.

Way to do this is simply to download the template, and replace (copy-paste) the content with your own material.

II. DESIGN TERMINOLOGIES

A. Hybrid Genetic Algorithm

A Hybrid Genetic Algorithm utilizes heuristics for improvement of offspring produced by crossover. Initial population is randomly generated. The offspring is obtained by crossover between two parents selected randomly. The layout improvement heuristics Remove Sharp and Local Opt [14] are used to bring the offspring to a local maximum. If fitness of the layout of the offspring thus obtained is greater than the fitness of the layout of any one of the parents then the parent with lower fitness is removed from the population and the offspring is added to the population. If the fitness of the layout of the offspring is lesser than that of both of its parent then it is discarded. For mutation a random number is generated within one and if it is less than the specified probability of the mutation operator a layout is randomly selected and removed from the population. Its layout is randomized and then added to the population.

B. Genotype Representation

The phenotypic representation for the placement problems is basically the pattern that describes the position of the blocks. Binary slicing trees are well suited to represent placement patterns and have already been used in genetic algorithms [15], [16].

During recombination, partial arrangements of blocks are transmitted from parents to offspring. The corresponding operation is the inheritance of sub trees from the parents. Encoding the tree in a string complicates this operation, since the string needs to be decoded into the slicing tree to execute the recombination, and then recoded into an offspring chromosome afterwards. There is no reason for using a string encoding except for the analogy to the natural evolution process, where the genetic information is encoded in a DNA string. When directly using the slicing tree as the genotype representation, further decoding or encoding the tree when applying genetic operators is mostly avoided. Genotype is often encoded as a binary-slicing tree, which defines the relative placement of the cells compose. As in each the orientations of the combined blocks are fixed. Therefore every tree depicts many possible orientations for the stated layout, which greatly improves the performance of the GA. Blocks or sub-patterns in a tree defining a layout, is always stacked vertically upon each other. The pattern characterized by the right successor of an inner tree node is always positioned on top of the pattern characterized by its left successor when combining both parts into a pattern or meta-block.

C. Genetic Operators

During the optimization process the placement of the blocks has to be changed. The genetic operators directly work on the tree-structure by combining subtrees of parents (crossover) and modifying the tree of an individual (mutation). The crossover operator takes two individuals (parent) out of which one offspring is composed by combining two subtrees, one from each parents [16]. Unfortunately, these parts usually do not add up to a complete layout. After the combination of the two subtrees the redundant blocks are deleted and the missing blocks have to be added at random positions to the tree to ensure that the offspring finally represents a correct layout. Mutation operator modifies either by exchanging simple blocks or a block (leaf) with a meta-block (subtree) or by exchanging two meta-blocks.

III. GENETIC PARTITIONING ALGORITHMS

Problem of VLSI circuit partitioning is non polynomial, hard and cannot be effectively solved by deterministic algorithms. Genetic algorithms belong to probabilistic and iterative class of algorithm and are stochastic in nature. Therefore they can be effectively, efficiently applied for VLSI circuit partitioning.
The problem involves dividing the circuit net list into two subsets and some of the connections (edges) are also cut. The number of edges belonging to two different partitions is the cost of a partition. The objective function captures the interconnection information and partitioning solution is optimized with respect to interconnection between the partitions with the constraint of forming balanced partitions.

A. BFS Algorithm

The information of interconnection between the components in the netlist is converted in form of adjacency matrix. This Adjacency matrix [17] information is then used to traverse the circuit in BFS algorithm so that the connected components remain clustered together as far as possible.

B. Initial population:

Once the BFS order of components is obtained it is processed to form the initial solution for GA by converting it into 32-bit chromosome. The 32-bit chromosome [18] contains integer values, with each integer value corresponding to each element of chromosome encoded to represent the partition number assigned and number of elements clustered to form single chromosome element. Value of jth cell of chromosome is n1n2, where, n1 indicates the partition number assigned and n2 indicates the number of components clustered. Though other sorting data structure algorithm can be used such as depth first search algorithm, spanning tree algorithm etc, breadth first search algorithm has been found to capture circuit information more effectively [19]. Using the initial solution, random population is generated of the population size specified by the user. For each individual of the population, cost is computed. Objective function captures the cost of number of interconnections cut between the partitions.

Fitness Evaluation

Using the cost computed, each individual is evaluated for its fitness function. Based on fitness values individuals are randomly selected using roulette wheel selection for crossover operation.

C. Crossover

Each individual is considered for selection as parent for crossover, with probability of selection proportional to its fitness value [20]. Flexibility is incorporated in crossover operation with the user specifying the value for multipoint crossover.

Offspring generated from crossover replace the lowest fit individuals of the population if their fitness value is higher else, no replacement is made in the original population. By the algorithm [14], new offspring replace the equivalent number of worst solutions from previous population which helps in survival of better solutions over several generations.

D. Mutation

After population replacement, mutation is performed on the bits randomly with small probability of mutation. Probability of mutation is very important, because the number of bits to be mutated depends on this probability. Mutation of bits is not similar to the traditional binary mutation operator, which is simple inversion of any random bits (depending on Probability of mutation), in the population [20]. Mutation changes the partition assigned to random number of components, where number of components depends on the probability of mutation. Even the partition assigned is generated randomly. Generally low values of probability of mutation are preferred so that population is not changed drastically which is critical. The population with mutated bits is then evaluated for fitness and again whole cycle of selection, crossover, replacement and mutation is followed and repeats for number of iterations of GA specified by the user. No stopping criteria is specified in the algorithm itself because one of the advantages of evolutionary approach to partitioning is, availability of ready solution at any stage, which if not globally optimal at least guarantees a good solution. But if no improvement is seen in the fitness and min-cut results for consecutive 100 runs on a small scale circuit, GA is terminated [20].

IV. GENETIC PLACEMENT ALGORITHM

Multilevel Genetic Placement Algorithm; called MGPA, is proposed to solve large-scale mixed-size placement problem. MGPA first adopts the well-known multilevel hyper graph partitioning algorithm [21]. Afterward, two key techniques, consisted of GFA [22], MOGFA are individually applied to handle different sets of partition. At the top level, an earlier version of our fixed-outline floor-planner FOGFA [23] is applied to address fixed die-size placement. Finally, the Capo standard-cell placer [24] is used to place standard cells. This new methodology is efficient to simultaneously handle building modules and standard cells. Unlike the methods [25] this approach has eliminated the need to remove overlaps between the movable objects which imply shorter runtime.
A. Multilevel Genetic Placement Algorithm (MGPA)

The multilevel genetic placement algorithm MGPA uses the multilevel partitioning algorithm to effectively reduce the great complexity [25]. The recursive bipartition continues until the size of each partition is smaller enough that floor-planner or placer can handle.

There are three different sets of partition during the multilevel partitioning:

1. The partition is regarded as a soft module, denoted as Ps, where all objects inside the partition are standard cells.
2. The second kind of partition consists of either mixed-size objects or only building modules, where the total area of building modules is much larger than the area of standard cells.
3. The third kind of partition contains mixed-size modules, where the total area of standard cells is much larger the area of building modules.

The number of second and third kind of partition is small that floor-planning techniques can handle. Thereafter, two floor-planning techniques, GFA and MOGFA [26], are respectively applied to determine geometric locations of second and third partitions with area and wire-length minimization. GFA is applied to handle the second kind of partitions to effectively shorten runtime. MOGFA is applied to tackle the third kind of partitions to simultaneously reduce total area and wire-length.

V. CONCLUSION

In this survey paper we have discussed some optimization algorithms which address the large-scale partitioning, floor-planning and mixed-size location problem for fixed diesize SOC network designs. These methods follow the multi-level hyper-graph partition system to effectively remove complexity. The techniques are individually applied for handling different sets of partition done. We observed that MGPA need not to remove overlaps between the movable objects. Our research aims at experimenting efficient techniques for large-scale partitioning, floor-plan & time-driven placements for improving the runtime and the wire length during placement processes. The need for lowering the power systems cost is being driven by many market segments. It is not very easy, as designing of low-power gives another path to an already complex design system and the design needs optimization for power, also with that Performance and Area.

Issues like Technology Scaling, Leakage power, Dynamic power management techniques, Low power interconnect, Development of power conscious techniques and tools for behavioral synthesis, logic synthesis and layout optimization and Power saving techniques have been discussed.

REFERENCES


