Synthesis of Low-Power DSP Systems Using a Genetic Algorithm

Marc S. Bright and Tughrul Arslan, Member, IEEE

Abstract—This paper presents a new tool for the synthesis of low-power VLSI designs, specifically, those designs targeting digital signal processing applications. The synthesis tool genetic algorithm for low-power synthesis (GALOPS) uses a genetic algorithm to apply power-reducing transformations to high-level signal-processing designs, producing designs that satisfy power requirements as well as timing and area constraints. GALOPS uses problem-specific genetic operators that are specifically tailored to incorporate VLSI-based digital signal processing design knowledge. A number of signal-processing benchmarks are used to facilitate the analysis of low-power design tools, and to aid in the comparison of results. Results demonstrate that GALOPS achieves significant power reductions in the presented benchmark designs. In addition, GALOPS produces a family of unique solutions for each design, all of which satisfy the multiple design objectives, providing flexibility to the VLSI designer.

Index Terms—Digital signal processing, low-power synthesis, VLSI design.

I. INTRODUCTION

The power consumption of very large scale integration (VLSI) devices has become an important parameter in recent years, largely due to the explosion in the use of portable communication and computing systems. Of particular interest in such systems are digital signal processing (DSP) devices. These devices are specialized processors used widely in complex functions such as telecommunications, data compression, and speech processing. The increasing requirements for portable systems to incorporate these functions has led to increased demand for low-power DSP devices. However, the design of DSP devices that offer complex functions with low-power consumption requires the use of advanced automated synthesis tools.

Traditional automated synthesis tools optimize the VLSI device for speed and area. This presented a complex solution space, especially when considered at the high level of abstraction (the design capture stage). The addition of power as a design objective compounds the complexity of the high-level synthesis task. Therefore, high-level low-power synthesis tools require a more robust search and optimization mechanism to produce designs with optimal tradeoffs between the objective parameters.

Genetic algorithms (GAs) [1] have proven to be a successful technique for tackling the complex problems inherent in the design and optimization of VLSI devices. Examples are VLSI synthesis tools such as that developed by Arslan et al. [2], which uses a GA for the structural synthesis of logic circuits. A tool for minimizing VLSI implementation area by using a GA to reduce the size of functional operators was developed by Martin and Knight [3]. The authors of this paper have previously demonstrated the application of GAs in low-power synthesis [4], [5] using a restricted GA with limited genetic operators.

This paper presents genetic algorithm for low-power synthesis (GALOPS), a GA for the high-level synthesis of low-power CMOS-based DSP designs. GALOPS uses problem-specific techniques for low-power synthesis. As illustrated here, the incorporation of these techniques into the GA framework requires modification of the standard genetic operations. The primary objective of GALOPS is to produce a minimum power design from an initial high-level specification, while tracking other performance constraints such as area and speed. The contributions of this paper are as follows:

• the formulation of a GA that is capable of handling the VLSI low-power synthesis problem, considering the specified performance constraints of CMOS devices;
• the development of problem-specific genetic operators that manipulate a library of low-power transformations. Standard genetic operators are modified to incorporate these transformations, in particular, a crossover operator is developed which recognizes and applies power-saving transformations to designs. The developed crossover operator preserves the notion of inheritance in a genetic algorithm;
• results that show the significant power reductions obtained using the GA synthesis technique (GALOPS). The effects of relaxing design constraints, such as area, have also been analyzed and compared;
• analysis of the capability of a GA-based synthesis tool to present multiple solutions to a problem, exploiting the multiple-solution nature of the GA search technique.

This paper is organized as follows. Section II discusses the need for low-power design and presents the advantages of considering power as a high-level objective parameter. Section III begins with an introduction to the key concepts of high-level synthesis. The section continues with a review of the area of high-level low-power synthesis and presents a detailed description of the low-power synthesis technique used by GALOPS. Section IV discusses the implementation of GALOPS with particular attention to the incorporation of the standard low-power synthesis techniques into a GA. Section V presents an analysis of the preferable rate of applying different genetic operators, as this has a significant effect on the quality of the final solution.

Manuscript received December 12, 1997; revised April 1, 1999, August 12, 1999, and January 26, 2000.
M. S. Bright was with the School of Engineering, Cardiff University, Cardiff CF24 0YF, Wales, U.K. He is now with the Broadcom Corporation, Bristol BS16 1EJ, U.K.
T. Arslan is with the Department of Electronics and Electrical Engineering, University of Edinburgh, Edinburgh, EH9 3JL, Scotland, U.K.
Publisher Item Identifier S 1089-778X(01)02136-1.
A significant component of a power synthesis tool is the power estimation module. Section VI presents the problem of high-level power estimation and the method of incorporating such a module into a GA’s fitness evaluation function. The implemented power estimation function satisfies the goals of enabling comparison between high-level designs, while maximizing the speed of the complex power estimation process.

Section VII presents a comprehensive benchmark set employed to illustrate the effectiveness of GALOPS as a high-level low-power VLSI synthesis tool. In Section VIII, the performance of GALOPS is analyzed and results are presented to demonstrate the ability of GALOPS to determine a number of unique solutions that satisfy the design criteria.

II. THE LOW-POWER DESIGN PROBLEM

In recent years, the portable computer-systems market has become one of the fastest growing areas of the information technology market [6]–[9]. The portability of these systems places severe limitations on size and weight, of which the battery cells are a major component. In addition to the drive for lighter and smaller systems, there is an increasing demand for portable systems to provide the wealth of multimedia and communication options that are available with the larger desktop systems. This places an increasing burden on the already lean power budget. Although considerable developments have been made in increasing battery capacity, the technology has peaked in recent years [7]. To provide a system with more functions and greater operating time, without increasing system size and weight, the power requirements of the portable systems have to be reduced.

At the VLSI device level, the trend toward integrating a greater number of components on a single wafer has increased the power consumption and, hence, the heat dissipation of these devices. This has led to many devices needing cooling systems as standard, which can add significantly to device cost [10]. For example, the 30 W of power dissipated by the DEC Alpha processor contributes to the high packaging costs [11]. The problem of heat dissipation is also beginning to impact the number of components that can be integrated onto a single CMOS VLSI device. Increased device integration will require that the power dissipation problem be addressed.

The requirements for longer battery life and lower heat dissipation have led to the increased importance of power consumption as a VLSI design parameter, as identified by Arslan et al. [6]. A variety of techniques have been developed to reduce power consumption at different levels of the CMOS VLSI design and fabrication process. For example, component sizing and floorplanning have been identified as important elements for power optimization at the physical level [6]. At another design level, the reduction of component threshold-voltage levels has been proposed [7]. The former technique requires considerable modification of the design process while the latter technique offers power reduction at the considerable cost of altering the CMOS VLSI fabrication process. In contrast, the consideration of power dissipation as a high-level design parameter requires a relatively small investment and will yield the greatest benefits [7]. This is because the lower level reduction techniques all operate within the small solution space under constraints set by the higher level design steps. If power consumption is considered as an initial high-level design parameter, it will provide greater freedom to search the power solution space [12]–[15].

Traditional high-level synthesis tools operate under the multiple design constraints of minimizing area and maximizing speed. Many of the tasks of traditional high-level synthesis are NP-hard [16]; the addition of power-reducing design techniques increases the complexity of the problem from two to three dimensions [17]. The NP-complete nature of the problem requires an efficient algorithm to tackle the power solution space. In this paper, we present GALOPS, a synthesis tool that uses a genetic algorithm [1] to search this complex power solution space efficiently, producing high-level designs automatically with lower power requirements.

III. HIGH-LEVEL LOW-POWER SYNTHESIS

A. Introduction to High-Level DSP Synthesis

The term “high-level synthesis” is used to describe the process of turning a behavioral description or algorithm into an architectural-level specification. The behavioral description describes the input and output behavior of the algorithm in terms of operations and data transfers separate from any VLSI implementation details. The architectural level maps the operations and data onto functional units and registers implemented in hardware, effectively producing a block diagram of the VLSI device. The main tasks of the high-level synthesis process are scheduling, allocation, and binding of the behavioral description to produce the layout of the hardware structure.

Scheduling is defined as the problem of mapping a set of operations in the behavioral description onto a set of time steps such that the functionality of the description is preserved. This is effectively an ordering process where the algorithm is analyzed to determine the order in which operations have to be performed. The goal of scheduling is to obtain a schedule length (i.e., the number of time steps) that obeys the required order of operations, satisfies the speed requirements, and minimizes the implementation area. The allocation and binding processes are typically performed in parallel, subsequent to the scheduling process (it is recognized that concurrent execution of all three steps, although a nontrivial task, may produce the best results [15]). Allocation assigns the scheduled operations to functional unit types (e.g., adders, multipliers). Binding assigns the scheduled operations to specific instances of the functional unit as there is typically more than one of each functional unit type. During this process, there are many decisions to be made which affect the final hardware implementation (e.g., numbers of hardware units to use, maximum number of time steps allowed). Performing the synthesis process under different criteria, such as low power, high speed, etc., will dictate these decisions and will produce different hardware implementations. For a more extensive analysis of the subject of high-level synthesis, the reader is referred to [15].

The VLSI industry’s requirement for high-level low-power synthesis [7] has led to the development of a number of low-power synthesis tools. For example, the tool developed by Raghunathan and Jha [18] exploits the difference between the
designed and maximum operating speed to produce low-power designs. This is possible because the maximum operating speed of designs is often greater than the required speed. Therefore, slower but less power-hungry hardware functional units can be used to reduce power while obeying operation constraints. Other tools described in [19] and [20] examine the allocation of functional units to operations within a signal-processing design. The allocation is performed to minimize the activity and reduce the power contribution of the controller section of the design. Both of these tools use constrained linear programming solutions to tackle the intractable nature of the optimum allocation problem. San Martin and Knight [21] describe the use of a GA for the selection of functional units (adders, multipliers, etc.) from a library of units with different power, area, and speed requirements. The GA selects the optimum functional units to minimize power while maintaining system speed.

The techniques described previously concentrate on optimizing the scheduled implementation of a high-level design rather than modifying the high-level design itself. Hence, these techniques operate on a fixed design. This limits the solution space for the power optimization techniques, hence producing restricted designs that do not fully exploit the potential for low-power implementation. Modification of the high-level designs before scheduling, which is the problem addressed in this paper, gives greater freedom to explore the power design space as the search is not restricted by decisions made at earlier stages in the design process.

The authors have developed a set of benchmark high-level DSP designs for evaluating the presented low-power synthesis system. The benchmark designs are a diverse set of DSP functions of varying complexity. They include examples of recursive and nonrecursive structures (structures with and without feedback) that have been used for benchmarking high-level design methods such as scheduling and allocation. To distinguish GALOPS from other synthesis tools in the literature, it has the following original features.

- Use of a specifically tailored GA for high-level low-power synthesis. GAs have shown considerable success in the complex combinatorial optimization problems of VLSI synthesis under multiple design constraints [2], [3], [26], [27].
- GALOPS produces a number of designs that satisfy the design criteria. Multiple solutions are produced because a GA uses a population of individuals to explore the solution space. At the end of the evolution process, the population consists of a number of distinct solutions that have the same power-consumption characteristics. These multiple solutions provide greater flexibility to the VLSI designer.
- The designs produced are platform independent, and hence can be applied to general hardware architectures, a desirable feature of a high-level synthesis system.

B. High-Level Power Reduction Using Transformations

This section discusses the factors contributing to power consumption before explaining how they are reduced through the application of the HLTs.

1) Power Dissipation: Power dissipation in a CMOS device consists of three elements: short-circuit power, leakage power, and switching power. The short-circuit and leakage components can be made negligible through the application of appropriate design techniques [6]. This leaves switching power as the main constituent, which is the power required to perform the switching events across the device [6], expressed as

\[ P_{\text{average}} = V_{DD}^2 \times C_e. \]  

(1)

From (1), \( V_{DD} \) is the supply voltage and \( C_e \) is the effective capacitance switched within the device (switched capacitance). This is a function of the device’s physical capacitance and switching activity. Note that \( f_{\text{samp}} \) the sampling frequency of the device, is not included in this formula. Typical high-level power tools operate under the constraint of preserving the initial sampling frequency; hence, the \( f_{\text{samp}} \) term is removed from the equation as it is constant and is not used in comparative power analysis [8].

2) Power Reduction: Equation (1) identifies that power has a quadratic dependency on the supply voltage. Therefore, a reduction in supply voltage is targeted as the most effective method of reducing power. However, reduction of the supply voltage increases the delays in a CMOS VLSI device, which will reduce the computational throughput, as shown in Fig. 1. Fig. 1 presents the effect on delay (normalized with respect to the delay at 5 V) of reducing the supply voltage. As an example, consider a reduction of the supply voltage to approximately 3 V. This results in a twofold increase in the delay of a device, halving its computation speed. The HLTs can be used to compensate for this delay, as discussed in Section III-C, thereby producing a device with the original throughput rate, but a lower supply voltage.

Equation (1) also identifies switched capacitance as a component of power dissipation, although it does not have as great an influence as the supply voltage. The HLTs can also be used to reduce the switching capacitance. The distributivity transformation [28] is an example of an HLT that reduces the number of operations and, hence, the physical capacitance of a design. This transformation uses mathematical precedence relationships to reconfigure the sequence and number of mathematical operations, which may reduce the required number of hardware units. Careful ordering and cascading of operations in a design [8] has also been shown to have an effect on the switching activity.

The disadvantage of HLTs is that they compound the already complex nature of the low-power synthesis problem. The use of only a single HLT, known as retiming [17], for design optimiza-
tion has been shown to be an NP-complete problem [22], [23]. The complexity of the problem implies that it is very unlikely that an algorithm could be developed that is guaranteed to find the best solution in polynomial time [22]. To solve the complex problem, many low-power synthesis tools use a combination of heuristic and probabilistic techniques to apply the transformations, such as the use of simulated annealing in conjunction with VLSI design rules in [22].

GALOPS uses a GA to access a central library of HLTs within a genetic framework. The HLTs are applied to individual designs within a population at predetermined probabilistic application rates to search the complex solution space inherent in low-power synthesis. The ability of the GA to escape local optima is of prime importance as the application of one set of HLTs that increase power consumption may subsequently enable the application of further HLTs that will yield a greater power reduction. GALOPS primarily targets voltage reduction to reduce power while concurrently tracking the effect of the transformations on capacitance and speed.

C. The High-Level Transformations

HLTs operate on high-level designs represented as data flow graphs (DFGs). A DFG consists of high-level functional blocks (adders, multipliers, etc.) and connection nets that denote data flow. An example of a typical DFG is given in Fig. 2. This example is a simple digital filter; samples are fed into the input of the DFG and are processed by the functional units. The adder sums the current input with a time-delayed (by the delay block) version of the filter’s output to produce the current output. This is an example of a recursive filter as it contains a feedback loop. HLTs operate on the functional blocks of the DFG to alter its VLSI-implementation characteristics (power, speed, area, etc.). By manipulating the functional blocks (e.g., changing the order of execution or number of blocks), the VLSI implementation (determined through high-level synthesis) of this algorithm can be modified while preserving functionality.

The HLTs used within GALOPS are chosen for their qualities of increasing system speed to allow for a reduction in supply voltage. The transformations are as follows.

- Retiming [28], [17], [16] is the process of moving the delay elements from the input of a functional element to its output in order to minimize the critical path. The critical path is the longest computational path between delay or input–output elements. Input samples can only be processed once all computation in this longest path has been completed. Therefore, the length of the critical path of the DFG governs the speed of a design. A shorter critical path produces a higher speed device.

- Automatic pipelining is a special combination of retiming and pipelining [30]. Delay elements are inserted on the primary inputs of the DFG and are then retimed through the DFG in an attempt to minimize the critical path.

- Loop unfolding [31] is a complex transformation that “unfolds” the DFG to create a parallel implementation. Speed increase is achieved as N parallel processing units have an effective processing speed of N times the speed of one unit [7]. The parallel designs produced by unfolding may have a large area and capacitance overhead. This overhead may be reduced with the application of the postponing principle [32], which postpones unfolding to first obtain the best solution with the other transformations; unfolding is then applied in an attempt to improve this solution.

Fig. 3 gives an example of HLTs applied to a DFG that represents a second-order digital filter. The arrows depicting the critical path are not part of the DFG. They are included to explain the operation of the HLTs. In this example, a combination of retiming and pipelining has decreased the critical path from four elements to two as delay elements represent storage rather than computational operations. This reduction in the critical path length yields a 100% speed increase. This speed increase is not required and is, therefore, traded off for a reduction in supply voltage, returning the device to its original speed. This is achieved by reducing the supply voltage from 5 to 2.9 V (as determined by the delay versus voltage model), producing a significant power reduction (over 50%).

IV. THE GENETIC SYNTHESIS PROCEDURE

GALOPS uses a GA to optimize the power-consumption requirements of a target DSP design, represented as an initial DFG. A typical GA is an evolutionary computation technique
that modifies a population of solutions over a number of generations to produce a solution to a specified problem. The GA utilizes evolutionary-inspired genetic operators, such as crossover and mutation, to solve complex problems that typically cannot be solved with conventional methods. Conventional GAs operate on binary-coded chromosomes and apply relatively elementary genetic operators such as the inversion of bits (mutation) or the exchange of bits (crossover). The implementation of GALOPS required the development of problem-specific chromosomes and operators that utilize standard GA concepts within a DSP specific framework. The previous description of the low-power design problem indicates that the main components of a GA-based design-specific synthesis tool are as follows:

- chromosome structure that represents candidate designs; this structure must facilitate the application of the HLTs, and provide functional and VLSI implementation information for the evaluation of the candidate designs;
- fitness evaluation function that performs the complex task of high-level power estimation rapidly;
- genetic operators that access a library of HLTs to alter the power characteristics of the candidate designs while preserving their functionality; the specific genetic operators utilize a library of HLTs to produce a DSP design with the same functionality, but lower power requirements than the initial DFG;
- standard genetic framework for the manipulation of a population of designs and the selection of individuals for modification.

This section describes the GALOPS synthesis procedure, concentrating on the development of problem-specific genetic operators and the fitness evaluation function.

A. The Genetic Algorithm

A genetic algorithm requires a diverse initial population to start searching the solution space from multiple locations. Random creation of solutions is one method of creating a diverse population. However, randomly created DFGs would have different, possibly impractical functions. As functionality of the DFG is of prime importance in the GA synthesis tool, the initial DFG to be optimized is used to create an initial population. A random selection technique is used to apply a single HLT to each member of the initial population, thus producing a population of DFGs with the same output function, but a different topology.

Parent selection is performed using the roulette wheel [1] proportional-fitness selection method. Experimental analysis of elitism [1] with a variety of designs was found to improve GA performance significantly. This analysis was performed with a fixed population size of 500 individuals. The task of determining the effect of population size and the selection of an optimum population size for particular designs is a matter of continuing research. The current system uses elitism to create 4% of the next generation from the fittest members of the previous generation. This value was determined through experimental analysis, and was found to produce the best overall GA performance.

Genetic evolution is terminated when designs with the specified criteria have been synthesized. The system works under the criteria of either attempting to find the best solution or attempting to find all designs that meet user-specified power requirements. However, for certain designs, it may not be possible to synthesize a low-power design that meets the specified requirements. As an addition to the target design termination criteria, the GA also terminates evolution if the best solution found does not change over a large number of generations. This termination criterion was set to 1000 generations for the results presented in this paper.

B. Problem Representation

The design to be synthesized is presented to the GA as a net list containing function and connection information for each element in the DFG. A preprocessor step converts this net list into the chromosome representation used within the GA. An example of a digital filtering DFG is shown in Fig. 4 along with its corresponding chromosome. The filter contains two functional elements (adder, multiplier), one storage unit (delay), one primary input, one primary output, and four connection nets (n1-n4).

The chromosome structure is a complete representation of the corresponding DFG. The structure was designed to facilitate the application of the power-reducing HLTs as well as to provide a representation that does not require complex encoding or decoding steps during fitness evaluation. The chromosome is “cross linked” so that connectivity data present in the DFG is preserved when encoded. This results in each gene containing connectivity data that list its input and output nets. For example, the multiplier of Fig. 4 has a unique label (Num) c4. Its function (Fnc) is stored as a multiplier. Nets n3 and n4, also stored within the gene, detail its input (In1) and output (Out) nets, respectively. Multiplier c4 is a constant coefficient multiplier. Hence, it is shown as having only one data input.

When assessing a chromosome’s fitness for power consumption, a number of real VLSI design parameters are used. To provide this information, each gene has the functional type (Fnc) of its corresponding DFG element stored. This type of information can be used to look up area, delay, and capacitance characteristics specific to that chromosome type.

The primary inputs of the DFG are encoded as functional elements. This provides convenient entry points in the chromosome
from which it can be analyzed for fitness evaluation or modified through the application of HLTs. The primary outputs are not required during the modification or evaluation processes so they are stored externally in a lookup table. This output table is used to decode the chromosome back into a high-level design at the end of the genetic synthesis process.

Chromosome modification through the application of HLTs produces chromosomes of varying sizes throughout the evolutionary process. For example, the pipeline transformation inserts delays into the DFG. A fixed-length chromosome, with “empty” genes left for future growth, is memory inefficient, and may not be large enough to accommodate all possible pipeline transformations. To ensure complete flexibility, the chromosome is of variable length, so genes can be added and removed during the modification process.

C. Genetic Operators

This section describes the mutation and recombination operators of GALOPS. The problems of using standard genetic operators are discussed, and modifications to these standard operators are presented.

1) Mutation: Standard mutation operators randomly modify the individual genes in the chromosome to produce genes of other values [1]. Such a mutation scheme used in the genetic synthesis process, which modifies high-level designs, would change the function of genes in the chromosome. If applied to a DFG, it would produce a corrupt DFG with incorrect functional operation. The low-power solution must be one that has no functional difference from the original solution.

One method of mutation used in genetic synthesis tools, where functionality of the chromosome is of paramount importance, is to use a repair operator [2]. The GA checks the mutated chromosome’s function. If the chromosome is corrupt, the repair operator is invoked to produce the desired function. This repair technique requires detailed knowledge of both the required and current functions of the chromosome. Such information for a DFG would be obtained through simulation, a computationally intensive task. The computation expense is not the only disadvantage of using the repair operator within this synthesis system. With the complex feedback paths and loops of a DFG, repairing the chromosome would require a considerable alteration of the DFG structure, which could negate the work of the mutation operation.

An alternative to using a repair operator is to modify the mutation process so that it does not corrupt DFG functionality. The HLTs are VLSI design techniques that dictate what modifications can be made to a DFG. The HLTs operate on individual elements within the DFG (genes within the chromosome) to produce a new DFG (chromosome) with different characteristics (fitness). Therefore, these HLTs are considered the mutation operators of the genetic synthesis system. An example of a mutation operation is shown in Fig. 5.

Predetermined application rates are used to select which transformation is applied; in this case, the retiming transformation. The selected delay is retimed through the multiplier to produce a new DFG with a reduced critical path (hence, faster speed) but the same function. The speed increase is traded for a supply voltage reduction, resulting in a lower power solution.

Each operator is applied at a specific rate; in the current system, retiming is applied to 20% of the population and pipelining and automatic pipelining are both applied to 1%. Unfolding is initially applied at a very low rate, which is increased after a number of generations. This follows the postponing principle introduced previously. These application rates were determined using a combination of design heuristics and experimental data as discussed in detail in Section V.

2) Crossover: The crossover operator, like the mutation operator, is also modified to produce DFGs with no change in functionality (i.e., prevent corruption). Conventional genetic crossover combines the characteristics of two parent chromosomes to produce child chromosomes with different, hopefully better characteristics. The characteristics of the chromosomes in the GALOPS system are dictated by the transformations that have been applied to them during genetic evolution. The noncorrupting crossover technique selects transformations from one parent to apply to the other parent, producing children with some of the transformation characteristics and, hence, the genetic characteristics, of both parents. The transformations are identified by analyzing the parent chromosomes for genes modified or inserted by specific transformations. Each transformation assigns a particular code to genes it modifies (i.e., all genes modified by a single transformation have the same code prefix on their Num; thus, a complete transformation can be recognized through the set of genes affected). This is effectively a “local” crossover operation that detects and reproduces a single transformation from parent to child, thus reducing the complexity of detection and the cost of applying this crossover operation. Strictly, the crossover operation can be considered as a special form of mutation, where transformation characteristics are inherited rather than resulting from the direct reproduction of genes. The presented crossover technique is analogous to a localized “two-point crossover” [33] technique, as the selection of an individual transformation targets a specific set of genes in the parent chromosome.

Fig. 6 illustrates the crossover procedure. Reproduction selects two parents from the population. The chromosomes are scanned to determine which transformations have been applied; parent 1 p1 has received a pipelining transformation, and parent 2 p2 has received a retiming and a pipeline transformation (these would be identified by prefixes on the affected genes, not shown in the figure). The pipeline of p1 is applied to p2 to produce child 2. The retiming operation of p2 is selected randomly and applied to p1, producing child 1. Each crossover operation applies one transformation to each parent. Where a transformation attempt
is rejected because the destination parent already has that transformation, a different transformation is selected randomly from the source parent.

3) **Reversible Transformations**: Initial experiments found that pipelining is a useful transformation that can create large increases in fitness. A pipeline operation on a chromosome confines all descendants of that chromosome to be pipelined designs. Consequently, the GA could become stuck in local optima. In effect, pipelines are inserted continually into chromosomes without any means of removing them, producing populations dominated by pipelined solutions. This occurs because the power reduction achieved with pipelining produces a large leap in fitness compared to the nonpipelined solutions; these pipelined solutions then dominate the natural selection process. GALOPS uses a reverse-pipeline operator implemented as an HLT for the removal of pipelines from pipelined designs. This helps prevent the GA from confining its search to one area of the solution space. Reverse operators were also designed for the retiming transformation, so that the “back” retiming operation moves all delay elements from the output of an element to its inputs.

V. **SELECTION OF OPERATOR RATES**

This section describes the experimental techniques used to determine a set of operator rates for the GALOPS system, a typical problem in any GA implementation. GALOPS mutation operator is the core technique for introducing diversity into the GA population through the HLTs. Therefore, the application rate of the mutation operator is seen as a critical parameter as it is the key mechanism by which new power-reduction properties are introduced into the population. In contrast, the crossover operator is focused on the manipulation of existing transformed structures to improve fitness performance.

The mutation operation in this synthesis problem consists of a number of suboperations (i.e., the individual HLTs) which, when applied at the correct rates for each suboperation, will guide the GA toward synthesizing optimum designs. In addition to individual application rates, the correct combination of rates is crucial for the optimal performance of the GA as the transformations are interactive, enabling or disabling the subsequent application of each other. We have carried out a number of investigations, detailed in this section, to determine an appropriate set of rates for optimal GA performance.

An exhaustive experimental analysis, where each possible combination of suboperator application rates is analyzed, would be a complex and time-prohibitive task. Therefore, the problem has been tackled by forming a number of mutation sets, each of which contains a set of suboperator application rates. The total number of suboperator application rates (and, hence, the number of mutation sets and required analyses) is limited by not considering small fluctuations in application rates, as they are not expected to have a large impact on the GA. To further reduce the complexity of the task, transformation heuristics are used to define the mutation sets. For example, retiming is applied at a higher rate than pipelining because pipelining is expected to incur a greater area penalty in comparison to retiming. Retiming is seen as an enabling transformation that greatly increases the potential application of other transformations.

Table I shows seven mutation sets of potential application rates expressed as probability percentages. Each set is applied to each of five benchmark designs discussed in Section VII-A, chosen to present a range of topologies and signal-processing applications to minimize the dependence of the determined values on the type of design.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Mutation Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Rates (%)</td>
<td></td>
</tr>
<tr>
<td>Retime</td>
<td>1</td>
</tr>
<tr>
<td>Back Retime</td>
<td>1</td>
</tr>
<tr>
<td>Pipeline</td>
<td>1</td>
</tr>
<tr>
<td>Automatic Pipeline</td>
<td>1</td>
</tr>
</tbody>
</table>

Sets 1–3 apply an approximate geometric progression to the application rates from 1 to 10. Sets 3–6 apply the retiming transformations at greater rates than the pipelining transformations in accordance with the transformation heuristics discussed previously. This set of potential application rates is a small sample of the possible combinations. Therefore, the determined application rates may not be optimal. However, the chosen sets allow for the analysis of the effect of different application rates without requiring a prohibitive number of combinations.

Each set is applied to each of the five benchmark DFG designs 50 times, resulting in 1750 (50 × 5 × 7) complete runs of the GA. Each design is synthesized 50 times, resulting in 1750 (50 × 5 × 7) complete runs of the GA. The graph in Fig. 7 demonstrates the effect of each set on determining the best solution.

Each complete run of the GA produces a design with the best found power reduction (reported as a percentage of the original power consumption). The best value from all 50 runs of a single design is taken as the best power reduction obtained for that design with the current set of application rates. To allow the results of all designs to be compared (as each design will produce different power reductions), the power reduction values for each design are normalized with respect to the worst result found for
that design, in this case, the design found with mutation set 1. A higher value on the y-axis indicates a design superior to the worst. The graph shows that set 1 produced the worst results for all of the designs. The performance of set 1 is so poor that the results of sets 2-7 appear indistinguishable.

However, the graph of Fig. 8 illustrates the results of sets 2–7 (i.e., set 1 omitted). In this graph, the $y$ values are normalized with respect to the worst design found with sets 2–7. Hence, there is improved clarity for analysis as the results of set 1 are so poor. This graph illustrates that all designs, with the exception of the FIR3 design, are affected by the different application rates. The FIR3 is unaffected because it is a relatively simple design that requires a few optimization steps.

It is clear that all of the designs have the best power reduction in set 6. Note that although some of the designs also receive the best power reduction in other sets, set 6 is the only set where all of the different designs receive the best power reduction. Therefore, the application rates of set 6 are chosen as the set of application rates for GALOPS.

The chosen set of rates applies the retiming transformations at a significantly greater rate than the pipelining transformations, supporting the heuristic assumptions made in analyzing the properties of the transformations for their optimization capabilities. This illustrates the potential of the GA synthesis tool as a means of analyzing the interaction between various optimization techniques.

VI. FITNESS EVALUATION

The GA uses a power-estimation module to assess the fitness of each design. The precise calculation of power consumption (using transistor-level circuit models) is extremely time consuming, especially in the case of the complex designs synthesized in this paper which will typically contain hundreds of thousands of transistors. These "traditional" power estimation techniques can take days to analyze a single design [34], a prohibitive timescale within a GA fitness function that is used many times in even a single generation. Therefore, the power estimation module needs to produce results relatively quickly to ensure
that designs are synthesized in a reasonable amount of time. An alternative strategy to accurate calculation is to use power estimation techniques that produce results quickly, while maintaining a required level of accuracy. In the case of power estimation for design exploration, consistency is more important than accuracy. A GA selects designs on the basis of comparative fitness; therefore, the actual power consumption of a design is not as important as its power consumption relative to the other designs in the population. This presents some scope for the simplification of the power estimation process, but the calculation of relative power consumption is still a complex time-consuming task.

This section presents the problem of consistent and fast high-level power estimation before describing the estimation module used in the genetic synthesis system.

A. High-Level Power Estimation

Traditionally, power estimation of VLSI systems has been performed at lower levels of the design process, such as the logic-gate level [34]. Even at this lower level, power estimation presents a complex problem. This complexity is increased greatly at higher levels of the design process because many of the factors that contribute to power dissipation such as area and capacitance are not finalized until lower levels of the design process. Taking each candidate design through the complete design process, to use accurate low-level estimation tools, would be prohibitive in terms of computation time. Power estimation needs to be performed quickly at the high level using parameters that can be determined from the high-level DFGs.

The estimation methods presented in [34] and [35] use signal correlation statistics (extrapolated from the layout of the DFG) and switching activity (obtained from simulations of the DFG) to determine accurate power estimations based on the design’s intended signal-processing function. This method uses the fact that the switching activity of an implemented design is related to its application. For example, the activity of speech signals is very different from that of television signals, although they may use the same filter design. Although this method can produce accurate results, it does limit the flexibility of the final design as power estimations are based on a particular application of that design.

The power factor approximation (PFA) technique [36] uses a method of characterizing the functional elements of the DFG (adders, multipliers, etc.) for power. The high-level power estimation is extrapolated by summing the individual power requirements of each functional unit required to implement the DFG. The PFA technique is an accepted high-level power estimation technique, producing results quickly to enable comparative analysis of alternative designs rather than precise calculation of the consumption of individual designs. The GALOPS estimation module is built on the PFA technique and is outlined in the following sections.

B. The Fitness Function

As the aim of GALOPS is to produce low-power designs, the objective function of the GA is the power dissipation equation presented in (1). A lower power design is assigned a higher fitness so the fitness function is the inverse of this objective function. This is shown in (2), where $F$ is equivalent to the inverse power of the design

$$ F = \frac{1}{V_{DD} \times C_e}. $$

To aid in the comparison of different designs, all fitnesses are scaled relative to the original design. Equation (3) shows the fitness ($F'$); $o$ denotes the original design and $n$ the design under consideration. Using this equation, all designs with a lower power than the original have a fitness of greater than 1

$$ F' = \frac{V_{DDo} \times C_{eo}}{V_{DDn} \times C_{en}} = \frac{\text{Power of Original Design}}{\text{Power of Current Design}}. $$

There are two main components of the power estimation module: supply voltage estimation and capacitance estimation. These are combined to give the total power and, hence, the fitness. Fig. 9 illustrates the fitness evaluation procedure.

1) Supply-Voltage Estimation: The initial task of estimating the supply voltage is the calculation of the length of the DFG’s critical path. This is compared with the length of the original design’s critical path to determine the speed ratio of the two designs. A piecewise-linear model of Fig. 1 (as used in [22]) is used to calculate the speed ratio to determine a supply voltage that will produce a design with no speed increase over the original design (unity speed ratio). This piecewise-linear model assumes that all candidate designs meet their speed requirements at an initial supply voltage of 5 V.

To illustrate the operation of the supply-voltage estimation module, consider a design with an initial critical path length of 4. The “calculate critical path” procedure determines that the new design under consideration has a critical path length of 2. A speed ratio of 2 is calculated by the “calculate speed ratio” procedure and this ratio is passed to the “calculate supply voltage” procedure. The supply voltage calculation uses the piecewise-linear model of supply voltage against delay discussed previously, using the given delay factor (the inverse of the speed ratio) to calculate the resulting supply voltage determined by the model. This procedure returns a supply voltage of 2.9 V, the voltage at which all delays in the design increase by 2. This increased delay negates the speed increase,

![Fig. 9. Overview of the fitness evaluation procedure.](image-url)
resulting in a design with a lower supply voltage, but the same computational throughput.

2) Capacitance Estimation: Switched capacitance estimation is far more complex than supply voltage estimation. It is a combination of the switching activity and the physical capacitance of the fabricated design; both of these parameters are difficult to estimate at the high level. As mentioned previously, GALOPS uses a capacitance estimation technique similar to PFA [36]. Low-level functional units are characterized for capacitance and area using 1-μm CMOS device technology. These designs are placed in a library for access by the power estimation procedure. The total physical capacitance and area are calculated by determining which functional units are required to implement the DFG and mapping the library designs onto them.

The capacitance estimation module also estimates the contribution of interconnect capacitance to the total capacitance. This is the capacitance of buses, wires, etc., required to implement the DFG. Interconnect capacitance is estimated from a statistically derived model [22] shown in Fig. 10, which uses the total area estimation. This model illustrates that an \( N \)-fold increase in area results in an approximate \( N^2 \) increase in interconnect capacitance. The “calculate capacitance” procedure in Fig. 9 combines the functional unit and interconnect capacitance estimation.

The current implementation of GALOPS does not include the estimation of control-logic capacitance. While this cannot be discounted, within the data-path intensive architectures discussed in this paper, the data-path units will account for the majority of the total capacitance [35].

The fitness function satisfies the goals of enabling comparison between high-level designs while maximizing the speed of the complex power estimation process. The PFA technique is based on the concept of enabling comparisons between alternative design solutions rather than the precise computation of a single design’s power consumption. The accuracy of the approach in determining the difference in power consumption between alternative designs [36] ensures that the fitness function produces a reasonable representation of the low-power solution space to enable qualitative selection strategies to be used.

VII. IMPLEMENTATION AND RESULTS

GALOPS is written in C and comprises over 5000 lines of code. The tool is compiled and executed on a Pentium Pro 200 MHz system with 64 MB of RAM. Different benchmark designs have different memory requirements. However, the designs presented here were all optimized within 16 MB of RAM.

A. Low-Power Design Benchmarks

The performance of GALOPS is illustrated with a variety of practical signal-processing designs. These benchmark designs are intended to provide researchers with a set of designs for the comparison of VLSI signal-processing synthesis techniques. The benchmark set comprises practical DSP algorithms of varying complexity, from simple feedforward filters to discrete transform algorithms. The benchmark set is chosen to incorporate designs of varying complexity, from the simple three-tap finite-impulse response (FIR) filter with five operations to the direct form of a recursive 8th-order Avenhaus filter with 34 operations. Both recursive and nonrecursive designs are included to provide a comprehensive test set for GALOPS.

Specifically, the benchmark set consists of the following:

- FIR3, FIR8, and FIR11 are 3rd-, 8th-, and 11th-order FIR filters, respectively;
- AV6D and AV8D are 6th- and 8th-order direct forms of the Avenhaus filter [22] that contain complex feedback paths;
- AV8P is a parallel form of an 8th-order Avenhaus filter, this parallel form also contains complex feedback loops, but it has a different DFG topology from that of the AV8D. The Avenhaus filters are complex structures that are commonly used in real-world signal-processing applications;
- Discrete cosine–sine transformation (DCST) is a combined discrete cosine transformation (DCT) and discrete sine transformation (DST) block [37]. The DCT and DST transformations are used for data compression of digital signals;
- LAT is a 2nd-order lattice filter that comprises two feedback stages;
- VOLT is a complex Volterra filter with a large internal feedback loop.

B. Optimization Without Unfolding

Unfolding is a useful transformation that can produce designs with low-power characteristics. However, its application incurs an area overhead. It is not always desirable to minimize power at the expense of this increased VLSI device area that unfolding can produce. Table II gives power consumption obtained as a percentage of the original design by using the proposed GA approach without the application of unfolding. Each design is synthesized 50 times; the results reported are the median values for each design as this represents a practical power consumption that can be achieved. The variance in achieving this median value was relatively low for all designs.

The “estimated size increase” column gives data for the area increase necessary to implement the low-power design as a percentage of the size of the original design. Supply voltage is the new \( V_{DD} \) for that design. The execution time and number of generations are provided to indicate the number of evaluations and length of time taken for GALOPS to produce the optimized design.
In all but one case, power consumption was reduced to less than 60% of the original design by using the proposed GA approach. This significant power reduction was achieved in spite of the maximum area increase for each design being less than 10%. As an example, the GA produced an FIR3 design with a 4% increase in area; power consumption was reduced to 19.2% of the original design by using the proposed GA approach. The power reduction was achieved by lowering the supply voltage from 5 to 2.14 V.

With the exception of the Volterra filter, the synthesized designs offer significant power savings through a reduced operating voltage. The suggested supply voltages for each design are significantly different from the industry standards of 5 V, and more recently, 3.3 V. Typically, practical designs are synthesized to produce power consumption within acceptable limits. This enables the designer to select supply voltages that may be higher than the minimum, but that do not require nonstandard voltages (e.g., a suggested voltage of 3.2 V actually using a standard 3.3 V supply).

However, where nonstandard voltages are required, they can be provided through the implementation of small-area power-efficient voltage-level converters on the integrated circuit (IC) [38]. The IC externally requires 3.3 V, but the on-chip voltage converter supplies the required low-power voltage to the device core. Such systems have been used to implement practical fabricated signal-processing algorithms, for example, in a low-power portable multimedia terminal [25]. An alternative scheme is used by the latest generation of mobile Pentium processor chips. While operating from a 3.3 V power supply, the chip has a core voltage of 2.45 V to reduce power consumption [39]. The required low voltage for the core is provided with the addition of an extra supply voltage input instead of using an on-chip voltage converter. Recent research [40], [41] highlights the use of multiple voltages on a single chip to optimize the power consumption and speed of individual functional units. A more recent development is the dynamic configuration of these on-chip voltage converters to produce the lowest power consumption for the current speed requirements.

The effect of the complexity of the designs on the run time of the synthesis process is also illustrated by the results. The higher order (more complex) FIR filters require more generations to produce designs. This is primarily due to the increase in the number of possible transformation moves, and hence solutions for each design. The increased run time is also due in small part to the extra computation required to evaluate the fitness of the more complex designs.

### Table II

<table>
<thead>
<tr>
<th>Design</th>
<th>Power Percentage (%)</th>
<th>Estimated Size Increase (%)</th>
<th>Supply Voltage (Volts)</th>
<th>Number of Generations</th>
<th>Execution Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR3</td>
<td>19.2</td>
<td>3.99</td>
<td>2.14</td>
<td>1</td>
<td>0.07</td>
</tr>
<tr>
<td>FIR8</td>
<td>9.4</td>
<td>6.94</td>
<td>1.47</td>
<td>260</td>
<td>67.18</td>
</tr>
<tr>
<td>FIR11</td>
<td>11.8</td>
<td>4.11</td>
<td>1.68</td>
<td>253</td>
<td>113.10</td>
</tr>
<tr>
<td>AV6D</td>
<td>21.3</td>
<td>0.85</td>
<td>2.30</td>
<td>174</td>
<td>40.51</td>
</tr>
<tr>
<td>AV8D</td>
<td>16.2</td>
<td>1.13</td>
<td>2.00</td>
<td>300</td>
<td>81.26</td>
</tr>
<tr>
<td>AV8P</td>
<td>21.4</td>
<td>1.34</td>
<td>2.30</td>
<td>160</td>
<td>129.58</td>
</tr>
<tr>
<td>DCST</td>
<td>30.3</td>
<td>0.29</td>
<td>2.75</td>
<td>8</td>
<td>1.08</td>
</tr>
<tr>
<td>LAT</td>
<td>59.6</td>
<td>0.57</td>
<td>3.85</td>
<td>2</td>
<td>0.17</td>
</tr>
<tr>
<td>VOLT</td>
<td>100</td>
<td>-</td>
<td>5.00</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Design</th>
<th>Power Percentage (%)</th>
<th>Estimated Size Increase (%)</th>
<th>Supply Voltage (Volts)</th>
<th>Number of Generations</th>
<th>Execution Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR3</td>
<td>5.2</td>
<td>808.35</td>
<td>1.15</td>
<td>333</td>
<td>83.32</td>
</tr>
<tr>
<td>FIR8</td>
<td>9.0</td>
<td>399.00</td>
<td>1.47</td>
<td>665</td>
<td>260.06</td>
</tr>
<tr>
<td>FIR11</td>
<td>8.9</td>
<td>396.82</td>
<td>1.47</td>
<td>854</td>
<td>468.70</td>
</tr>
<tr>
<td>AV6D</td>
<td>21.3</td>
<td>0.85</td>
<td>2.30</td>
<td>160</td>
<td>40.51</td>
</tr>
<tr>
<td>AV8D</td>
<td>16.2</td>
<td>1.14</td>
<td>2.00</td>
<td>303</td>
<td>81.26</td>
</tr>
<tr>
<td>AV8P</td>
<td>21.4</td>
<td>1.34</td>
<td>2.30</td>
<td>175</td>
<td>129.58</td>
</tr>
<tr>
<td>DCST</td>
<td>30.3</td>
<td>0.30</td>
<td>2.75</td>
<td>8</td>
<td>1.08</td>
</tr>
<tr>
<td>LAT</td>
<td>59.5</td>
<td>199.43</td>
<td>3.85</td>
<td>277</td>
<td>28.24</td>
</tr>
<tr>
<td>VOLT</td>
<td>100</td>
<td>-</td>
<td>5.00</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
feedback paths, and hence has a small decrease in power due to the unfolding transformation. During the synthesis procedure, unfolding of the larger Avenhaus filters produces designs larger than the maximum size specified as an implementation limit. These larger designs are not selected by the genetic algorithm as valid power-reduced implementations; hence, unfolding does not produce any decrease in the power of the Avenhaus designs compared to that found without unfolding.

The increased execution times of the examples that have benefited from unfolding are due to the postponing principle explained previously. This principle holds off the application of unfolding until the best fitness of designs within a given population has not increased for a number of generations, hence increasing the execution time. Fig. 11 illustrates the effect of postponing with the fitness profiles obtained for the FIR8 filter; the two curves illustrate the performance in the cases of application and nonapplication of unfolding. The best found nonunfolded design is obtained before generation 400; subsequent evolution does not improve the design. The postponing principle holds off unfolding until the fitness is steady for a number of generations, then applies unfolding at approximately generation 625, increasing the fitness of the nonunfolded design.

All area estimations are based on a 1:1 allocation procedure [3], where each functional operation within the DFG is implemented with a single execution unit on the VLSI implementation. The 1:1 technique is well suited to the exploitation of maximal parallel processing for low power, but it does have a larger area overhead than other allocation techniques.

In the case of optimization without the application of unfolding, the FIR3, DCST, and LAT designs have comparatively smaller power reductions than some of the more complex designs. The results presented report the power reduction achieved compared to the original power consumption. As the smaller designs have a relatively shorter initial critical path, the low-power synthesis techniques do not have as much scope for critical path reduction of these designs as compared to the larger designs. These smaller designs would have lower absolute power consumption than the more complex (larger) designs.

The blank entries in the columns of the VOLT filter indicate that no power reduction was obtained. This is because the critical path of this filter consists of a complex feedback loop. The transformation set used by GALOPS is incapable of reducing the length of this loop. This is a limitation of the current version of GALOPS, which may be addressed by the inclusion of more HLTs in the transformation library.

The issue of comparing GALOPS results with other systems has been discussed previously in Section II. A direct comparison between the few research-based systems is not feasible at these early stages of low-power design through high-level synthesis. This is due primarily to the different numbers and types of designs analyzed, the different power analysis/optimization techniques, and the different target VLSI implementations. However, to put the results in perspective, typical percentage power reductions obtained with the HYPER-LP system are used for relative comparison. For the three DSP systems optimized by Chandrakasan et al. [22], power was reduced to an average of approximately 12%, with an associated average area increase of approximately 500%. GALOPS reduced the power of nine DSP systems, which range in complexity from comparable systems to those presented in [22] to significantly more complex and larger systems. Power was reduced to an average of approximately 30%, but with a smaller area increase of approximately 225%. Although the results are based on significantly different designs, utilized different optimization techniques, and different transformation sets, they serve to illustrate the magnitude of power reduction and associated area increases that are obtained with high-level power-reduction techniques.

D. Evolution of Multiple Solutions

One advantage of using a GA-based synthesis approach is the capability of producing multiple designs that satisfy the performance characteristics specified by the designer. These designs have a different topology, but the same power consumption requirements. The designs have different scheduling and functional unit requirements, which gives greater flexibility in the subsequent stages of the VLSI design process. This feature of GALOPS enables the expert knowledge of the design engineer to be incorporated into the synthesis process.

After the fittest design has been found, natural selection favors its reproduction over the less-fit members of the population. Therefore, the exploration of the search space is concentrated in the area of the fittest design, increasing the production of similar designs. Fig. 12 gives an example of the richness of the GALOPS population of designs throughout the evolution process.
The graph depicts the number of unique solutions within a population size of 500 that have the best power-design characteristics found throughout evolution, but different VLSI implementation characteristics. These results are for the synthesis of the 8th-order Avenhaus filter. For this run of GALOPS, the best design was found on generation 114; in subsequent generations, the number of unique fittest designs in the population increased to a maximum of 35 (with an average of 18).

The number of unique designs in the population varies over the course of the evolution. This is because these unique designs have the same fitness; therefore, there is the same selective pressure on each one irrespective of their uniqueness. This fluctuation in the number of unique designs could be improved by modifying the elitism operator to copy each unique design across to the next generation, ensuring a steady increase in the number of unique designs. However, this may encourage the GA to get stuck in local optima during the evolution process as suboptimal but currently fittest designs could produce a large number of unique designs that would dominate the population.

The current implementation of GALOPS stores the unique designs produced in each evolution. This preserves the unique designs without altering the search process.

VIII. Conclusion

A genetic algorithm approach to high-level low-power synthesis has been introduced. The GA uses nonstandard chromosome representation with problem-specific genetic operators developed for the synthesis tool. The flexibility of the chromosome representation ensures that the GA is capable of synthesizing signal-processing designs of varying complexity and size.

The problem-specific operators modify the standard genetic crossover and mutation techniques in order to apply standard DSP design transformations within the GA framework, to ensure that design functionality is preserved. This modification of the standard genetic operators incorporates expert DSP VLSI design knowledge into the GA-based synthesis tool.

The GA successfully searches the complex space inherent in high-level low-power synthesis. The results presented for a variety of signal-processing architectures demonstrate that the GA is an effective tool for DSP synthesis.

One advantage of the GA-based synthesis technique used by GALOPS is the ability to exploit a GA’s inherent parallelism. A GA samples a number of points in the search space throughout its exploration. In the case of the low-power synthesis problem presented here, this creates a number of designs with the same power-consumption requirements, but different VLSI-implementation characteristics. This provides the design engineer with added flexibility.

The use of a transformation library to supply the GA provides a framework for the addition of new transformations to investigate their power-reducing applications or to determine the effect of interaction between transformations on providing the best solution. The transformations used together with the application rates determined through experimental results and analyses produce low-power designs without affecting design function or suffering loss in performance.

REFERENCES


Marc S. Bright received the B.Eng. and Ph.D. degrees from the Cardiff University, Wales, U.K., in 1995 and 1998, respectively.

He is currently a Design Engineer with the Broadcom Corporation, Bristol, U.K. He was with the Electronics Circuits and Systems Research Group of Cardiff University. He is a Member of the Agilent (formerly Hewlett Packard) Center for High Frequency Engineering and the Director of the ERDF Center For High Frequency Engineering Support. His research interests focus on wireless digital communications, including the development of digital signal processing solutions, the use of heuristic techniques in electronic design, computer-aided design tool development, and power amplifier linearization.

Tughrul Arslan (M’98) received the B.Eng. and Ph.D. degrees from the University of Hull, Hull, U.K., in 1989 and 1993, respectively.

He is currently a Senior Lecturer in the Department of Electronics and Electrical Engineering at the University of Edinburgh, Scotland, U.K., where he leads the digital VLSI activity. He is also a Lecturer at the Institute for System-Level-Integration, Livingston, Scotland, U.K. His research interests include low-power VLSI and system-on-chip design, high-performance DSP architectures, and evolutionary algorithms and their application to circuit design.

Dr. Arslan is a Member of the Institution of Electrical Engineering (U.K.).