A Genetic Approach to Gateless Custom VLSI Design Flow

Mohammed Shoaib, Noor Mahammad Sk and Kamakoti.V
Reconfigurable and Intelligent Systems Engineering Group
Department of Computer Science and Engineering
Indian Institute of Technology, Madras, Chennai - 600036, India

Abstract— **In this work, we propose a novel technique for evolving transistor netlists from truth table descriptions of arbitrary digital circuits. The proposed methods incorporate the effective use of Genetic Algorithms (GAs). In typical semi-custom and custom design flows, logic optimization is done at the gate level after Boolean translation of the input truth table. The final transistor netlist is then deduced from the simplified gate logic to be laid out on a chip. However transistor level optimizations after the boolean simplification step would still not lead to the minimum number of transistors. This final optimization level is non-existent in present custom design flows. This work aims to address this need. A salient feature of the proposed technique is the bypassing of gate level representation and optimization in the VLSI design flow. We provide genetic methods to directly optimize truth table inputs using transistor level simplification. This eliminates the intermediate gate level optimization step and provides optimized transistor netlists which could be used for dynamic library cell generation for custom and semi-custom designs on the fly.**

I. Introduction and Background

Automation in modern microelectronics has resulted in multiple methods for the design and optimization of digital circuits. These automated methods incorporate gate level optimizations followed by the use of standard library cells to map the designs to hardware. In several custom designed Integrated Circuits (ICs), the use of standard libraries translates to hardware redundancy. This is because, standard library cells, due to their limitations in size and number, often result in having several unused transistors for the custom logic. Consequently, the use of tailored cells for specific applications become necessary. Application specific generation of such cells utilizes on-chip hardware effectively besides providing fast and flexible designs operating with lesser redundancy and better performance.

Realization of custom circuits - cells or blocks, involve three implementation phases [1]:

- Creation of transistor circuit topologies which provide a specific digital function.
- Sizing and ordering of the transistors in the circuit topology.
- Placing, routing and compacting the transistors in layout.

Each of the above stages involve trade-offs which must be optimized across all stages. This work proposes to address the first phase of transistor circuit topology creation, automatically.

In literature, much attention has been given to the sizing [2], [3] and placing of transistors [1], [4], [5] in custom and semi-custom circuits. Using Genetic Algorithms (GAs), in [4], Murphy, et. al, describe the placement optimization of a cell followed by the *extraction of* the netlist. They employ sigmoidal transistor characteristics for an Artifical Neural Network (ANN) model unlike the switch characteristics in our case. Their major aim is the use of primitive components and reduction of the parasitic capacitance rather than a topological optimization. In [5], Bahuman, et. al, describe a GADO model for a custom cell. But their starting point is the placement optimization unlike the configuration optimization in our case.

In this work, we focus on the direct transistor netlist generation using Genetic Algorithms for optimization. Genetic methods have been applied in the past for gate level synthesis [6] besides specific optimization methods for Pass Transistor Logic (PTL) [7], Complementary PTL (CPL) [8], Differential PTL (DPTL) [9], Double PTL (DPL) [10] and other non-complementary MOS logic styles. Mazumder and Rudnick, in [11] provide a good insight into the problems in VLSI design and synthesis techniques.

The methods we use in this work involve the use of Genetic Operators to evolve transistor netlists for a certain functional requirement described by an input truth table. The netlist generation, because of the characteristics of the genetic operators, inherits direct optimization at the transistor level. This gives an optimized transistor netlist which would otherwise be derived after converting the truth table to its minterms and then applying the Boolean simplification operations. In the latter case, the netlist obtained would still not be optimized to exploit internal topological optimizations of the transistors. The main contribution of this work is to propose the genetic methodology of direct evolution for the transistor netlist from functional descriptions of cicuits. The paper also provides a methodology for incorporating a gateless optimization algorithm in the custom circuit design flow to provide the creation of custom library cells *in-situ*, Fig. 1. This favorably enhances the performance of custom circuit syntheses and provides better utilization of transistor resources besides automation and simplification of the design flow, by eliminating the boolean optimization step. In the following paragraph, we present the basic concepts of Genetic Algorithms [12] necessary to understand this paper.

Genetic algorithms work on a set of *chromosomes/genotypes* called the population. Each chromosome represents a solution to the problem which is associated with a *fitness* value that reflects how good it is compared

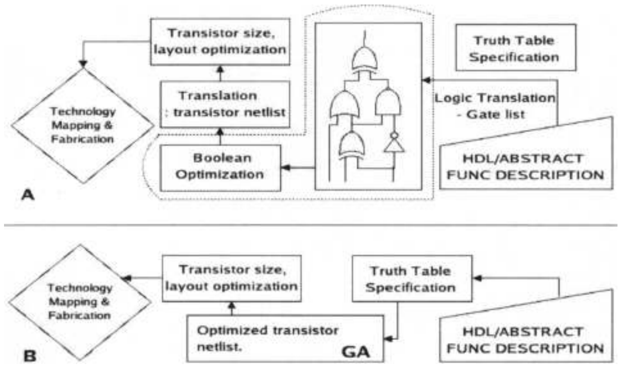
Transistor size, layout optimization
Technology Mapping & Fabrication
Translation : transistor netlist
Boolean Optimization
Truth Table Specification
Logic Translation - Gate list
HDL/ABSTRACT FUNC DESCRIPTION
A
Transistor size, layout optimization
Truth Table Specification
Technology Mapping & Fabrication
Optimized transistor netlist.
GA
HDL/ABSTRACT FUNC DESCRIPTION
B

Fig. 1. Custom design flow simplification using the proposed genetic method : The genetic flow eliminates the intermediate Boolean simplification step from the normal gate optimized flow (A) providing a direct transistor optimized netlist (B), ready for size/layout tuning in custom circuits.

TABLE I

GENE STRUCTURE IN A CHROMOSOME USING .SIM ENCODING [a]

T_{ij}	S_{ij}	n^1_{ij}	n^2_{ij}	W_{ij}	L_{ij}	f_i	δ_{ij}
p	CLK	V_{dd}	1	4	2		
n	$R_{ij}(A_1..A_N)$	1	$R_{ij}(2..\overline{N+1})$	4	2		
n	$R_{ij+1}(A_1..A_N)$	$R_{ij+1}(S_{ik}..S_{ij-1})$	$R_{ij}(2..\overline{N+1})$	4	2		
.	.	.	.	.	.		
.	.	.	.	.	.		
n	CLK	$max[n^{1,2}_{ij} \ldots n^{1,2}_{ij+N}]$	gnd	4	2	f_i	δ_{ij}

[a] T_{ij}: Transistor type, S_{ij}: Transistor gate node, W_{ij}: Transistor width in μm, L_{ij}: Length in μm, N: Number of Inputs, R_{ij}: Uniform Random Selector

to the other solutions in the population. The variation process comprises of *crossover* and *mutation*, which concoct material by partial exchange among *genotypes* and by random alterations of data strings. The frequency of these operations is controlled by certain pre-set probabilities which require heuristics appropriate for the particular problem at hand. The *representation, variation, evaluation* and *selection* operations constitute the basic GA cycle or *generation*.

II. GENETIC TOPOLOGICAL SYNTHESIS

A. Representation and initial population

Gene representation in the Allele: Representation of the genes for evolution is a critical choice to keep the circuit topology valid and provide faster convergence. Eq. (1) shows the chosen representation. Each transistor is represented as a triplet $< S_j, n^1_j, n^2_j >$, where S_j stands for the node to which the input signal and the gate of the j^{th} transistor are connected. n^1_j and n^2_j are the nodes to which the source and drain of the j^{th} transistor are connected. A *chromosome* is a sequence of such triplets which are equal in number to the input signals determined from the truth table. These form netlist inputs to the *IRSIM* simulator for fitness evaluation and selection. A subsequence of these signal triplets can be used for mapping the inputs (or transistor gates) to one of the variables $A_1, A_2 \ldots A_N$. In other words, the S_j values of the triplets in the subsequence shall be a one-one mapping from among A_i's, $0 \leq i \leq N$.

Gene Structure: The internal gene structure of a chromosome is shown in Table. I. The representation using a *Perl parser* is designed to conform to the spice netlist. This can be directly used for evaluation, using the *IRSIM* switch level simulator.

$$\left[S_j\ n^1_j\ n^2_j \,\middle|\, S_{j+1}\ n^1_{j+1}\ n^2_{j+1} \,\middle|\, \ldots\ldots \,\middle|\, S_{j+N-1}\ n^1_{j+N-1}\ n^2_{j+N-1} \right] \quad (1)$$

The methodology for the generation of the **initial population**

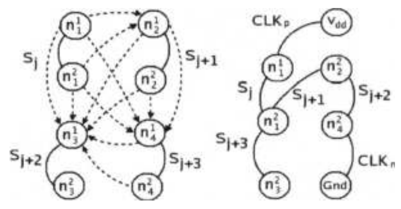

Fig. 2. Initial population generation with the corresponding transistor topology and SFG for the netlist. The dotted lines contribute to the random node choice for the next chromosome. The final obtained netlist is functionally valid.

is shown in Fig. 2 and in Table. I. The first node, n^1_0, for the i^{th} chromosome in the population is chosen to be 1 to which the *pMOS*, *CLK* signal is connected. The second node is randomly chosen between (N+1) and 2. These form the source and the drain for the first *nMOS* transistor. For the second and subsequent j^{th} device, the first node is chosen randomly from among the previously chosen nodes, $< n^1_{ik}, n^2_{ik} >$, where $0 \leq k \leq j$ and the second node is chosen randomly between (N+1) and 2. This ensures circuit connectivity and appropriate intermixing of the nodes to provide a broad outreach in the search space. A matrix of P chromosomes is chosen this way to form the valid initial population.

B. Variation: Crossover and Mutation

Algorithm 1 CROSSOVER(n)

Require: An integer $0 \leq n^{1,2}_{ij} \leq N + 1$
Ensure: Network connectivity and $n^1_{0j} = 0$.

```
1: for all $i$ such that $0 \leq i < P$ do
2:     for all $j$ such that $0 \leq j < \delta_j$ do
3:         $S^{'}_{ij} = R(S_{ij} \ldots S_{ij+N-1})$
4:         $n^{1,2}_{ij} = \xi(n^{1,2}_{ij})$
5:         if $j = 0$ then
6:             return $n^1_{ij} = 1$
7:         end if
8:     end for
9:     for all $k$ such that $\delta_j \leq k < N$ do
10:        $S^{'}_{ij} = R(S_{ij+1} \ldots S_{ij+1+N-1})$
11:        $n^{1,2}_{ij} = R(n^{1,2}_{ij} \ldots n^{1,2}_{ij+\delta_j})$
12:        $n^2_{ij} = \xi(n^2_{ij+1})$
13:    end for
14: end for
```

The initial population generated as described in Sec. II-A is used to proceed with the evolutionary variation. **The crossover** operator algorithm is shown in Algorithm.1. For the j^{th} chromosome in a population size of P, δ_j transistor selections from the j^{th} chromosome and $(N\text{-}\delta_j)$ selections from the $(j{+}1)^{th}$ chromosome are used. The copy operator, ξ is used to copy the corresponding nodes for the δ_j devices. The first node is then set to 1 as in Sec. II-A to ensure circuit connectivity in the *phenotype*. For the rest of the $N\text{-}\delta_j$ transistors, the first node is chosen randomly from among the previous $2\delta_j$ nodes. The second node for all the genes in the chromosome other than the first one is copied (ξ) from the $(j{+}1)^{th}$ chromosome. The select-scalar, δ_{ij} is calculated as shown in Eq. (2).

$$\delta_{ij} = log_2 l - \left\lceil \frac{f_i}{log_2 l} \right\rceil \tag{2}$$

where, l is the length of the truth table input by the user and f_i is the fitness value for the i_{th} *chromosome* set in the *genotype* matrix. **The mutation** operator works similar to the initial population generation described in Sec. II-A. These randomly mutated and the varied chromosomes $(P_{co/m})$ total to P-1. These are appended at the end of the chromosome matrix with P elements to obtain a total of $2P$-1 chromosomes in the next generation.

C. Selection and termination:

Algorithm 2 SELECTION(n)

```
Require: An integer $0 \leq n_{ij}^{1,2} \leq N+1$.
Ensure: P: Population; G: Generation
1: for all $g$ such that $1 \leq g < G$ do
2:     $P_g = P_i + P_{co/m}$
3:     $f_g = \varepsilon(P_g)$ {irsim $*.proc\ g_j.sim\ g_j.cmd$}
4:     $P_{ng} = \Lambda[P_g(1 \ldots P)]$ {sort and select P}
5:     if $g = 0.1G$ then
6:         $P_g^{'} = P_i + P_m$ {variation mutation}
7:         $P_g^{'} = \Lambda[P_g(1 \ldots P)]$ {sort select}
8:     end if
9: end for
```

With the variation generated $P_{co/m}$=P-1 chromosomes appended at the end of the initial population, P_i=P, a new *genotype* matrix is created. For all of these, the **fitness** (ε) is evaluated. This is an $\oplus$ (*XOR*) operator with the input truth table supplied by the user. This compares the deviation of the current chromosome from the required truth table. The ε is evaluated using the ***IRSIM*** simulator. From the resulting $2P$-1 chromosomes generated, the sorting operator Λ, sorts the chromosomes according to their fitness values and the top P of them are selected to move on to the next generation. This way, only the best characteristics of a generation are passed on to the next generation. After about 10% of the total generation size G, mutation is introduced to increase the generation gap and introduce diversity in the current population. When the best fit chromosome is found from the sorted matrix, the algorithm is **terminated**. The selection operation is algorithmically described in Algorithm.2. An elitist model is also used in the design.

III. EXPERIMENTAL RESULTS

Test case simulations for the proposed design flow were run using an embedded switch level *IRSIM* simulator. Fig. 3 shows the evolved fitness values over the iterations numbers for a test input truth table whose boolean functions are shown in Fig. 4 and Fig. 5. The netlist obtained from the genetic evolution using the operations described in Sec. II provides a quick way of custom generating library cells. The fitness value is evaluated by summing up the exclusive-OR vector derived using the input (required) truth table from the user and the evolved truth table response for the stimulus vector obtained from *IRSIM*. *When the fitness value reaches* 0*, the two responses match and the netlist obtained from the genetic method is valid.* Fig. 4 and Fig. 5 show evolved netlist translations to

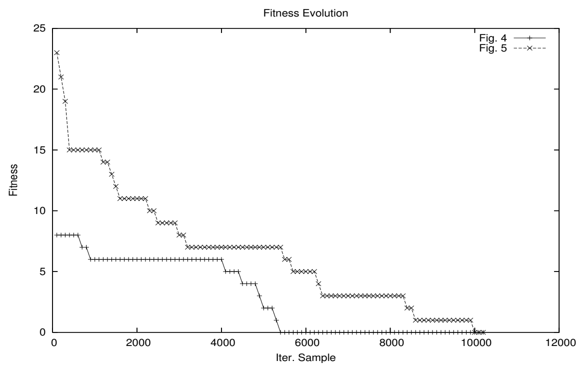

Fig. 3. The fitness evolution samples over intermediate generations, plotted for cases in Fig.4 and 5 with four and five inputs respectively.

dynamic CMOS circuit schematics. Our genetic methodology guarantees a convergence to a point which gives the simplified netlist incorporating all the boolean simplification rules. This means that the final netlist obtained after evolution has the minimum transistor count. The netlist could be used directly as a library cell instead of simplifying the logic at the gate level and then translating it to the transistor netlist which may still miss out a few optimizations in the transistor topologies. This transistor level simplification can be carried ahead into layout level optimization. Better transistor sizing can also be obtained by using the stochastic methods described in [2]. This would be the next step in optimization to obtain the best layout and size for *characterizing a cell.* Table. II shows the convergence

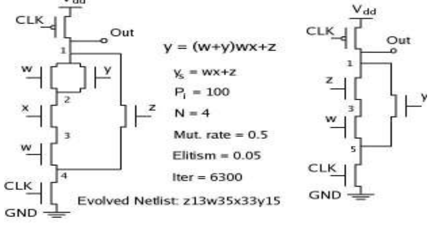

Fig. 4. Evolved SPICE Netlist for the testcase with N=4, P=100 and G=23

rates and parameters used for the experiments conducted on a few test cases for the proposed genetic methodology. It is

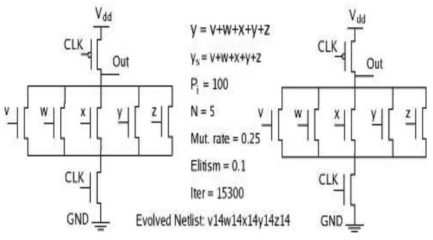

Fig. 5. Evolved SPICE Netlist for the testcase with N=5, P=100 and G=153

evident that the convergence is practical, albeit stochastic. Our genetic methodology could easily be incorporated into any digital circuit design flow to enable the creation of *dynamic library cells* on the fly. Fig. 1 shows one such methodology where the genetic netlist creation forms the fundamental step in the process of custom and semi-custom circuit design.

IV. Future Work

From the design methodology and the results described in Sec. II and III, it is evident that our methods are scaleable to incorporate four and five input truth tables with extremely practical speeds of convergence. It is important to note that the search space complexity in the problem is of the order 2^{2^n}, which is enormous even for a four or five input truth table. For higher order functions, convergence rates become a serious issue. However, the *shannon's decomposition*, laid out in [13] and the equation below, could be effectively used to exploit some parallelism.

$$\begin{aligned} f(x_n, x_{n-1}, \ldots, x_0) = f(x_n, x_{n-1}, \ldots, 1)x_0 \\ + f(x_n, x_{n-1}, \ldots, 0)\overline{x_0} \end{aligned}$$

The decomposed functional topology with fewer inputs can easily be evolved in parallel using our genetic methods and the whole truthtable can be realized by combining the decomposed parts using the dynamic CMOS logic. Scalable functionality can hence be incorporated into the proposed method at various levels. Experiments with the scaleable models could be the future direction for the current design methodology. This would make it feasible to work with arbitrary truth table inputs. This automatic transistor level optimization of the topology starting from truth tables, completely avoiding the Boolean simplification approach is the first known methodology to the best of our knowledge. The transistor sizing [2] and placement optimization [1], [4] and [5] can easily be incorporated in the current model for the complete genetic custom design flow.

V. Conclusion

The results and designs set out in this paper clearly describe the implementation techniques for the genetic evolution of optimized transistor netlists starting from truth table descriptions. These can be effectively used to generate custom library cells on the fly. Although demonstrated for the dynamic CMOS and the domino logic case, the design methodology can generically be extended to include static CMOS and other logic styles using the principle of duality. Appropriate heuristics in the evolution of the transistor netlist starting from truth table descriptions are shown to perform practically. These include Boolean simplifications and other optimizations using genetic operators of crossover and mutation. The methodology bypasses the normal way of Sum of Product (SoP) or Product of Sum (PoS) formulation of a function description followed by gate level optimization and transistor netlist generation. The netlist obtained directly from the functional specification can be used for layout preceded by optimizations for sizing and placement.

TABLE II
Exemplary Circuit results for test truth table inputs with the mutation and elitism rate over multiple generations

Boolean Expr	Elite Rate	Mut Rate	N	iter	final net
$Y = (x+y)(yz+x)+xy$ $Y_s = xz$	0.1	0.1	3	1300	$x12, y22, z234$
$Y = x\overline{yz} + \overline{x}y\overline{z} + xy\overline{z}$ $+\overline{xy}z + x\overline{y}z + \overline{x}yz$ $Y_s = x+y+z$	0.1	0.1	3	2300	$x14, y14, z14$
$Y = wx\overline{yz} + wx\overline{y}z + wxyz$ $+wxy\overline{z} + \overline{wx}yz + \overline{w}xyz + w\overline{x}yz$ $Y_s = wx+yz$	0.1	0.1	4	6600	$w14, x14, y12, z25$
$Y = (w+y)wx+z$ $Y_s = wx+z$	0.05	0.5	4	6300	$d13, a35, b33, c15$
$Y = v+w+x+y+z$ $Y_s = v+w+x+y+z$	0.1	0.25	5	15300	$v14, w14, x14, y14, z14$

References

[1] Lefebvre, M, Marple, D and Sechen, D, *The future of custom cell generation in physical synthesis*, Proc. 34th Annual Conference on Design Automation, pp.446-451, 1997

[2] Rogenmoser, R, Kaeslin, H and Blickle, T, *Stochastic Methods for Transistor Size Optimization of CMOS VLSI Circuits*, Proc. 4th Intl. Conference on Parallel Problem Solving from Nature, pp.849-858, 1996

[3] Heusler, L.S, *Transistor sizing for timing optimization of combinational digital CMOS circuits*, PhD Thesis, ETH Zurich, 1990

[4] Ho, M.C, Leung, S, Kurokawa, H and Choy, O.C, *Digital logic synthesis using genetic algorithms*, 2nd Intl. Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications, GALESIA, pp 296-301, September 1997

[5] Bahuman, A, Bishop, B and Rasheed, K, *Automated Synthesis of Standard Cells using Genetic Algorithms*, Proc. IEEE Symposium on VLSI, pp.126-133, April 2002

[6] Hounsell, B.I and Arslan, T, *A Novel Genetic Algorithm for the Automated Design of Performance Driven Digital Circuits*, Proc. Congress on Evolutionary Computation, pp.601-608, vol.1, July 2000

[7] Geun Rae Cho and Chen, T, *Mixed PTL/Static Logic Synthesis Using Genetic Algorithms for Low-Power Applications*, Proc. Intl. Symposium on Quality Electronic Design, pp.458-463, August 2002

[8] K.Yano, et al, *A 3.8ns CMOS 16x16-b multiplier using complementary pass-transistor logic*, IEEE J.Solid-State Circuits, vol.25, pp.388-395, Apr.1990

[9] J.Pasternak et al, *Differential Pass Transistor Logic*, IEEE Circuits and Devices, pp.23-28, July 1993

[10] Makoto Suzuki, et al, *A 1.5ns 32b CMOS ALU in Douple Pass-Transistor Logic*, IEEE J.of Solid-State Circuits, vol.28, no.11, pp.1145-1151, November.1993

[11] Mazumder,P and Rudnick, E.M, *Genetic algorithms for VLSI design, layout and test automation*, Prentice Hall, pp.264–266, 1999

[12] Goldberg, D.E, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company Inc., 1989

[13] Woods, S and Casinovi, G, *Efficient solution of systems of Boolean equations*, IEEE/ACM International Conference on Computer-Aided Design, ICCAD-96, pp.542 - 546, November 1996