CCSOpt: A Continuous Gate-Level Resizing Tool

Dimos Ntioudis[†,*], Christos Kalonakis[*], Panagiotis Giannakou[†,*], Charalampos Antoniadis[†,*], Georgios Stamoulis[*], Panagiota Tsompanopoulou[†,*], Nestor Evmorfopoulos[*], John Moondanos[†,*], Georgios Dimiriu[†,*]
[*]Department of Electrical and Computer Engineering
University of Thessaly
[†]Centre for Research and Technology / Institute for Research and Technology Thessaly
{ntioudis, hrkalona, panagian, haadonia, georges, yota, nestevmo, jmoondan, dimitriu}@inf.uth.gr

***Abstract*— Optimal continuous transistor/device sizing has been a holy grail in the EDA community. However, efforts to this end have been hampered by the sheer size of the optimization problem (millions of variables and constraints), modeling issues especially in the timing domain. This research work proposes *Continuous Cell Size Optimizer* (CCSopt), a continuous gate-level sizing tool that finds the optimum cell shrinkage by setting as the objective the minimization of the circuit delay. Notice that because of only shrinking the cells the power consumption of the circuit is reduced as well. CCSopt comprises a hybrid heuristic approach and state-of-the-art algorithms for finding the optimal transistor sizes. In addition, CCSopt can exploit the computational power of parallel architectures in order to decrease execution time and enable analysis of very large-scale integrated circuits.**

Keywords*— *transistor sizing, power optimization, delay optimization, logical effort, incremental static timing analysis

I. INTRODUCTION

The proposed activity is building on the learnings from both academic and industrial attempts to tackle a difficult yet attractive design problem. The approach taken is to perform continuous sizing optimization but in a constrained mode, in order to arrive at solutions that are reliably implemented in silicon, and easily integrated into mainstream design flows. Transistor sizing tools have been around since the publication on TILOS [1] [2]. Initially the sizing effort was focused on transistor sizes only, for which a number of approaches have been developed (posynomial sizing [3], logical [4] AMPS [6]). Further strains of the aforementioned basic approaches have been proposed initially for timing and area optimization, and, later for multi-objective cost functions involving mainly power [7] [8]. Most of the sizing tools are path based, meaning that they treat the transistors of gates along a path as an optimization sub-problem, which can cause serious conflicts especially in similarly timed reconvergent fanout paths. A major undertaking has been to observe design constraints while performing transistor sizing. Minimum and maximum slope constraints have been the most difficult to implement as they are not very compatible with any of the sizing methods that have been proposed thus far. Minimum and maximum transistor sizes, maximum delay constraints, and fixed relative transistor sizing are more straightforward to implement. Recent academic papers and patents incorporate interconnect capacitance and, in some cases resistance, to account for the delay in interconnect lines [9].

The rest of this paper is organized as follows. In section II we present the software architecture. In section III we give background material on certain useful fragments of cell resizing. Section IV introduces an enchanted version of the *Static Timing Analysis* (STA) method. In section V we present the implementation steps of the CCSopt tool. Section VI presents experimental results on several benchmarks. Finally, section VII provides ideas for future work.

II. SOFTWARE ARCHITECTURE

CCSopt is a stand-alone tool, which was built on top of *Unified Logical Effort* (ULE) methodology which takes into account the interconnect resistance and capacitance, in order to achieve high convergence rate to the optimal cell sizes solution. The core of the tool (see Figure 1) consists of a fast STA engine which evaluates the design's timing information throughout the execution of the algorithm. The inputs of the tool consists of the design's external topology or cell connectivity information (.v file), the cell's internal information such as internal connectivity and delay (.lib file), the parasitic information derived after placing and routing (.spef file), along with a set of instructions for the algorithm (.cfg file). The outputs of the tool consists of the transformed design (.v file) along with the new cells scale factors (.scf file).

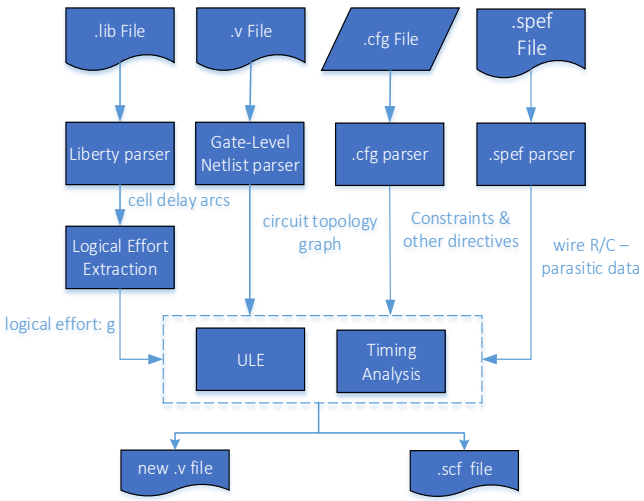

Figure 1 - Software Architecture

III. UNIFIED LOGICAL EFFORT

As VLSI circuits continue scaling, the contribution of wires to the delay is increased and thus it cannot be neglected. This characteristic occurs not only with respect to long wires connecting separate modules but also to the interconnect within logic modules where the delay introduced by the wires connecting closely coupled gates approaches or in some circumstances exceeds the gate delay. Logical effort optimization for gates without wires is based on equal stage efforts:

$$g_i \frac{C_{i+1}}{C_i} = g_{i+1} \frac{C_{i+2}}{C_{i+1}} \quad (1)$$

, where g_i and C_i are, the logical effort and the input capacitance of the cell i, respectively. The useful Logical Effort (LE) rule that the path delay is minimum when the effort of each stage is equal breaks down, because interconnect has fixed capacitances which do not correlate with the characteristics of the gates. This drawback of LE methodology is described by its authors as *"one of the most dissatisfying limitations of logical effort"*.

A. Delay Model of Logic Gates with Wires in Paths with Branches

The logical effort model is modified to include the interconnect delay [3]. This change is achieved by extending the gate logical effort delay by the wire delay, establishing a Unified Logical Effort model. Thanks to the Elmore [12] delay model the delay of a circuit comprising logic gates and wires (see Figure 2) can be easily calculated.

The circuit in Figure 2 shows the general structure containing a side branch with RC interconnect and/or a fanout load with arbitrary capacitance where R_b and C_b are the resistance and capacitance of branch wires, respectively, and C_f is the fanout load capacitance. The ULE optimum expression can be generalized for any combination of side branch wires and fanout gates by determining the total effective capacitance of the fanout branches for each stage of the path:

$$C_{BF} = \sum_{1}^{n} C_{b_n} + \sum_{1}^{m} C_{f_m} \quad (2)$$

, where n and m are the number of branch wires and fanout gates in a path, respectively. Taking into consideration the last equation, the general ULE optimum expression for the input capacitance is determined [11]:

$$C_i = \sqrt{C_{i-1} \times C_{i+1}} \times \sqrt{1 + \frac{C_{w_i}}{C_{i+1}} + \frac{C_{BF_i}}{C_{i+1}}} \times \sqrt{\frac{g_i}{g_{i-1} + \frac{R_{w_{i-1}} \times C_{i-1}}{\tau}}} \quad (3)$$

B. Relaxation Method

In order to simplify the solution, a relaxation method is proposed in [11]. The technique is based on an iterative calculation along the path while applying the optimum conditions. Each capacitance along the path is iteratively replaced by the capacitance determined from applying the optimum expressions, shown in equation 3, to two neighboring logic gates. The technique consists of the following steps:

a) *(Iteration)* Replace each capacitance by the value determined from applying the optimum expressions on two neighboring logic gates.
b) *(Stop check)* If any of the new values differ by more than a given precision from the previous value, reiterate step a else stop.

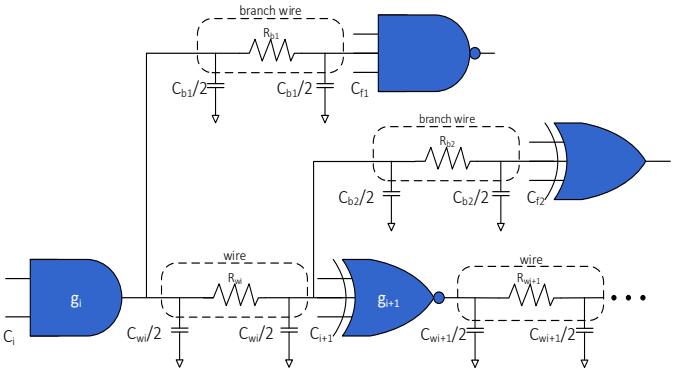

Figure 2 A logic path segment including RC interconnect and two branches

C. Logical Effort Parameter Extraction

There are multiple methods for extracting the logical effort parameter. A good standard cell library (.lib file) contains multiple sizes of each common gate. The sizes are typically labeled with their drive strength [15]. For example, a unit inverter may be called inv_x1. An inverter of eight times unit size is called inv_x8. It is often more intuitive to characterize gates by their drive strength, x, rather than their input capacitance. If we redefine a unit inverter to have one unit of input capacitance, the logical effort can be extracted by:

$$g = \frac{\frac{C_{in}}{C_o}}{x} \quad (4)$$

In particular, since every input pin has different capacitance, according to its state (falling, rising), there will exist two logical effort: g values for every input pin (g_{fall}, g_{rise}).

IV. ENHANCED STATIC TIMING ANALYSIS

The resizing algorithm performs the ULE method for delay evaluation and minimization in paths composed of CMOS logic gates. Each path, that will be evaluated, can be chosen arbitrarily from a pool of paths, but this technique would not yield the best results. A choosing criterion must be used in order

to select the most suitable path every time. The proposed algorithm uses the delay as the classification criterion when examining paths.
Timing analysis must be performed in order to sort the available paths according to their delay in descending order (late timing analysis). The delay models that have been used for cells and interconnect are NLDM and Elmore, respectively.

A. *Multi-threaded STA*

The implemented algorithm for STA has two embarrassingly parallel regions, that do not share any data, and there is not any need for a locking scheme to be developed. The first region corresponds to the evaluation of wire delay, because every net is completely independent from each other. The computation that is required to be performed in this region, requires path tracing to be involved in the interconnect graph, in case of paths with loops, where Elmore delay model cannot be applied, the maximum spanning tree algorithm is deployed to alleviate the problem. The second region corresponds the parallel calculation and propagation of the timing information, concerning the cells that are in the same level, because these cells do not share any common points in the graph of the circuit. This region of STA is the main hotspot of the resizing algorithm, since the timings of the design need to be revaluated over and over again. Note that in very large designs, the STA algorithm has to be performed thousands of times.

B. *Incremental STA*

STA is required to be performed every time a set of changes has happened in the design, which is the resizing of some cells within a path. An initial STA propagates the timing information in the design, starting from its primary inputs. Also a STA from the PIs has to be performed whenever a cell changes to update the timing information in the design Although this approach is sensible in terms of algorithm's correctness, it does not take into account that some portion of the design will stay unchanged, in terms of timing information, even if a change has taken place. A better approach would be to first find points that will be affected by the change, and then deploy the STA from those points. An enhancement to that approach can be achieved by knowing beforehand all the nodes, in which the arrival times will be required, for the critical path extraction, further reducing the arrival time propagation space.

V. GATE LEVEL RESIZING ALGORITHM

The core of the resizing engine is the ULE method, which discovers the optimal cell input capacitances for a given path, and therefore the optimal cell sizes. The method, as mentioned before, takes into account branches and wire load (Resistances/Capacitances), along with the slope at every pin in the path (Rise/Fall). The path is evaluated, traversing it in backwards order, for a number of iterations until the values of all the input capacitances have not changed, in comparison to a predefined error threshold, from those in the previous iteration.

The ULE method, evaluates a given path, which was not re-examined before by the resizing algorithm. In order to filter the paths into examined (Partially) or not, a sub-path extraction algorithm was implemented. This algorithm receives as input the critical path from the STA. The path is then tokenized into smaller, not examined segments of the path, if any, and each sub path is then processed by the ULE method. In the case of disallowing upsizing, the ULE method fails to process the given path, which will lead to even further preprocessing of the path.

Aiming to examine as many as possible cells for resizing we list systematically all possible terminal nodes existing in the circuit graph by keeping a set of them. The set is initialized with the Primary outputs (POs) of the circuit and it is updated with new terminal nodes, the pins of a cell we have already examined. The algorithm finishes when there are no terminal nodes in the set. The ULE method populates a list of changed

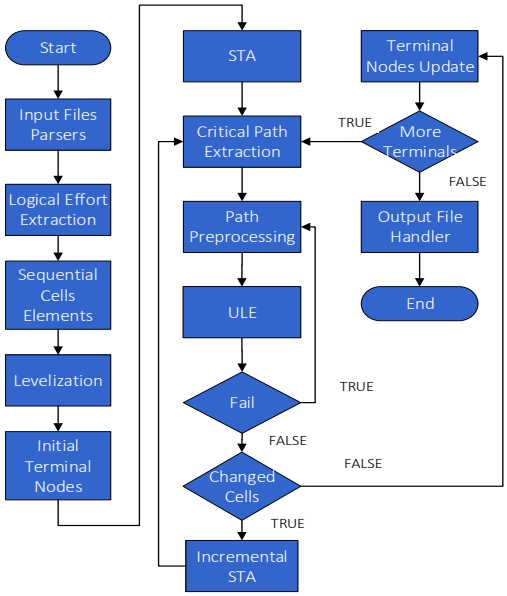

Figure 3 CCSopt algorithm overview

cells, along with the affected nets. The cells are defined as changed, only if they have been replaced by a newer downsized. The ULE, often does not result to the replacement of any cell in the path and as a result the resizing algorithm falls into a dead-end. However, the algorithm must continue the examination of the rest of the circuit, by finding the next in line critical path.

Since no cell was changed during the last path examination, there is no need to apply a STA. The problem that arises, concerns the critical path extraction which results in the same path as before. In order to alleviate that problem, the terminal node that produced the dead-end has to be removed from the set.

VI. EVALUATION

In this section, a number of benchmarks is presented, indicating the number of the resized cells, using the proposed resizing

algorithm. The suite of benchmarks has been take from TAU 2015 timing contest and ISPD contest. To the best of our knowledge it is the first time a continuous cell sizing tool is presented and as a result there are no results to be compared with the provided ones.

Table 1 Benchmarks

Benchmark	# Cells	# PI	# PO	#Resized Cells	Execution Time (s)
ac97_ctrl	14341	84	48	3603	15.545
aes_core	22938	260	129	2965	29.054
des_perf	105371	235	64	16665	161.352
mem_ctrl	10531	115	152	2186	13.710
pci_bridge32	19057	162	207	4234	25.073
systemcaes	6484	260	129	1236	9.055
systemcdes	3441	132	65	469	3.776
tv80	5285	14	32	773	6.492
usb_funct	15743	128	121	3593	18.424
wb_dma	4195	217	215	891	5.052
vga_lcd	139529	80	109	34146	290.549
cordic_ispd	45359	34	64	20118	52.970
des_perf_ispd	138878	234	140	83548	198.536
edit_dist_ispd	147650	2562	12	88152	251.985
fft_ispd	38158	1026	1984	13824	45.411
matrix_mult_ispd	164040	3202	1600	68545	261.750
pci_bridge32_ispd	40790	160	201	21701	44.429
usb_phy_ispd	923	15	19	551	1.044
netcard_iccad	1496719	1836	10	369598	17942.690

VII. Conclusions & Future Work

We have presented a continuous gate-level resizing tool that takes into consideration the interconnect capacitance and resistance. It also takes into account reconvergent fan-outs and arrives at a stable solution in all cases without the possibility of divergence. Moreover, the evaluation results guarantee high accuracy within acceptable timeframe.

A number of possible extensions and changes should be revisited, that will allow CCSopt to have better quality and performance. Some of those extensions refer to a) reducing the number of cells that do not get resized, such as the first cells of the off-path branches, b) using a better criterion, in addition to only allowing down-sizing, to further reduce the power consumption, c) changing the delay calculation method, to use the CCS model as described in the lib file, to better approximate the delays of a cell, since the NLDM is not so accurate in the sub-nanometer regime, and d) altering the incremental STA, in order to only propagate the slews and arrival times through cells that lead to active terminal nodes, which will improve the overall performance.

Acknowledgements

This work was supported by EU and the Greek State through ESPA 2013-2017, Action SYNERGASIA 2011, Project Code: 11SYN 5 719

References

[1] J. Fishburn and A. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in Proceedings of the IEEE International Conference on Computer-Aided Design, 1985.

[2] A. E. Dunlop, "Transistor Sizing for integrates Circuit". U.S. Patent No.4827428.

[3] S. S. Sapatnekar, B. V. Rao, P. M. Vaidya and S. M. Kang, "An exact Solution to the Transistor Sizing Problem for CMOS Circuits using Convex Optimization," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1993.

[4] I. Sutherland, D. Harris and B. Sproull, Logical Effort - Designing Fast CMOS Circuits, Morgan Kaufmann Publishers, 1999.

[5] L. Sutherland, "Determining transistor widths using the theory of logical effort". U.S. Patent No.6629301.

[6] H. H. F. Jyu, "Minimization of circuit delay and power through transistor sizing". U.S. Patent No.6209122.

[7] L. G. Jones, "Method and apparatus for designing an integrated circuit". U.S. Patent No.5666288.

[8] R. F. Leimbach, "Method of optimizing signal timing delays and power consumption in LSI circuits". U.S. Patent No.4698760.

[9] A. Morgenshtein, "Logic circuit delay optimization". U.S. Patent No.12292931.

[10] F. R. Sproull and E. S. Ivan, "Logical Effort:designing for speed on the back of an envelope," in IEEE Advanced Research in VLSI, 1991.

[11] A. Morgenshtein, E. Friedman, R. Ginosar and A. Kolodny, "Unified logical effort - a method for delay evaluation and minimization in logic paths with RC interconnect.," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[12] C. W. Elmore, The trasient response of damped linear networks with particular regard to wide band amplifiers, vol. 19, J. Appl. Phys., 1948, pp. 55-63.

[13] TAU Workshop, "tauworkshop," 9 2 2015. [Online]. Available: https://sites.google.com/site/taucontest2015/resources

[14] ISPD Contest,"ispd" 9 2 2015. [Online]. Available: http://www.ispd.cc/contests/15/web/benchmark.html

[15] Neil H. E. Weste, David Money Harris, CMOS VLSI Design: A Circuits and Systems Perspective 4th, 2010, USA.