Timing closure:

the solution and its problems

Ralph H.J.M. Otten
Eindhoven University of Technology
Eindhoven, The Netherlands

Abstract— In this paper we summerize the derivation of the size equations, the key to *timing closure*. Next we present a number of problems when applying these equations in practice. The main ones are network generation, discrete libraries, size constraints, and resistive interconnect.

I. Introduction

Timing closure is the dimensioning of a logic network such that timing constraints are satisfied. The now popular formulation is that logic synthesis not only produces a netlist with a set of *functional* nodes (logic gates) and their interconnections, but also the delay of each gate by prescribing its restoring effort (or gain), defined as the quotient of output and input capacitance. A fundamental result from [2] states that if delay is to be kept constant the gate size has to scale linearly with its load capacitance. Given the netlist with the restoring effort of each gate, and given the capacitances imposed on the nets, sizes have to be assigned to the gates in such a way that the capacitance ratios are realized simultaneously. In section II.B, theorem 3 gives necessary and sufficient conditions when this is possible.

But that is merely a mathematical statement, assuming that all gates can be arbitrarily and continuously sized. Neither unfeasibly small sizes, nor unrealistic large sizes are prevented, and the not so uncommon restriction to a limited set of library sizes is not considered. Connections are assumed to be without resistance, only lumped capacitances are assumed. The fact that interconnect strategies constrain capacitances and therefore sizing, adds question marks to the beautiful results of section A.

II. Timing closure

A. Size equations

Starting from the model of figure 1 a delay formula arises which is the sum of two terms, the *effort delay* and the *parasitic delay*[8, 9]:

$$\tau = bR_{tr}C_L + bR_{tr}C_p = br_oc_o\frac{C_L}{C_{in}} + br_oc_p = \frac{g}{f} + p. \quad (1)$$

The right-hand expression is called *sutherland delay*. The parasitic delay $p = br_oc_p$ is independent of size. The effort delay g/f is a product of *computing effort* $g = br_oc_o$, and *restoring effort*

$$\frac{1}{f} = \frac{C_L}{C_{in}}$$

The computing effort is also size independent, but in general depends on the function, topology and relative transistor dimensioning of the gate type. The important observation is that τ can be kept constant by fixing $f = C_{in}/C_L$. This leads to a new paradigm in synthesis[2, 7]: any delay imposed by synthesis can be realized, provided that the sizes of the gates can be continuously adjusted, and the imposed delay exceeds the parasitic delay. Note however that the derivation replaced the gate by a single linear "effective" resistance and a linear input and drain capacitance.

For more general charge or discharge networks of mos-transistors a single sizing factor can be derived[2]:

Theorem 1 *If*

1. *each transistor can be modelled by an effective resistance inversely proportional to the device-width,*
2. *each node i in the (dis)charge network can be modeled by a linear capacitance C_i composed of a constant part and device dependent parts, and*
3. *the gate delay can be approximated by a summation over all nodes of R_iC_i where R_i represents the total resistance between node i and the output node[10],*

then the delay of the gate remains constant if all device widths scale linearly with with the load capacitance C_L

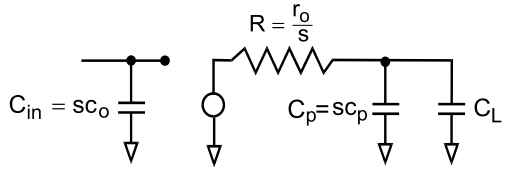

Fig. 1. Gate model for obtaining a size independent delay expression

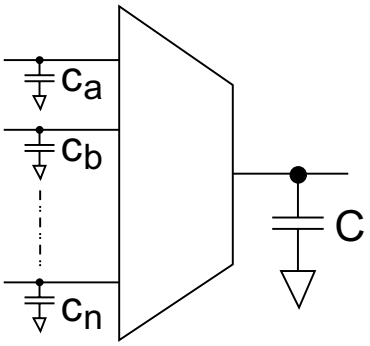

Fig. 2. Fixed relation of each input c with output c

The third condition reflects the elmore intuition of approximating delay by adding the time constants of single rc-sections, each consisting of a node capacitance and the total resistance through which it is charged or discharged. Of course, the delay is an approximation for an already idealized network (linear "effective" components, lumped capacitances, etc), but experiments[1] support the stated fact extremely well[2]:

THEOREM 2 *The delay of a logic gate can be kept constant by scaling the devices linearly with the (external) load.*

The derivation of theorem 1 also implies that under constant delay all input capacitances of a gate scale linearly with the load. This leads to an analog "*restoring effort*" as in the sutherland delay for a buffer, that is, a single factor f. So, with reference to figure 2, for every input x we have $c_x \propto fC$ where the proportionality constant can be different for different inputs of the gate (but these constants do not change with the restoring effort, and therefore not with f!). It is reasonable to assume that the size of a gate is proportional to the sum of its transistors, and therefore proportional to the sum of the input capacitances:

$$\text{gatesize} \propto \sum_{x \in \{a,b \cdots n\}} c_x \ .$$

Since each c_x is proportional with fC, the size of a gate is proportional to fC as well[2], and this proportionality constant is called the *area sensitivity* of that gate type: afC. The area sensitivity of the gate equals the gate area if the restoring effort is 1 and at the output there is a unit capacitive load.

COROLLARY 1 *The size of a gate is proportional to the capacitance at the input which under the constant delay paradigm equals the capacitance at the output multiplied by f as imposed by synthesis.*

In the context of a network the implications of corollary 1 can be worked out by writing the expression for the total capacitance at a single node (see figure 3) and then collect them in vector form as follows. The capacitance at node i is the imposed capacitance q_i at that node (this can be the external capacitance to be driven by the network or the wiring capacitance, but lumped and without wire resistance) and the (scaled) input capacitances in its fanout:

$$c_i = q_i + \sum_{j \in fanout(i)} n_{ij} f_j c_j .$$

In figure 3 the summation for node i contains two terms: $c_i = q_i + n_{ij} f_j c_j + n_{ik} f_k c_k$. If we make $n_{ij} = 0$ whenever gate j is not in the fanout of gate i, and equal to the proportionality constant that comes with the gate type of gate j (accounting for function, topology, and relative sizing) we can write in general

$$c_i = q_i + \sum_{j=1}^{n} n_{ij} f_j c_j \ .$$

Collecting the imposed capacitances in a vector $\mathbf{q}$, the capacitances at the output in a vector $\mathbf{c}$ and the reciprocals of restoring effort in a diagonal matrix $\mathbf{f^D}$, yields the following relation:

$$\mathbf{c} = \mathbf{q} + \mathbf{N}\ \mathbf{f^D c}$$

or

$$\left(\mathbf{I} - \mathbf{N f^D}\right) \mathbf{c} = \mathbf{q} \qquad (2)$$

The matrix $\mathbf{N}$ has the zero/non-zero pattern of the incidence matrix of the (directed) network, and contains the

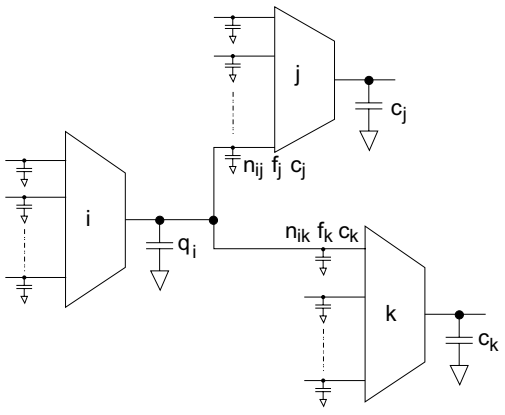

Fig. 3. Effort relations in a network

[1] The same experiments also supported the validity of the sutherland model by showing the invariance of the socalled *self loading*: $\frac{c_p}{c_p + c_L}$.

[2] That there is a single factor f for each gate means that by relative dimensioning of the pull-up and pull-down networks, the transfer behavior can be manipulated and constant delay synthesis will not change that. However, if each input of a gate requires a separate relation to the output capacitance, it would not complicate the formulation

relative sizing of the transistors in the values of the nonzeros. Both, the pattern of $\mathbf{N}$ and $\mathbf{f}$ are imposed by logic synthesis, and the entries come from the library. They should be such that equation 2 has a positive real solution.

COROLLARY 2 *The node capacitances are related to the imposed capacitances by a linear transformation.*

Once all node-capacitances are known, that is the vector $\mathbf{c}$, the area of the network is can be written as

$$(\mathbf{a}\cdot\mathbf{f})^{\mathbf{T}}\mathbf{c} = (\mathbf{a}\cdot\mathbf{f})^{\mathbf{T}}\left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)^{-1}\mathbf{q}$$

where the dot indicates componentwise multiplication. Based on this relation one can determine whether inserting buffers can save area.[3] Note however that although logic synthesis is to produce a network of gates with restoring effort assigned to each gate, the area of the gates and the total network is not known, because the capacitances at the nodes are not known.

B. Solving the size equation

For establishing existence conditions for continuous solutions, we look at three different cases: combinatorial networks, strongly connected networks and general networks.

In case of *combinatorial networks*, that is no memory elements and no cycles,etc., the network can be represented by an acyclic directed graph. Consequently, there exists a topological ordering of the nodes. Using that to order the equations in the set 2 yields a lower triangular matrix that can be solved by backsubstitution.

If the network is strongly connected, the matrix $\mathbf{N}$ is irreducible[4], and therefore $\mathbf{N}\,\mathbf{f}^{\mathbf{D}}$ as well. The question whether the equation system 2 has positive solutions ($\mathbf{c}>\mathbf{0}$) for any $\mathbf{q}>\mathbf{0}$ has been studied extensively. It is known as an *open leontief system*. The main result from this setting is

THEOREM 3 *The equations*

$$\left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)\mathbf{c} = \mathbf{q}$$

with $\mathbf{N}$ *a nonnegative irreducible matrix and* $\mathbf{f}$ *a strictly positive vector, has a solution* $\mathbf{c},\ \mathbf{c}\geq\mathbf{0},\ \mathbf{c}\neq\mathbf{0}$ *for any* $\mathbf{q}\geq\mathbf{0},\ \mathbf{q}\neq\mathbf{0}$ *fif* λ*, the perron-frobenius root (that is the dominant eigenvalue) of* $\mathbf{N}\,\mathbf{f}^{\mathbf{D}}$*, is smaller than* 1.
In that case, there is only one solution $\mathbf{c}$*, which is strictly positive and given by*

$$\mathbf{c} = \left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)^{-1}\mathbf{q}.$$

[3]Buffers can only be inserted if they do not cause violations in the timing constraints; buffers are allowed to introduce additional delay locally only if some slack time is available there.

[4]The usual definitions of irreducibility using powers of matrices or matrix decomposition are equivalent, but in the present context indirect and not so useful.

A number of useful corollaries easily follow from that result. We mention

COROLLARY 3 *If it exists,* $\left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)^{-1}>0$ *fif* $\lambda<1$.

COROLLARY 4 *If none of the row sums of* $\mathbf{N}\mathbf{f}^{\mathbf{D}}$ *exceeds unity, and at least one is less than unity, then* $\lambda<1$.

A condition equivalent to $\lambda<1$ and avoiding the solving of eigenproblems is from the following theorem[5]

THEOREM 4 $\lambda<1$ *fif all leading principal minors of* $\left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)^{-1}$ *are positive, where the i-th leading principal minor is calculated from the first i rows and columns of* $\left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)^{-1}$.

Under the stronger (than $\lambda<1$) condition of corollary 4 one can derive that when increasing the imposed capacitance at one node, the gate driving that node gets the greatest absolute increase in input capacitance. This does not imply the greatest increase in area, because that also depends on the area sensitivities. About relative changes something can be said under the minimal conditions of theorem 3:

THEOREM 5 *If the components of the vector* $\mathbf{q},\ \mathbf{q}\geq\mathbf{0}$ *and* $\mathbf{q}\neq\mathbf{0}$ *change by amounts of* $\mathbf{\Delta q}$ *such that* $\mathbf{q}+\mathbf{\Delta q}\geq\mathbf{0}$ *and* $\mathbf{q}+\mathbf{\Delta q}\neq\mathbf{0}$*, while* $\lambda<1$*, then for each* i

$$\min\left\{(0,\min_{\{j|\Delta q_j<0\}}\frac{\Delta c_j}{c_j}\right\}\leq\frac{\Delta c_i}{c_i}\leq\max\left\{0,\max_{\{j|\Delta q_j>0\}}\frac{\Delta c_j}{c_j}\right\}.$$

This implies that if only the imposed capacitance at a particular node changes while all other imposed capacitances remain the same, the gate driving that node changes by the greatest percentage (both, its input capacitance and its size, regardless of its area sensitivity).

Finally, if the network is not strongly connected, and consequently $\mathbf{N}\mathbf{f}^{\mathbf{D}}$ is reducible, we have to resort to weaker results of the perron-frobenius theory. The strict positivity in corollary 3 has to be replaced by nonnegativity, that is

$$\lambda>1\Longrightarrow\left(\mathbf{I}-\mathbf{N}\mathbf{f}^{\mathbf{D}}\right)^{-1}\geq 0,$$

but theorem 4 is still valid, also for networks that are not strongly connected. Non-negative solutions for the components of vector $\mathbf{c}$ are therefore still ensured, but some may be equal to 0.

C. Numerical solution of the size equations

An iteration equation for solving the equation set 2 presents itself in a natural way:

$$\mathbf{c}(k+1) = \mathbf{N}\,\mathbf{f}^{\mathbf{D}}\mathbf{c}(k)+\mathbf{q}.$$

[5]The condition of this theorem is known as the hawkins-simon condition

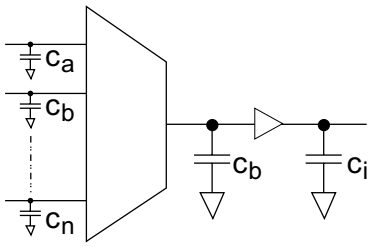

Fig. 4. Inserting a buffer

In fact, it is the well-known *jacobi* method and $\mathbf{Nf^D}$ is the socalled *jacobi matrix* of the system. However, it is just one out of many ways of *splitting* the coefficient matrix for iteratively solving the equations. Another familiar splitting leads to the *gauss-seidel* iteration:

$$\mathbf{c}(k+1) = (\mathbf{I}-\mathbf{L})^{-1}\mathbf{U}\ \mathbf{c}(k)+(\mathbf{I}-\mathbf{L})^{-1}\mathbf{q}.$$

where $\mathbf{L}$ is the strictly lower triangular part of $\mathbf{Nf^D}$ and $\mathbf{U}$ the strictly lower triangular part[6]. The inverse obviously exists (and is actually equal to $\sum_{i=1}^{n-1}\mathbf{L}^i$ with n the number of gates in the network). Let us denote the dominant (non-negative) eigenvalue of the *gauss-seidel* matrix $(\mathbf{I}-\mathbf{L})^{-1}\mathbf{U}$ with λ_{GS} (the gauss-seidel matrix is reducible). The well-known stein-rosenberg theorem then implies

THEOREM 6 *If $\lambda < 1$ then $0 < \lambda_{GS} < \lambda$*

COROLLARY 5 *Both the jacobi and the gauss-seidel method do converge when $\lambda < 1$, and the latter converges asymptotically more quickly.*

Other iteration schemes are possible, and may be faster. *Successive overrelaxation* is such a candidate. But to achieve this computational advantage in convergence, more knowledge about the eigensolutions, and the dominant eigenvalues is needed. To do the analysis, or calculating useful bounds might not pay of, and therefore the preferred approach is *gauss-seidel iteration.*

D. Area recovery

Inserting buffers might decrease the total area of a module. However, time critical paths should not get buffers inserted, because they cause additional delay. Only when there is a certain amount of slack, buffers can be inserted if they provide a decrease in area. We will show that all potential insertion points can be determined at synthesis time, that is before the node capacitances, and thus the gate sizes are known.

[6] $\mathbf{N}$ has only zeros on the diagonal because the input and output of driving gates are never directly connected (except for generating the inversion voltage as for fast sensing, but that is outside the present application).

The delay of a buffer is given by equation 1, that is $\tau_s = g_b/f_s + p_b$ with g_b and p_b as library constants. Area decreases by the insertion only when

$$a_i f_i C_i > a_i f_i c_b + a_b f_s C_i \quad \text{or} \quad a_i f_i > a_i f_i \frac{c_b}{C_i} + a_b f_s.$$

Note however that c_b/C_i is precisely equal to f_s, so that

$$a_i f_i > (a_i f_i + a_b) f_s$$

The condition for area recovery is therefore

$$\frac{1}{f_s} > 1 + \frac{a_b}{a_i f_i}$$

Substituting this in the delay equation of the buffer shows that the added delay by inserting a buffer is at least

$$p_b + g_b\left(1 + \frac{a_b}{a_i f_i}\right)$$

and this has to be compared with the available slack at that point. Clearly, all variables are either chosen by synthesis or library constants, and consequently, the comparison can be performed before sizing.

THEOREM 7 *A network with restoring efforts assigned in order to meet the timing requirements, can be reduced in size by buffer insertion fif there is a gate in the network with area sensitivity a_i and restoring effort f_i^{-1} such that*

$$p_b + g_b\left(1 + \frac{a_b}{a_i f_i}\right) \tag{3}$$

where g_b, p_b, and a_b are the computing effort, the parasitic delay and the area sensitivity of the buffer to be inserted.

This means that the locations for potential area recovery can be determined at synthesis time. For it is synthesis that creates the network (that is the matrix $\mathbf{N}$), selects the gates and assigns restoring effort to them. However, inserting a buffer at a certain node changes the slacks on all paths containing that node. Each insertion can invalidate many other potential area recovery points. And since sizing still has to take place, it is not known at synthesis time how much is gained by inserting a buffer. A generic procedure should therefore have synthesis insert buffers at all the candidate points, and minimize

$$(\mathbf{a}\cdot\mathbf{f})^{\mathbf{T}}\left(\mathbf{I}-\mathbf{Nf^D}\right)^{-1}\mathbf{q}$$

without violating the timing requirements by assigning values to the $\mathbf{f}$-components that belong to the inserted buffers, where the value 0 indicates no buffer in the object function, and $-g_b/p_b$ should be used in the delay equations.

III. Network generation

A. Heuristics

In addition to providing a network topology (the matrix **N**, and the identity of the gates), synthesis has to come up with the vector **f**. The complete problem is to produce the cheapest (smallest) network realizing the logic and timing specification. This is very hard, and effective heuristics have to be developed.

A simpler problem is: given a network topology what restoring effort should each gate have in order to satisfy the timing requirements without demanding too much space and/or unrealistic size variations over gates. Of course, the parasitic delay should not already exceed timing requirement on any path. Assuming the parasitic delay is small enough, the remaining time budget has to be distributed over the gates, which then is to be translated in a value for **f**.

Sutherland's hypothesis of uniform restoring effort [8] might be helpful here. It states that given a network with an equal number of gates on every path from primary input to primary output, a capacitive load at each primary output, and a driving capability at each primary input, the network is fastest when every stage on all input-output paths has the same effort. This is obviously true for a cascade of inverters, and it can be easily extended to networks with equal fanout in a stage. But also counter-examples can be easily constructed. Nevertheless, the principle may be useful as a heuristic. Note however that it is driven by the speed optimization, not by timing requirements. So, if valid, it shows the potential for perfomance guarantees, and subsequent relaxing and area recovery should produce an acceptable network with restoring effort assigned.

Other heuristics, among which equal delay distribution and zero slack algorithms, come to mind, but only extensive experience can provide sufficient evidence about their effectiveness.

B. Technology mapping

A side benefit is that technology mapping becomes efficient under this constant delay paradigm. Technology mapping is known to be an efficient cost (area) optimizer of convergent networks[7]. It has been shown recently [4] that technology mapping for load independent speed optimization can be solved efficiently for all acylic (and therefore combinational) networks. Thus the standard first step of technology mapping in logic synthesis of partitioning into trees need not be done and hence the optimum solution can be found.

Besides, elegant ways of capturing several networks into a representation were introduced [6]. To take advantage of these discoveries ways must be found to keep track of other performance characteristics than speed and to prevent excessive usage of computer time and memory.

[7] Often called leaf-dags. These are essentially trees of which the primary inputs may feed several gates.

IV. Floorplanning

A. Iterations

In sections II.B and II.C the imposed capacitances are assumed to be given constants. When solving the size equations by iteration these capacitances can be updated if adequate information becomes available. This is the case when a floorplan[8] is available at each stage. This floorplan does not have to be updated: only floorplan optimization has to be applied. This can be done efficiently if the floorplan satisfies certain structural restraints. For if these restraints guarantee that the floorplan possesses slicing property, many optimizations can be performed in polynomial time.

The scenario can be as follows:

1. generate a point configuration on the basis of the net list and minimum gate sizes
2. find the best slicing structure compatible with that point configuration ("best" can be interpreted as minimal area)[9]
3. iterate until convergence:
   (a) optimize the floorplan with area as objective
   (b) derive the wire capacitances on the basis preliminary positions (this yields the components of **q**)
   (c) solve the size equations
   (d) generate new shape constraints with the resulting sizes

Note thet the floorplan is not changed in this scenario, only the geometrical estimates. Of course, from time to time regeneration of point configuration and slicing structure is feasible!

B. Resistive interconnect

The approach in the previous section is only applicable if resistance on the wires can be neglected. For small modules (actually for all modules synthesis can handle today, and in the foreseeable future) this is an acceptable assumptions. But flat designs easily exhibit wires for which this is no longer sustainable. The equations cannot be modified such that resistance is taken into account

[8] To avoid confusion, a floorplan is **not** a rectangle dissection! A floorplan is a data structure that fixes relative positions between objects. Floorplan optimization determines the shapes under so-called shape constraints and optimizing an objective function while preserving the relative positions.

[9] This can be done in polynomial time, although speed-up techniques are necessary for more than thirty points.

and the above scenario can be maintained. For example, if half of the wiring capacitance is before and half is after a wire resistance R_W the delay formula becomes without load:

$$\begin{aligned}\tau &= b\left(R_{tr}(C_p + C_W) + R_W\frac{C_W}{2}\right) = \\ &= b\left(R_{tr}C_W + \frac{1}{2}R_W C_W + R_{tr}C_p\right) = \\ &= b\left((r_o c_g + \frac{1}{2}R_W C_{in})\frac{C_W}{C_{in}} + r_o c_p\right) = (g+r)h + p\end{aligned}$$

where r is **not** size-independent.

Simply negelecting resistance in large designs will lead to significant deviation fro projected performance!

V. Wire planning

A. Time budgetting

Wire planning refers to the approach of basing early decisions on the layout of global wires. Global wires in this concept are wires whose delay can be improved by segmentation and buffering. Local wires are assumed to be inside modules that in this stage are considered to be points. A wire plan is therefore a point placement of modules in a rectilinear space, that is, possibly under the presence of larger preplaced blocks. Several analyses on these wire plans are possible (e.g. detour free wirning, valid retiming, sliceability), but the final result should be a kind of floorplan with delays on the global wires. The remaining budget under the timing requirements can be consumed by the modules. The time budgetting problem is then to create the smallest area chip that satisfies the timing requirements. The trade-offs between area and delay of the modules can be assumed to be convex. This problem can be efficiently solved.

B. Size constraints

Wire planning seems to be the solution to the resistive interconnect problem. Only global wires are affected by it and designed before the synthesis of the modules. Within the modules the assumption of resistance free interconnect is acceptable, and the constant delay method can be applied.

However, there is now the problem of fixed capacitances: global wires are optimally buffered, yielding wire delays invariant under module positioning (in a detour free layout), but do have fixed input and wiring capacitances. This constrains the sizing. The simple example of cascades of inverters, well known from textbooks since the pioneering work of Mead and Conway, serves well to illustrate Sutherland's uniform stage effort, but also the necessity of having either the delay time as a variable to choose with consequences for the inverter sizes, or the input and output capacitance of the chain, but then constraining the performance that can be achieved.

VI. Library discretization

All derivations assumed continuous sizability of gates. It is not unlikely that gates are only available in discrete sizes. The selection of available sizes[1], algorithms that use these limited libraries[5], and the discretization problem in synthesis[3] deserve careful analysis, but the problems seem to be certainly surmountable. That is, if a solution exists under the assumption of continuous sizability, these studies and experience up to now show that effective solutions exist for reasonably rich discrete libraries, or adequate sets of cell generators.

Cell generation is most likely the big challenge in a constant delay approach. The set of functions can be quite small, but extensive research is necessary to determine which sizes should be made available. Ultimately, a library of cell layout generators seems to be the way to go. In addition, yield is also an issue here.

Conclusions

Timing closure is an intriguing novelty, with undeniable power but also serious unsolved questions. The panel brings together representatives of companies that seriously pursue the conversion of these ideas into tools. It will be also interesting to see how they cope with the problems of timing closure, the ones mentioned here as well as the ones encountered during their development.

References

[1] F.Beeftink, P.Kudva, D.S.Kung, L.Stok, *Gate-size selection for standard cell libraries*, Proceedings of the International Conference on Computer Aided Design, pp 545-550, 1998.

[2] J. Grodstein, E. Lehman, H. Harkness, B. Grundmann, Y. Watanabe, *A delay model for logic synthesis of continuously-sized networks*, ICCAD, Nov. 1995

[3] P. Kudva. *Continuous optimizations in synthesis: the discretization problem*, Proceedings of the International Workshop on Logic Synthesis, pp408-418, 1998.

[4] Y. Kukimoto, R.K. Brayton, P. Sawkar, *Delay-optimal technology mapping by dag covering*, Proceedings 35th Design Automation Conference, 1998

[5] D.S. Kung, *A fast fanout optimization algorithm for near-continuous buffer libraries*, Proceedings of 35th Design Automation Conference, pp352-353, 1998.

[6] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness *Logic Decomposition during Technology Mapping*, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, pages 813-833, August 1997.

[7] R.H.J.M. Otten, L.P.P.P. van Ginneken, N.V. Shenoy, *Speed: new paradigms in design for performance*, ICCAD, Nov. 1996

[8] I. Sutherland, R. Sproull, *The theory of logical effort: designing for speed on the back of an envelope*, in *Advanced Research in VLSI*, UC Santa Cruz, 1991

[9] I. Sutherland, R. Sproull, D.Harris,*Logical effort: designing fast cmos circuits*, Morgan Kaufmann publishers,inc, 1999.

[10] J.L.Wyatt Jr, *Signal propagation delay in rc models for interconnect*, chapter 11 (pp254-291) in *Circuit analysis, simulation and design, 2*, Elsevier Science Publishers B.V., 1987