EXHIBIT B, PART ONE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
REQUEST FOR FILING APPLICATION Under Rule 53(a),(b)&(f)
(No Filing Fee or Declaration)
RULE 53(f) NO DECLARATION

PATENT
APPLICATION

Asst. Commissioner for Patents
Washington, DC 20231

Atty. Dkt.

54355/238258

| | M# | Client Ref |

Date: April 2, 1998

Sir:

1. This is a Request for filing a new PATENT APPLICATION: ☐ Design ☒ Utility, entitled:

2. Title: TIMING CLOSURE METHODOLOGY

    without a filing fee or Oath/Declaration but for which is enclosed the following:

3. ☒ 1 Abstract page(s).

4. ☒ 75 Pages of specification (spec. and claims); 5. ☐ Specification in non-English language

6. ☒ 16 claim(s); and

7. ☒ 14 Drawings: sheet(s) per set: 14 1 set informal;

8. ☐ formal of size: A4 11"

9. ☐ DOMESTIC/INTERNATIONAL priority is claimed under 35 USC 119(e)/120/365(c) based on the following provisional, nonprovisional and/or PCT international application(s):

Application No.	Filing Date	Application No.	Filing Date
(1) 06/068,827	12/24/97	(3)	
(2)		(4)	

10. ☐ FOREIGN priority is claimed under 35 USC 119(a)-(d)/365(b) based on filing in

Application No.	Filing Date	Application No.	Filing Date
(1)		(3)	
(2)		(4)	

11. _ Certified copy(ies) ☐ enclosed / ☐ previously filed on in U.S. Application No. filed on

12. ☐ This is a reissue of Patent No.

13. ☐ Amend the specification by inserting before the first line --This is a ☐ Continuation-in-Part ☐ Divisional ☐ Continuation ☐ Substitute Application (MPEP 201.09) of:

    13(a) ☐ National Appln. No. / filed . --(M#)
    13(b) ☐ International Appln. No. filed which designated the U.S. --

14. ☐ See top first page re continuing application ("X" only if info is there)

    14(a) ☐ extension to date: ☐ concurrently filed ☐ not needed ☐ previously filed

15. ☐ Prior application is assigned to by Assignment recorded on at Reel/Frame

16. ☐ Also enclosed:

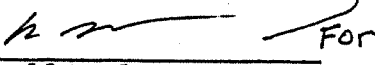17. This application is made by the following named inventor/s:

a. Name: Lukas P.P.P. van Ginneken   Country of Citizenship:
    Residence (City):   San Jose, California   The Netherlands
    Post Office Address: 4792 Country Lane
                          San Jose, California 95129

b. Name: Country of Citizenship: **United States of America**
    Residence (City):   , (State/Foreign Country):
    Post Office Address:

c. Name: Country of Citizenship: **United States of America**
    Residence (City):   , (State/Foreign Country):
    Post Office Address:

18. FOR ADDITIONAL INVENTORS, check box
    and attach sheet with same information regarding additional inventors .

Pillsbury Madison & Sutro LLP
Intellectual Property Group

1100 New York Avenue, N.W.
Ninth Floor, East Tower   By: **David A. Jakopin**, Reg. No. 32,995
Washington, D.C. 20005-3918
Tel: (650) 233-4790
Atty/Sec: For
Fax: (650) 233-4545   Arnold M. de Guzman
                                  Reg. No. 39,955

TIMING CLOSURE METHODOLOGY

1. Field of the Invention

This invention relates generally to the field of integrated circuit design and more specifically to a methodology for meeting timing constraints in the design of digital circuits.

2. Background of the Invention

In designing electronic circuits and systems, computer-automated design systems are used for defining and verifying various prototype circuit configurations. As part of the circuit definition, delay constraints are specified by the circuit designer. These delay constraints should be satisfied when the prototype circuit is fabricated.

In conventional approaches to circuit design, the following steps are typically performed:

(1) the load capacitance for each cell in the circuit is estimated using a fanout-based model;

(2)   the size of each cell is set to optimize

      timing of the circuit;

(3)   the cells are placed, and the net (wire)

      lengths of the circuit are estimated;

(4)   the wires are routed; and

(5)   final analysis is made to determine

      whether timing closure (i.e., satisfaction

      of timing constraints) is achieved.

In step (2), the sizes of cells within the circuit

are chosen and held constant once chosen. The

placement algorithm used thereafter will assign

different net lengths between cells, and these

lengths have conventionally been difficult to

predict prior to placement. While net lengths have

been estimated prior to placement by use of an

estimation function or table which gives the load

value of a net based on the number of fanout gates,

this estimation function is usually inaccurate.

This difficulty in accurately predicting net

lengths leads to unpredictable delay effects after

cell placement occurs. For example, some nets turn

out to be longer in length than expected. These

longer nets cause longer delays which prevent

satisfaction of timing constraints in the digital

circuit. Thus, under the conventional design

approach, timing closure is not certain until after

placement.

Failure to achieve timing closure after

placement leads to additional expenses and other

problems for the designer. To correct for failure

to achieve timing closure, the designer has the

option of fixing the design manually, which is

difficult and time consuming because the

automatically optimized digital network is not easy

to understand. As a second option, the designer

may change the Hardware Description Language (HDL)

specification and repeat the design process.

However, timing closure will again not be certain

until after placement. Thus, the design process

must again be repeated before the designer can

determine if the HDL specification changes were

successful in enabling timing closure.

A common method for dealing with inaccurate net

load estimates is by estimating the net load at a

considerably larger value than typically estimated.

Although this method increases the probability of

meeting timing constraints after placement, it

causes the sizes of the gates to be considerably

larger than necessary. Gates which are larger than

the necessary size are wasteful in both silicon

area and power consumption. This leads to chips

which are larger, more expensive to produce, and

use more electrical power than necessary.

Another problem with the conventional circuit

design approach concerns the timing analysis

required during optimization and during placement.

The timing analysis performed throughout the

conventional circuit design process is very time

consuming, and accounts for most of the run time of

a conventional circuit design system.

A further disadvantage of the conventional

design approach relates to the net length

modifications performed by the placement program.

Depending on the location chosen for each gate,

each net length may be modified. As each net

length is modified, the capacitive load of the net

will change. As a result, the delays of the gates

driving the net will change. Therefore, the

delays, which were carefully optimized during the

logic design, are very different in value after

cell placement, thereby contributing to poor

network optimization.

Additionally, most of the progress in the state

of the art for digital circuit design can be

characterized as increased integration which has

led to increasingly complex software systems which

are slow, and difficult to design and maintain.

A further disadvantage with conventional design

approaches is in the difficulty of iterating

between placement and sizing, since the logic

synthesis program is often operated by the logic

designer who also wrote the HDL specification, but

the placement program is often operated by the

silicon chip manufacturer, after the design is

complete.

One proposal has been made to keep delay

constant while expressing size as a linear function

of the gate load. See, J. Grodstein, E. Lehman, H.

Harkness, B. Grundmann, and Y. Watanabe, "A Delay
Model for Logic Synthesis of Continuously-Sized
Networks", Digest Int. Conf. On Computer Aided
Design, pp. 458-462, San Jose, CA Nov. 5-9, 1995.
Under this proposal, as the gate load changes, the
gate size automatically grows sufficiently to hold
the delay constant.

The constant delay model has been proposed in a
mapping algorithm. See, E. Lehman, Y. Watanabe, J.
Grodstein, and H. Harkness, "Logic Decomposition
During Technology Mapping", Digest Int. Conf. On
Computer Aided Design, pp. 264-271, San Jose, CA
Nov. 5-9, 1995. However, this proposal does not
provide a good method for choosing the constant
delays and, in addition, it only applies the
constant delay model to mapping.

Similarly, the constant delay model is proposed
for fanout optimization. See, K. Kodandapani, J.
Grodstein, A. Domic, and H. Touati, "A Simple
Algorithm For Fanout Optimization Using High
Performance Buffer Libraries", Digest Int. Conf. On
Computer Aided Design, pp. 466-471, Santa Clara,

Nov. 7-11, 1993. While the above-mentioned references note the importance of the constant delay model, they do not note the importance of gain, also termed "electrical effort".

In Sutherland and R. Sproull, "The Theory of Logical Effort: Designing for Speed on the Back of an Envelope", Advanced Research in VLSI, pp. 3-16, University of California, Santa Cruz, 1991, the delay is noted as being dependent on gain. A size independent formulation of the delay optimization problem is also presented, but the solution is intended for use as imprecise, scratch-pad type calculations, and not part of an overall integrated, automated solution to cell placement in the design of integrated circuits. In V. Kumar, "Generalized Delay Optimization of Resistive Interconnections Through an Extension of Logical Effort, Proc. Int. Symp. On Circuits and Systems, 1993, vol. 3, pp. 2106-2109, Chicago, IL, May 3-6, 1993, the methods above are used to analyze long wires with a significant amount of RC delay.

In U.S. Patent No. 5,654,898, a method and
apparatus for determining an integrated circuit
layout is shown and described whereby timing-driven
buffer sizing is performed to satisfy timing
requirements. However, this solution also relies
on imprecise calculations and does not teach or
suggest an overall integrated, automated solution
to cell placement in the design of integrated
circuits.

What is needed and what has been invented is a
method and apparatus for overcoming the foregoing
deficiencies, and for maintaining timing closure
upon placement and routing of the digital circuit
or network.

Summary of the Invention

The present invention broadly provides a method for designing an integrated circuit layout based upon an electronic circuit description and by using a cell library containing cells that each have an associated relative delay value, comprising the steps of:

(a) selecting a plurality of cells that are intended to be coupled to each other with a plurality of wires and that can be used to implement the digital circuit based on the electronic circuit description;

(b) determining an initial intended location of each of the selected plurality of cells on the integrated circuit, the step of determining the initial intended location of each of the selected plurality of cells including the step of determining an initial intended area of each of the selected plurality of cells, the initial intended area of at least some of the selected plurality of cells being determined using the associated relative delay value of the selected cell and the

initial intended lengths of some of the wires

coupled to the selected cell;

(c) finalizing the location and area of each

of the selected plurality of cells and the lengths

of the wires; and

(d) routing the digital circuit to generate

the integrated circuit layout using the finalized

location and area of each of the selected plurality

of cells and the finalized wire lengths.

The present invention makes possible an

advantage of facilitating the maintenance of timing

closure throughout the design process. The

invention makes possible another advantage of

compensating for the unpredictable delay effects

due to the placement and routing steps in the

design process.

Still another advantage made possible by the

invention is the avoidance of costly timing

analysis of conventional approaches. In contrast,

timing analysis is required during placement and

throughout the conventional design process.

The following are also made possible by the invention by establishing circuit area, instead of timing, as an optimization objective. First, excess circuit area which is formed in one module can be compensated in another module. In contrast, excess timing in one module cannot generally be compensated by another module. Second, area is a measurable parameter when comparing two different circuits. In contrast, it is difficult to compare two circuits for performance advantages, based on timing, since timing may be different for every circuit path. Third, if circuit area is established as a constraint, then routing completion is not guaranteed to be achieved.

Brief Description of the Drawings

Fig. 1 is a schematic diagram of a host computer system which is capable of implementing the present invention;

Fig. 2 is a schematic circuit diagram of a combinational network which can be designed by the computer system of Fig. 1;

Fig. 3 is a schematic circuit diagram of a sequential network which can be designed by the computer system of Fig. 1;

Fig. 4 is a flowchart describing a method of the present invention;

Fig. 4A is a schematic diagram of a real, non-ideal inverter;

Fig. 4B is a schematic diagram of a NOR gate;

Fig. 5 is a flowchart describing a method of mapping;

Fig. 6A is a schematic diagram of a portion of a circuit shown for discussion of "arrival" time calculations;

Fig. 6B is a schematic diagram of the circuit of Fig. 6A shown with its next circuit stage;

Fig. 6C is a schematic diagram of the circuit of Fig. 6B shown with its next circuit stage;

Fig. 7A is a schematic diagram of a gate chain with four levels of logic;

Fig. 7B is a schematic diagram of the gate chain in Fig. 7A after transformation wherein the number of levels of logic is reduced;

Fig. 8 is a flowchart describing a method of buffering according to the invention;

Fig. 8A is a schematic diagram of a portion of a digital circuit wherein buffers are inserted to optimize network area;

Fig. 9 is a schematic diagram of a portion of a digital circuit shown for describing a method for stretching and compressing of gate delays;

Fig. 9A is a graph showing the relationship between the gate gain C_{out}/C_{in} and the gate delay D;

Fig. 9B is a flowchart describing a method for stretching and compressing gate delays according to the present invention;

Fig. 10A is a schematic block diagram of a logical hierarchy of a network according to the invention;

Fig. 10B is a schematic block diagram of a physical hierarchy of a network transformed from the logical hierarchy of Fig. 10A;

Fig. 10C is a partial top elevational view of a plurality of "buckets" wherein cells are placed inside the buckets; and

Fig. 11 is a partial perspective view of two wires in a net shown for describing adjustment of capacitive load values to control timing constraints after routing.

Detailed Description of the Invention

Those of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to those skilled in the art.

Referring in detail now to the drawings wherein similar parts or steps of the present invention are identified by like reference numerals, there is seen in Fig. 1 a schematic diagram of a host computer system 100 which is capable of implementing the present invention. The host computer system 100 preferably includes a computer 105, a display 110, a printer 115 and a plotter 120. The computer 105 is typically a personal computer or workstation. The display 110 displays both graphical and textual materials relating to the design of integrated circuits. Also included in the computer system 100 is a keyboard 130 and a pointing device 135 such as a "mouse". When operating with software tools used in integrated

circuit design, the computer system 100 essentially

becomes a series of electronic circuits for

accomplishing specific design functions.   One

example of such software tools is the Aquarius

which is commercially available from Avant!

Corporation of Fremont, California.

Fig. 2 shows a portion of a typical integrated

circuit 150 which can be designed by the computer

system 100 (Fig. 1) and which is provided as

additional background material for understanding

the present invention.   The digital circuit 150

comprises a plurality of gates, including gates i,

j, and k.   The gates can be combinational gates

whose function is represented as Boolean expression

based on, for example, the operators AND, OR and

NOT.   The gates can also be registers.

Each gate (e.g., gate j) has one or more input

155 and a single output 160.   The gates in the

circuit 150 are coupled together by a plurality of

nets.   For example, the gate i has a first input

connected to a net 165a and a second input

connected via net 165b to the output of the gate k.

The gate j has a first input connected via net 165c

to the output of gate i, a second input connected

to the net 165d, and an output connected to the net

165e.

Gates whose outputs are connected to the inputs

of a gate are collectively called the "fanin" of

the latter gate. Thus, the gate k is in the fanin

of the gate i. Gates whose inputs are connected to

the output of a gate are collectively called the

"fanout" of the latter gate. Thus, the gate j is

in the fanout of the gate i.

The digital circuit 150 performs a logic

function by processing digital binary input data in

a number of cycles. The input data is presented to

the digital circuit 150 at the primary inputs 170,

and the result of the computation of the digital

circuit function is presented at the primary

outputs 175. Typically, the computation of the

digital circuit function requires one or more

cycles. During each cycle, the gate functions are

calculated, and the calculation results are stored

in registers for use in the next cycle.

The circuit 150 of Fig. 2 is known as a

combinational network. Fig. 3 is a schematic

diagram of a sequential network 180 which can also

be designed by the computer system 100 (Fig. 1).

In the sequential network 180, the feedback loops

typically include at least one register. For

example, the feedback loop 182 includes the

register 184.

Fig. 4 is a flowchart describing an overview of

the automated method of the present invention that

is implemented using a computer program containing

sequences of instructions that implement each of

the various steps described hereinafter. The

automated program thus implements a method that

includes the following steps below. It is noted

further that in the flowchart of Fig. 4, each of

the steps has its own distinct computer

instructions (i.e., each of the steps 200-230

comprises further steps). In step 200, a netlist

generated by synthesis (or other circuit

description) is read.   In step 205, structural

optimizations are performed to reduce the number of

connections in the digital circuit, while

maintaining the overall circuit function.

Additionally, library analysis 207 of library 209

may be performed during, or prior to, step 205.   As

discussed in further detail below, during the

library analysis 207, the delay is determined for

each gate to be used in the digital circuit.   In

step 210, the digital circuit is mapped to the

library 209 of cells.   In step 215 buffers may be

added as an option to the digital circuit to

optimize the circuit area.   In step 220,

"stretching" and/or "compressing" of gate delays

may be performed to assist in satisfying the

predetermined timing constraints and to optimize

the network area.   In step 225, cell placement

occurs, during which the size of a cell is

determined to maintain the constant cell delay

characteristics according to the present invention.

In step 230, network routing is performed, and the

net load of a cell may be adjusted to further

tradeoff between timing constraints and circuit

area. After step 230, the chip layout is generated

(step 235).

Read Netlist

In step 200 (Fig. 4), the invention will read a

netlist which is a detailed interconnection listing

of boxes in the target technology from which

automated logic diagrams may be produced for

integrated circuit fabrication. The netlist is

typically in Verilog or VHDL format and is

generated from synthesis tools such as the Synopsys

Design Compiler™ from Synopsys, Inc. or BuildGates™

from Ambit Design Systems, Inc. The synthesis tool

performs parsing of the complex Verilog or VHDL

constructs which describe the desired logic

function. The synthesis tool further performs

sequential optimization such as behavioral

synthesis, retiming, resource sharing, scheduling,

state machine optimization, and register mapping.

Library Analysis

During, or prior to, step 210 in Fig. 4, a
library analysis 207 is performed on the library
209 which will be used for the circuit design.
Library analysis 207 is not dependent on the actual
circuit which is being synthesized and thus can be
performed separately from the circuit design
process shown in Fig. 4. During the library
analysis 207, the delay D is determined for each
gate to be used in the digital circuit. Assume a
gate in the library 209 has a capacitive load value
C and a size value S wherein S is a continuous,
positive real variable which linearly increases
both the driving capability and area of the gate.
The delay D of a gate can be approximated by
equation (1):

(1) $D = f(C/S)$

The delay D is non-negative and increases as the
C/S value increases.

The delay D value depends on the delay provided
by a cell plus the delay provided by the net (wire)
load of the cell. Thus, in order to ideally

maintain the delay D at a constant value prior to and after placement/routing, the net load resistance must be taken into consideration. According to the invention, the gate size is adjusted after cell placement based on changes in the capacitive load in order to maintain D as "constant," as will be described hereinafter in further detail. However, since the wire load also has resistance, the value of D (after gate sizing) may actually vary by a negligible amount from the pre-determined D value. Since this variation is negligible, D may be approximated as "constant" prior to cell placement and after cell placement and gate size adjustment.

The delay D value may also be different for different inputs of the gate and it may also be different for the falling transition and rising transition of a signal propagating through the gate. The C/S value is defined as the "typical load" since it is the load value which can be driven by a gate of size "1". The library analysis will determine a "good" value for C/S for each gate

in the library based on gain considerations. It is known to those skilled in the art that to attain maximum gain in a set of series-connected ideal inverters, the gain of each inverter should be e, which is about 2.7. However, an actual inverter is non-ideal and therefore has a parasitic self-load which results in a parasitic delay which is fixed and independent of the size of the inverter and of the load capacitance driven by the inverter. The principal contributors to the parasitic delay are the capacitance of the source/drain regions of the transistors and the Miller capacitance between the source/drain and the gate of transistors which drive the output of the inverter. If this parasitic delay is taken into account, then to obtain maximum gain in the series-connected inverters, the gain of each inverter is set at greater than 2.7, for example about 3.6.

Discussion is now turned to the method of finding the preferred gain of gates other than inverters. For any circuit or network, a good operating point is the point at which a good trade-

off between gain and delay can be achieved.   To
optimize this trade-off, the "continuous buffering
assumption" is used which is based on the ideas
described in I.E. Sutherland and R.F. Sproull, "The
Theory of Logical Effort: Designing for Speed on
the Back of an Envelope," *Advanced Research in
VLSI*, pp. 1-16, UC Santa Cruz, 1991.

The method for finding the constant delay is
based on a comparison of the gate to an inverter.
The method is based on the idea that, at some
point, it becomes more advantageous to use a buffer
to increase the gain rather than to increase the
gain of the gate itself.   By adding more buffers,
wherein each added buffer is of a larger size than
a previously added buffer, the gain of the
combination (gate and buffers) can grow
exponentially as delay increases.

In the following analysis, it is assumed that
continuous buffering can be made whereby it is
possible to insert an infinitesimally small amount
of buffering.   On a given path with several gates,
several fractional buffers can be combined into a

single real buffer. Thus, for a circuit with many levels of logic, continuous buffering may be achieved approximately or approaches reality.

Assuming continuous buffering can be made, the delay of a series of ideal inverters can be written as $\tau\log(h_{inv})$ wherein τ is a time constant and h_{inv} is the inverter gain. In other words, the gain of a series of buffers increases exponentially with its delay.

Consideration is now turned to the delay of a gate which is coupled to a series of ideal inverters. As shown in equation (2a), the delay d_{gate} of the gate partly depends on the load of the gate, wherein the load of the gate is the load of the combination (C_{out}) divided by the gain h_{inv} of the inverters.

$$(2a) \qquad d_{gate} = P_{gate} + R_{gate}(C_{out}/h_{inv}) + \tau\log(h_{inv})$$

The parameter P_{gate} is the intrinsic delay of a gate, while R_{gate} is the gate resistance.

To minimize d_{gate} for a given C_{out}, the derivative of d_{gate} (equation (2a)) is taken with

respect to h_{inv}, and the result is shown by equation (2b).

(2b) $\partial d_{gate}/\partial h_{inv} = (R_{gate}C_{out}/h^2_{inv}) + (\tau/h_{inv})$

By setting equation (2b) equal to zero (0), equation (2c) is derived.

(2c) $R_{gate}C_{out} = \tau h_{inv}$

At the point where a buffer is not needed, $h_{inv} = 1$ and $R_{gate}C_{out} = \tau$. By substituting equation (2c) into equation 2(a), the delay of each gate is the intrinsic delay P_{gate} plus the time constant τ.

The interpretation of using this constant delay is that this delay is the cross-over point between using buffers versus stretching for gain. In other words, the delay of a gate on a critical path should never be slower than this cross-over point value. This provides a clear criterion for dealing with gain or delay optimization problems. If more path gain is required, a buffer is a faster solution than slowing down the gate. Also, use of a buffer will be a more area-efficient solution.

These results above are used as follows. For a given library, the inverter is first analyzed. Using the result that the gain of an inverter should be about 3.6, the typical load C_{typ} is derived as shown by equation (2d):

(2d) $C_{typ} = (3.6) C_{in}$

wherein C_{in} is the input capacitance of the inverter. The typical load is used to derive the constant delay for the inverter. Since this is a real, non-ideal inverter, the following rule is applied, as shown in equation (2e).

(2e) $R_{inv} C_{inv} = \tau$

R_{inv} is determined from a delay equation as represented by equation (2f).

(2f) $R_{inv} = \partial d_{inv} / \partial C_{inv}$

Applying the same rule to all gates in the library yields equation (2g).

(2g) $(\partial d_{gate} / \partial C_{gate}) (C_{gate}) = (\partial d_{inv} / \partial C_{inv}) (C_{inv})$

Throughout the following example, as shown in Figs. 4A and 4B, it is assumed that the resistance R_{inv} for a P-channel transistor is twice as high as

the resistance R_{inv} for an N-channel transistor of

the same width. Reference is first made to Fig. 4A

which shows a schematic diagram of a real, non-

ideal inverter 250. The inverter 250 includes the

P-channel transistor 252 and the N-channel

transistor 254. If it is assumed that the

transistors 252 and 254 each have a resistance R_{inv}

of 1 unit, then the P-channel transistor 252 has

twice the width of the N-channel transistor 254.

Assume further that the input capacitance C_{in} of the

inverter 250 is 3 units (wherein the N-channel

transistor 254 has an input capacitance of 1 unit

and the P-channel transistor 252 has an input

capacitance of 2 units). The typical load of the

inverter 250 can be derived using the fact that the

gain of a non-ideal inverter should be about 3.6.

Thus, based on equation (2d), the typical load C_{typ}

is shown by equation (2h).

(2h) $C_{typ} = (3.6)C_{in} = (3.6)(3) = 10.8$

The typical load value C_{typ} can be used to derive

the delay of the inverter 250 wherein $R_{inv} = 1$ and τ

$= R_{inv}C_{inv} = (1)(10.8) = 10.8$.

Referring now Fig. 4B, the NOR gate 256

includes the P-channel transistors 258 and 260 and

the N-channel transistors 262 and 264. Assuming

all the transistors in Fig. 4B have the same size,

then the pull-up resistance is 4 units (through

both P-channel transistors 258 and 260) and the

pull-down resistance is 1 unit (through either the

N-channel transistor 262 or the N-channel

transistor 264). The typical load is determined

based on equation (2i).

(2i) $R_{gate}C_{typ} = \tau$

For the pull-up resistance wherein $R_{gate} = 4$ and $\tau =$

10.8, equation (2i) leads to $C_{typ} = 2.7$. For the

pull-down resistance wherein $R_{gate} = 1$ and $\tau = 10.8$,

equation (2i) leads to $C_{typ} = 10.8$. It is

undesirable to have different answers for C_{typ} for

the same gate. Since there can only be one real

capacitance in the NOR gate 256, a number of

options are available. For example, the average

between the two C_{typ} values (2.7 and 10.8) may be

chosen. Alternatively, the most constraining value

for C_{typ} may be chosen. In general, such an

unbalanced gate as shown in Fig. 4B cannot be tuned

to work well for both the rising and falling

transitions of a propagating signal. A large

discrepancy indicates a poorly dimensioned gate.

As stated above, each gate has more than one

delay equation due to multiple gate inputs. Also,

different delay equations may correspond to the

rising signal transition and to the falling signal

transition.

Structural Optimizations

In step 205 (Fig. 4), the invention performs

structural optimizations after reading the netlist

from step 200. During this step, the structure of

the circuit and the Boolean functions of the gates

are changed to reduce the total number of

connections, without changing the overall function

of the circuit. Structural optimizations can include behavioral optimizations (such as resource sharing), sequential optimizations (such as retiming), algebraic optimizations (such as kernel extraction), and Boolean optimizations (such as redundancy removal). The classes of optimizations above are well known to those skilled in the art.

Mapping

In step 210 (Fig. 4), the circuit is mapped to a library 209 of cells. Thus, the logic functions of the circuit gates are implemented with actual cells from the library 209. By mapping, circuit delay and other physical attributes can also be determined. In step 210 of the present invention, a conventional mapping process may be used. Fig. 5 shows the mapping process (step 210) as further including the additional steps of decomposition 300, cluster generation 305, matching 310, and covering 315. The decomposition step 300 breaks down the large commutative gates into two-input commutative gates.

The cluster generation step 305 selects matches in the logic. Clusters with low chance of success are preferably avoided, such that the total number of clusters to match do not grow very large.

The matching step 310 is the process of finding a library function which can implement the function of the cluster. This step is implemented by use of a conventional matching algorithm such as the algorithm described in K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching", Proceedings of the 24th ACM/IEEE Design Automation Conference, Miami Beach, FL (June 1987), pp. 341-347, IEEE Computer Society Press 1987.

The covering step 315 determines which matches are used to actually implement the digital circuit. A timing driven covering method based on dynamic programming may be used in this step. According to this covering method, an implementation of a circuit is chosen from a set of possible implementations based on the arrival time of an implementation. (An arrival time of the data at a gate is computed by taking the maximum arrival time

of the fanin gates plus the delay measured from the
input pin to the output pin of the gate).   An
implementation of the circuit with the fastest
arrival time may be chosen during the covering step
315.   In contrast to the present invention, a
conventional mapping process has a disadvantage of
not achieving the accurate calculation of "arrival
times," since the arrival times depend on the
chosen implementations at multi-fanout points.   In
the conventional process, the chosen
implementations at such multi-fanout points are
mutually dependent.

Discussion is now turned to calculation of the
arrival time for an implementation of a circuit
according to the present invention.   Reference is
made to Figs. 6A-6C which show schematic diagrams
of portions of a circuit in order to describe
arrival time calculation for a gate.   In Fig. 6A,
the arrival time for a gate 330 (in stage 335) is
determined by the arrival times of its fanin gates
(at lines 340, 345 and 350) plus the signal delay
from input terminals 355 to the output terminals

360 of the gate 330. The configuration of the
fanouts at stage 365 are thus not factored in the
arrival time calculation for gate 330. In Fig. 6B,
the arrival time for a gate 370 is determined by
the arrival times of its fanin gates 330 and 372
plus the delay between input terminal 375 and
output terminal 380 of gate 370. Similarly, in
Fig. 6C, the arrival time for a gate 385 (in stage
387) is determined by the arrival times of its
fanin gates 370 and 390 plus the delay between its
input terminal 395 and output terminal 400.

The process above is repeated for the gate 402
which has an output coupled to an output terminal
404. Thus, the arrival time at an output terminal
404 of the circuit of Figs. 6A-6C can be determined
based on the arrival time of the gate 402.

Thus, as shown in Figs. 6A-6C, in the constant
delay approach of the invention, other fanouts of
multi-fanouts are irrelevant to the arrival time
calculation. The arrival times can thus be
computed by traversing the circuit from the inputs
to the outputs in a levelized manner. By

determining the arrival time for gates on a given

path, the total delay of the path in the circuit

can be determined.

Level Reduction

Before the cell placement step (step 225 in

Fig. 4), it is possible to further revise the

circuit than conventionally possible, due to the

fact that the delays are held constant. Since the

timing constraints are met and the circuit

modifications to be made are known, it follows that

the effects on delay can be predicted when circuit

modifications are made. One specific change or

modification which can be made in the mapped

circuit is logic level reduction. Reference is

made to Figs. 7A and 7B to illustrate an advantage

of using the constant delay model according to the

invention. Fig. 7A is a schematic diagram of a

gate chain 550 including gates 555, 560, 565, and

570. Initially, the gate 555 has fanins connected

at lines 575 and 580. The gates 560, 565 and 570

have fanins connected at lines 585, 590 and 595,

respectively. A local transformation is then used

to reduce the number of levels in the logic in the

gate chain circuit 550. The result of the

transformation is shown as gate chain circuit 550'

in Fig. 7B. The number of levels in the logic is

reduced by bringing the gate 555 forward. In the

constant delay model approach, the effect of this

transformation can be easily predicted. Changes in

the gate loads do not affect delay, since delay is

maintained as constant while gate size will be

adjusted (during or after placement) to compensate

for the load change. The only change which affects

delay (of the gate chain circuit 550) is the change

of the fanin of gate 555. This delay change can be

predicted by simple addition of gate delays

provided by the fanins connected at lines 590, 575,

and 580 (see gate chain circuit 550' in Fig. 7B).

In order for the transformation shown in Fig.

7B to be valid, it is necessary that gates 555,

560, and 565 are fanout free. If the gates 555,

560, and 565 are not fanout free, then they are

made fanout free through copying logic. For

example, assume in the circuit 550 (Fig. 7A) that

the gate 555 has an additional fanout 597. The

gate 555 in the circuit 550' (Fig. 7B) can no

longer drive the additional fanout 597 since the

gate 555 function may have changed in the circuit

550'. Thus, in the circuit 550', there is provided

the copying logic 555' (which is a duplicate of the

gate 555 of circuit 550) for driving the fanout

597.

In contrast, a disadvantage of the conventional

design approach is as follows. Under conventional

logic design, copying logic will increase the load

on the gates whose outputs are connected to lines

575, 580, 585, and 590. In the example of Fig. 7B,

the copying logic 555' increases the load on the

gates whose outputs are connected to lines 575 and

580. To predict whether or not the transformation

improved delay, it is necessary to run a complete

static timing analysis with accurate delay models.

If the transformation (from circuit 550 to 550')

were actually harmful to delay, then the

transformation would have to be undone.

Area Analysis and Net Weights Calculations

Discussion is first made on area analysis during which sizes of gates of a mapped circuit are calculated. A gate size is calculated by dividing the gate's actual load by the gate's predetermined typical load C/S. As an example, assume the actual load for a given gate i is expressed as C_i wherein C_i is a capacitance measurement of the gate load. The actual load C_i is approximated by equation (3).

(3) $C_i = \omega_i + \sum C_j * (1/h_{ij})$

The parameter ω_i represents the net (wire) load for a given gate i (wherein the net load can be estimated using a conventional net load model such as the above-mentioned fanout-based model) plus any other fixed load such as the load of the primary output of the circuit implementation. The parameter $\sum C_j * (1/h_{ij})$ is the input load wherein the subscript j is the fanout gate of the given gate i, and h is the electrical effort between the gates i and j.

Equation (3) can be expressed in matrix notation as shown in equation (4):

(4) $\overline{c} = \varpi + (\overline{H})(\overline{c})$

wherein $\overline{H} = [1/h_{ij}]$ which is the matrix of the electrical effort. The parameter $\overline{c}$ is the vector of the gate loads wherein $\overline{c} = [C_i]$ which is the matrix of the gate loads, while ϖ is the vector of the wire loads ω_i.

According to equations (3) or (4), in order to calculate the size of a given gate i, the size of its fanout gate j must first be calculated, as shown with respect to Figs. 6A to 6C. The size of the fanout gate j depends on its output capacitive load C_j. This dependency on the fanout gates requires performing the gate load calculation downstream of the data flow. If the digital circuit is a combinational network (see, e.g. circuit 150 in Fig. 2), then gate load calculation initiates at the primary outputs 175 and traverses the circuit in a leveled order toward the primary inputs 170.

If the digital circuit is a sequential network
(see, e.g., circuit 180 of Fig. 3), then there may
be one or more loops (e.g., loop 182) which result
in a cyclic dependency (i.e., there is no
"rightmost" gate). Gate load calculation can start
anywhere in the cycle, and calculation in the cycle
is performed several times until the load
capacitance values converge or have sufficiently
small differences. However, a condition may exist
when the load capacitance values do not converge
and increase by progressively larger amounts every
cycle calculation. This increase in load
capacitance values can be detected if the
calculated load values exceed a preset maximum
value after a fixed number of cycle calculations.
When the calculated load values do not converge,
then the particular circuit 180 has an infeasible
solution, which indicates that the digital circuit
is not expected to work at the set speed because
the circuit gain is too small. Changes are
required to increase the circuit gain, and these

changes will usually lead to an increase in circuit delay.

In the above example, the size S of a gate i is determined by dividing the actual load C_i by the predetermined typical load C/S of the gate i. The size S is a scale factor which is applied to all transistor channel widths of a gate in order to determine the area of the "sized gate". The size S is also a scale factor which is used to scale the gate's output load driving capability and its input pin loads.

The area of the sized gate is determined by equation (5).

(5) area of sized gate = S * (area of gate)

The area of the mapped digital circuit can be estimated based on the sum of the total areas of the sized gates plus the net area (which is estimated from the total length of all nets in the circuit).

During the mapping step, possible circuit implementations may be evaluated based on net weight calculations. Thus, the following

discussion now turns to the calculation of "net

weights." The net weight represents the

sensitivity of the total area of a digital circuit

with respect to the load of a particular net. As

an example, the net weight of a given gate, which

is immediately coupled to the primary inputs of a

digital circuit, is equal to its area per unit

load. Using equation (6), the net weight of the

other gates in the digital circuit are then

calculated in a leveled order towards the primary

outputs of the digital circuit.

(6) $w_i = \partial A/\partial C_i = \partial a_i/\partial C_i + \sum w_j * (1/h_{ij})$

In equation (6), the subscript j represents the

fanin gate of a given gate i. The parameter w_i is

the net weight of the gate i, while A is the total

area of the digital circuit, a_i is the area of the

gate i, and C_i is capacitive load of the gate i.

As stated above, h_{ij} is the electrical effort

between a gate i and j.

Equation (6) can be expressed in matrix

notation as shown in equation (7):

(7) $\overline{w} = \overline{\alpha} + (\overline{H})(\overline{w})$

wherein $\overline{H} = [1/h_{ij}]$, $\overline{\alpha} = [\partial a_i/\partial C_i]$, and $\overline{w} = [w_i]$.

The net weight value from equation (6) can be also used to later calculate the cell area of the cells after the placement step (step 225 in Fig. 4). This cell area is expressed by Equation (8):

(8) $A = \sum (w_i)(\omega_i)$

wherein the parameter ω_i represents the wire load for a given gate i.

During mapping, the calculated net weight value can be used to determine which matches are used to implement the digital circuit, since the calculated net weight value makes it possible to take into consideration the circuit area. As shown in equation (6), the net weight for a given gate i only depends on the logic of the fanin cone (which was already mapped), wherein the net weight represents the sensitivity of the total area of the circuit with respect to the load on a cell. As further shown in equation (8), the area of a cell increases if its net weight increases. Thus, for example, assume that for a desired circuit

function, a first circuit implementation has a higher net weight than the net weight of a second circuit implementation. The first circuit implementation will therefore have a larger area for any possible load than the second implementation. Therefore, disregarding delay, the second circuit implementation is more preferable than the first circuit implementation, even though the area values of both circuit implementations may have not been determined at this stage of the design process.

As shown by the above example, the net weight calculations permit circuit implementation decisions to be made, partly based on the gate area. The net weight calculations permit the area of one cell to be compared relatively to the area of another cell wherein both cells have the same function in the desired digital network. An advantage of the net weight calculations is that when comparing the area of cells, the actual area values of the cells are not required to be known.

Buffering

Prior to cell placement, certain matches may be changed by insertion of buffers in the digital circuit, as shown in step 215 in Fig. 4. Under this step, adding a buffer will increase delay, but also save on area since the source gate can be smaller in size due to a smaller load. Large loads are preferably driven by buffers, particularly when the signal timing is not critical. Buffering allows the source cell to be sized down, thereby resulting in a reduction in circuit area.

The buffering step of 215 (Fig. 4) is discussed in further detail with reference to Fig. 8. In step 650, locations in the circuit are determined where a buffer can be added so that buffer insertion will still permit timing constraints to be met. This determination is made by subtracting the delay of the buffer from the "local slack", to give the value of the predicted slack after buffer insertion. Slack is zero or positive if the timing constraints are met. In addition, all slacks in the circuit can be summarized by the "network