Chapter 1

Clock Tree Generation

When complexity (i.e. number of gates in a design) increase, the need to distribute clock signals in a controlled manner becomes more important. A large, pipelined chip may easily contain hundreds of clocked elements (latches, flip-flops, etc.), so it is obvious that a lot of buffering is needed. Since it is usually desired to have the clocked parts switch at the same time, care has to be taken when designing the clock distribution.

This chapter describes one tool, *Envisia^{TM} Clock Tree Generator*[1], which can create clock trees according to various time constraints.

1.1 Overview

In order to run CT-Gen the normal design flow in Silicon Ensemble is broken up after the placement stage and a def-file describing the design is saved. This file is then fed into CT-Gen along with some library files and after the clock tree has been generated the design is imported back into SE for routing, see figure 1.1. CT-Gen can be called from within SE or it can be run as a stand-alone tool, this manual will describe the procedure for the latter.

Based on the constraints given CT-Gen will insert buffers and inverters, forming a tree structure, into the clock distribution. Any existing buffering in the clock path will first be removed. The available components are picked from the timing file read into the generator.

This tool can of course be used on other heavily loaded signals, such as reset, but that is more complicated procedure since a lot of inputs then have to be defined as clocked.

[1] Trademark of Cadence Design Systems, Inc.

CT-Gen works on a placed netlist. This makes it possible to estimate the extra delays caused by the wire capacitances since the program knows the distance between the clock tree components and how to calculate the parasitic capacitance of the wires.

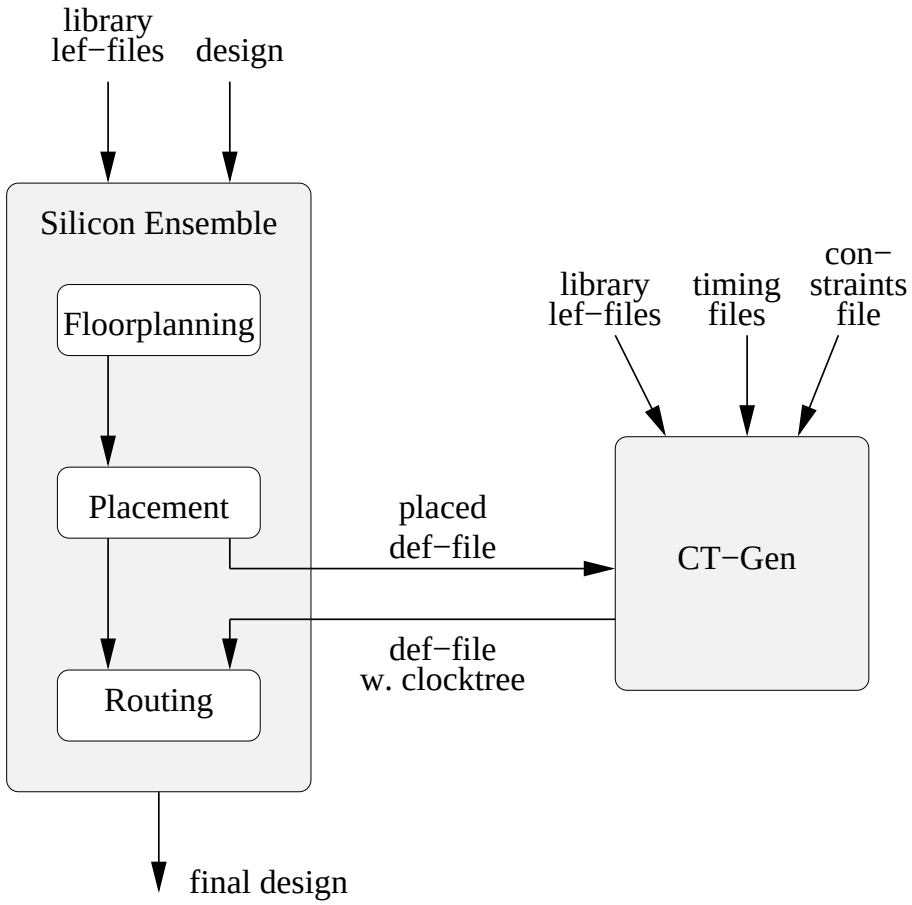

Figure 1.1: The design flow for clock tree generation.

- **lef-file:** Description of the standard cells available, logic gates and IO-pads. It also contains the layout rules for routing the design and process parameters for delay calculation.
- **def-file:** The netlist of the design, used cells and how they are connected.
- **timing-file:** This is a vendor-supplied file with information of the timing characteristics for the standard cells. A functional specification is also included to enable the program to select suitable buffer components for the clock tree.

A clock tree might look something like the one in fig 1.2 which depicts a three level clock distribution. The tree will be built from the designated **root pin** to the clocked **leaf pins**. All the clock inputs of the standard components are defined as clocked in the timing file. If the clock is gated it might be necessary to define that input as clocked, otherwise CT-Gen will try to trace the clock net through that gate. This is true for the output pads, for instance.

The names of the inserted components and nets are derived from the original ones, with a level number added. R_L2_I1 means a component on level 2 in the tree, from the root (R).

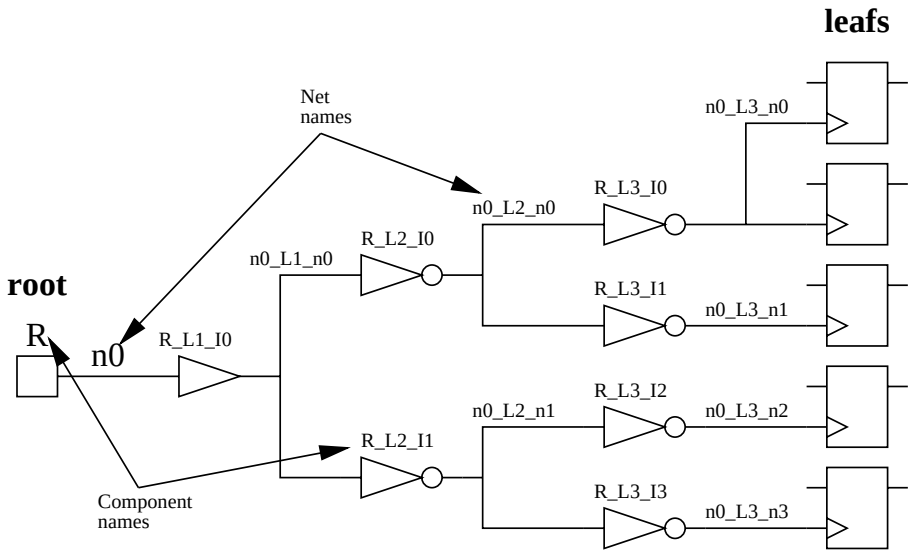

Figure 1.2: A three level clock tree.

1.2 Function

In order to work properly CT-Gen needs two user-supplied files in addition to the library definition and the timing files. These are the **Command File** and the **Constraints File**.

1.2.1 Command File

The **Command File** is a set of commands to be executed by the program and might look like this

```
set_rundir 'CTGEN'
set_format se
read_technology
  lef_files 'file1' 'file2' ...
  ctlf_files 'ctlf1' 'ctlf2' ...
read_design
  def_file 'placed.def'
read_constraints
  constraints_file 'ctgen.constraints'
build_clocktrees
remove_overlaps
write_design
  def_file 'ctgen.def'
```

The commands has the following functions

- **set_rundir:** Specifies a directory to use for the run. It will be created if it doesn't exist. All log, report, and other necessary files will wind up here.
- **set_format:** Designates the intended place-and-route tool. Use 'se' for Silicon Ensemble.
- **read_technology:** Inputs all needed technology- and timing data.
- **lef_files:** The files containing the rules for routing and the outlines for the involved cells and blocks.
- **ctlf_files:** Compiled timing library files. These files describes the delay for the different cells according to the load on them. It is also used to select the buffers and inverters to be used in the clock tree. It can also contain some constraints (maximum rise-time on the inputs for example).
- **read_designs:** Read in the design itself.
- **def_file:** Denotes that it is a 'Design Exchange Format' file.
- **read_constraints:** The file with the user-defined constraints.
- **build_clocktrees:** Start the run.
- **remove_overlaps:** The inserted cells are initially placed without consideration of any existing ones. This command instructs CT-Gen to shuffle around the cells to make room for the added buffers.
- **write_design:** Write out the resulting file.

1.2.2 Constraints

Constraints are the restrictions given to CT-Gen. These are in the form of what delays that can be accepted in the clock distribution. The constraints that the user can specify are as follows, figure 1.3.

- **max insertion delay:** Maximum delay from root to leaf pin.
- **min insertion delay:** Minimum delay to leaf pin. This is usually set to 0 but in some cases a higer value is required.
- **max skew:** The time difference between the fastest and the slowest clock path. It is possible that CT-Gen finds several solutions satisfying the given set of constraints.
- **max transition time:** The 10% to 90% transition time at a leaf pin.

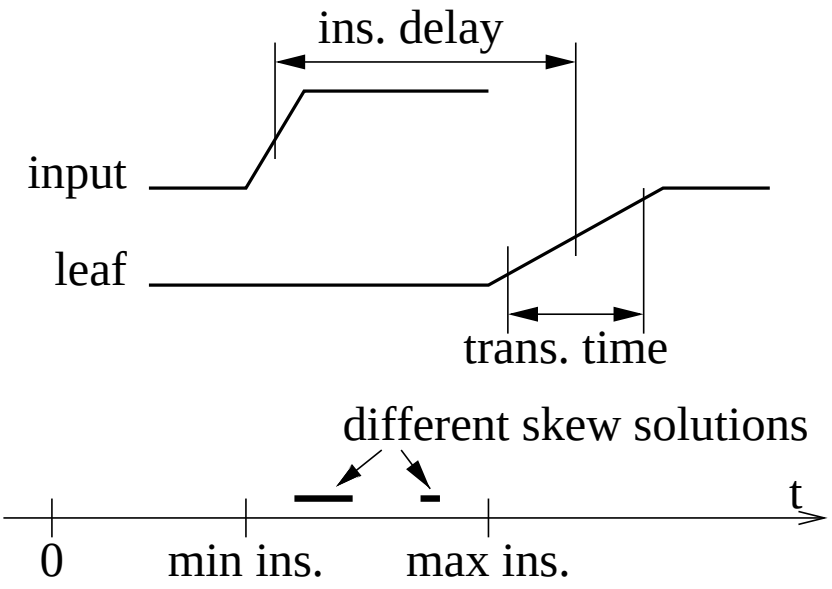

Figure 1.3: User Constraints.

The timing file of the library componets also contain a lot of constraints that CT-Gen has to take into account. These are of the form maximum transition time on inputs and max capacitive load on outputs.

```
#
# A nice constraints file
#
specify_tree             # First tree
  root_iopin 'pin'
  leaf_pins 'Component' 'pin' rising
            'Component' 'pin' rising
  leaf_ports 'Cell' 'pin' rising

set_constraints   # t in [ns].
  waveform t_rise  t_high  t_fall  t_low
  min_delay t
  max_delay t
  max_skew t
  max_transition t

specify_tree             # Second tree
  root_pin 'component' 'pin'
  no_gating rising

set_constraints
  ...
  ...
```

Command Explanation.

- **root_iopin:** Define pin as starting point for clock tree.
- **root_pin:** Use this if root is not a top-level pin.
- **leaf_pins:** Define specific pins as leafs, i.e. stop tracing the clock when encountered.
- **leaf_ports:** Leaf-pin on all instances of that cell.
- **no_gating:** Stop tracing at gates which is everything not defined as an inverter or buffer in the timing file.
- **waveform:** Description of what the input signal might look like.

When there is need for a Clock Enable (CE) signal, as in figure 1.4, that has to be set in order to activate the clock it can be handled vith the following command which must be **first in the constraints file.**

```
set_constants
  iopins 1 'CE'
# pins value 'comp' 'pin'   # If pin not on top level.
```

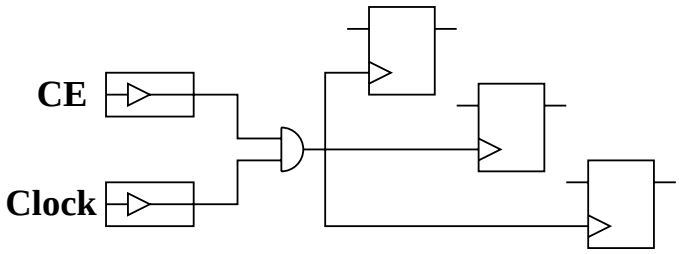

Figure 1.4: Clock Enable Configuration.

1.3 Usage

Follow this procedure when using CT-Gen.

- Generate a placed design by Using Silicon Ensemble. If there has to be power routing that covers the core area, stripes for example, that has to be done also. CT-Gen will not affect the power. Do not generate any core filler cells.
- Create a netlist with the command *Export > DEF*.
- Since this version of CT-Gen can not handle indexed components in the def-file it has to be modified. A small script performing some changes is available. Use it like

  $AMS_DIR/fixdeffile.def or *$ALCATEL_LIB/fixdeffile.def*

  depending on the design kit used.

  This script will change indexed component ($heffacomp[13]$) to $heffacomp\_13$. The input file will not be modified but a new one will be created, 'fix_file.def'.
- Compose the command and constraints files. Sample files for AMS or Alcatel will be generated by the *build_se_dir* script.
- Start the execution with the command

  ctgentool ctgen.commands

  from a window in which the Cadence environment has been set up.

1.3.1 Logs and Reports

If an error occurs when running, CT-Gen will stop and indicate a log file which contains information about the error. Log files will reside in the subdirectory *<CT-GEN>/log* where <CTGEN> is the catalog specified in the command file.

If the run is successful, a fair number of report files will be produced and placed in *<CTGEN>/rpt*. The major report files are of the form **<stage>.<type>** where **<stage>** is one of

- **initial:** Before any modifications.
- **clock:** After the clock tree has been added.
- **final:** When the clock tree components are placed correctly.

and **<type>** is one of

- **analysis:** A complete description of best and worst path found.
- **timing:** The same without the details.
- **trace:** Describes the clock path/tree with its components.
- **violations:** Lists any violations against the given constraints.

The file **clock.structure** shows the structure of the generated tree. For the tree in figure 1.2 it could look like this

```
define_structure
  from_pin 'R' 'Y'
    noninverting_tree 'BUF' 1
      to_rising_clock_pins 'INV' 2 'INV' 2
```

clock.changes lists the names for all added components and nets.

1.3.2 Back to SE

The resulting def-file can be read into Silicon Ensemble just like any other. A mac-file for this purpose will be generated when the SE directory is first initialized and might look like this for the AMS process.

```
##--
##-- loadctgen.mac
##--
FINPUT LEF F $AMS_DIR/artist/HK_0.35/LEF/csd/csd.lef ;
INPUT  LEF F $AMS_DIR/artist/HK_0.35/LEF/csd/HRDLIB_3M.lef ;
INPUT  LEF F $AMS_DIR/artist/HK_0.35/LEF/csd/IOLIB_3M.lef ;

INPUT DEF FILENAME "ctgen.def";
```

Do **not** load any pad-file since the pads is now included in the def-file.

The core can now be filled with the appropriate filler-cells and the design routed normally. It might, however, be wise to start by doing the clock net so that it can be routed as straight as possible. This can be achieved by the commands

```
set var wroute.timing.driven false ;
clockroute all ;
```

from the command line in SE.