SYNOPSYS®

DC Ultra Library Guidelines

October 1999

Table of Contents

Introduction 2

New Delay Model for Synthesis 2

Pre-Synthesis Library Analysis 3

- Logic Clustering 3
- NLDM Generation 4
- Input Capacitance Model Generation 4
- Summary of Library Analysis 5

Library Evaluation 5

- Function Classes 5
- Number of Sizes 6
- Size Consistency 6

Library Developers Guidelines 7

- Example Library Analysis 8

Using The New DC Ultra Algorithms 10

Introduction

Design engineers know that their EDA tools are highly dependent on the quality of technology libraries, especially for high-performance ASIC and system-on-a-chip (SoC) designs. Until now, tools would operate with varying degrees of success even if they had a sub-optimal library. Synopsys has added a new delay optimization algorithm to Design Compiler Ultra™ (DC Ultra) that offers push-button delay quality of results (QoR) improvement when using a suitable technology library. This algorithm uses a new logic synthesis load-independent delay model that is a hybrid between the constant delay model and the traditional non-linear delay model (NLDM).

A key feature of how this algorithm is used is an initial library analysis step that can automatically determine whether a technology library is suitable for the new optimization algorithm. This is important because the result of the new algorithm depends on the composition of the cell library. This paper is targeted to library providers who want to tune their libraries so that such high-end QoR capabilities can be fully utilized.

New Delay Model for Synthesis

The delay of cells in a technology library are typically represented using NLDM tables. Recently, a more simplistic linear delay model has been proposed for efficient logic synthesis of high-performance designs. This delay model is also referred to as the constant delay model. In this model, the delay of a timing arc of a gate, t, is represented as:

$$\tau = R \bullet Co + p$$

where R = output pin resistance
Co = output load of the timing arc output
p = intrinsic delay of a gate

This equation can also be represented as:

$$\tau = (R \bullet Ci) \bullet (Co / Ci) + p$$

where Ci = input capacitance of the timing arc input

The term $(R \bullet Ci)$ also refers to the logical effort of the gate and the term (Co / CI) also refers to the electrical effort or gain of the gate.

The constant delay model assumes that the delay of the timing arc remains constant. An important property of the constant delay model is that the delay of a timing arc is independent of load, i.e., the delay does not depend on either Co or Ci, but merely on the ratio of the two. This property is very powerful in the early stages of logic synthesis because the actual load of a gate is unknown at that time.

While this simplistic delay model is beneficial, in reality several other factors need to be considered for modeling the delay of a gate. Some of these include:

- Impact of transition times on delay
- Complex gates with different input capacitances for different input pins
- Limited discrete sizes in the technology library
- Design rules such as maximum capacitance and maximum transition associated with gates

The new DC Ultra delay optimization algorithm is significantly different from the constant delay model, because in the new algorithm, the delay of a gate is never constant. Moreover, it addresses the real-life issues mentioned above and uses this delay model for logic synthesis.

Pre-Synthesis Library Analysis

DC Ultra's new delay optimization algorithm analyzes the discrete cells in a technology library and derives a continuous (scalable) model for each analyzed logic function in the library. The optimization algorithm then works with this continuous model. At the end of optimization, the netlist is discretized, i.e., the scalable cells in the netlist are converted to discrete cells in the technology library. Following this, the traditional technology-dependent optimization techniques in Design Compiler are used for delay, design rules and area optimization. After library analysis, the algorithms automatically perform library evaluation to judge if a library is suitable for optimization.

Library analysis derives a scalable library from the discrete technology library for single output combinational cells. The cells in this scalable library are characterized with the new delay model. Library analysis is performed independent of whether a library is suitable for optimization. This library analysis consists of several steps including:

- Logic clustering
- NLDM generation
- Input capacitance model generation

- Logic Clustering

A discrete library typically has several cells (sizes) that implement the same logic function. These cells are designed to optimize various design parameters such as power, area, delay or porosity. To generate a scalable cell for any logic function, library analysis clusters cells that implement the logic function and have similar characteristics. For example, the gain of a discrete cell is defined as the ratio of output load capacitance to average input capacitance of the discrete cell. Therefore, cells in the same cluster should have similar delay numbers for all the timing arcs from input pins to the output pins for same gain values.

Figure 1 is a graph of the rise and fall delays of discrete inverter cells from a typical library with gain values set to three. On the x-axis of the graph is the input capacitance of the discrete inverter on a logarithmic scale (which is the same as average input capacitance). On the y-axis are the rise and fall delays of the corresponding inverter. The cells within the two black lines will most likely be included in the cluster as they have consistent delay characteristics while the remaining cells will be excluded. As shown by the graph, a cluster of eight cells will be created. Most of the cells not included in the cluster have been derived by internal buffering. Cells derived using internal buffering will most likely be ruled out during logic clustering as they typically have poor correlation with other non-buffered cells.

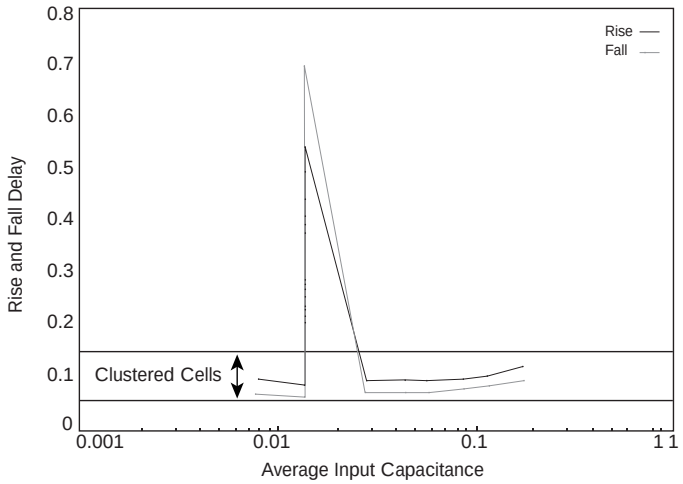

Figure 1: Delay vs. Capacitance

The input capacitance of corresponding input pins of discrete cells should have similar ratios. Figure 2 is a graph of the ratio of input pin capacitances of two input NAND gates in a typical library. For two input gates, similar values for the ratios of input pin capacitances imply similar ratios of input capacitance of the corresponding input pins of discrete cells. On the x-axis of the graph is the average input capacitance of the discrete NAND gates on a logarithmic scale, and on the y-axis is the ratio of pin capacitance. During clustering, cells above the line will most likely be included in the cluster while cells below the line will be excluded on the basis of inconsistent input pin capacitances.

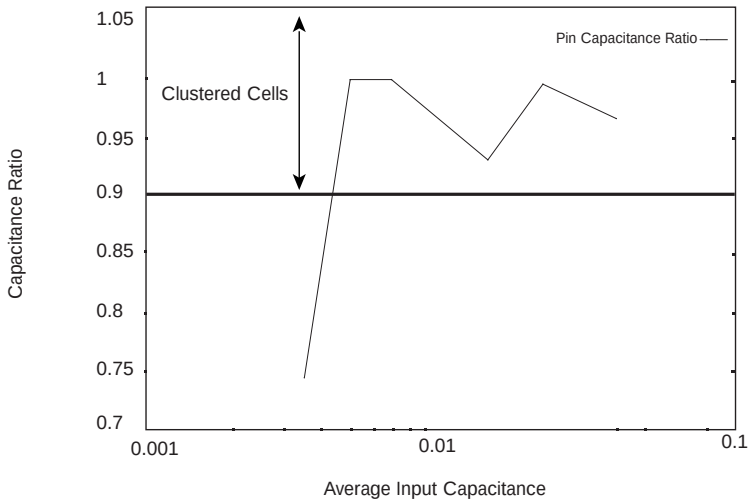

Figure 2: Capacitance Ratio vs. Input Capacitance

Both of the above measures are used to cluster cells from a discrete library. If many clusters are derived for a particular logic function, the new DC Ultra algorithms currently use only the single best cluster where the errors related to the above metrics are minimal and the size of the cluster is above a certain threshold. Cells in all other clusters are ignored during this analysis. However, technology-dependent optimization in Design Compiler considers all discrete cells in the technology library.

- NLDM Generation

Library analysis derives a gain-and-slew-based (load-independent) NLDM for all scalable cells. Different NLDMs are computed for all timing arcs of the scalable cell for both rise and fall delays and for rise and fall transitions. The NLDMs are computed using the following techniques:

- Average delay for all the discrete cells corresponding to the scalable cell is used to generate a NLDM.
- The NLDM model is set identically to a preferred discrete cell within the cluster corresponding to the scalable cell.

- Input Capacitance Model Generation

To account for variances in input pin capacitances between different input pins of a cell, library analysis derives, for each input pin, a model for input pin capacitance as a linear function of the scaling size (scaling size of a library cell is defined as the average input capacitance over all the inputs of the gate). The scaling sizes and input pin capacitances of all discrete cells corresponding to the scalable cell are used to derive this model. This model tends to be fairly accurate if the clustering has been done correctly. Different models are generated for different input pins as they can be significantly different for input pins of complex gates. For example, the select and data lines of a multiplexor have different input capacitance models.

- Summary of Library Analysis

DC Ultra's library analysis creates a scalable technology library (that is internal to Design Compiler) by analyzing discrete cells from the target technology library. The scalable cells in the scalable library are characterized with a new load-independent delay model, in which delay is modeled as a non-linear function of gain and slew. In addition, library analysis also creates models for area and input capacitance of scalable cells. The analysis algorithms use the scalable cells as a basis for optimization. At the end of optimization, the algorithms discretize the scalable cells to the closest discrete cells available in the target technology library. Following this, Design Compiler performs technology-dependent optimizations primarily for delay optimization, design rule fixing and area recovery.

The accuracy of the algorithms depends upon the accuracy of the scalable model derived by analysis. After performing library analysis, a library evaluation step determines if the target technology library is suitable for optimization.

Library Evaluation

Library evaluation uses several metrics to focus on how the scalable cells are derived in the first place and how they are used by the new DC Ultra algorithms. These metrics also provide a reasonable understanding of which discrete cell characteristics would qualify the elements as suitable for optimization.

Library evaluation only considers a subset of the combinational elements that were analyzed during library analysis. Library evaluation is based mostly on three main metrics:
- Function classes
- Number of sizes
- Size consistency metrics

- Function Classes

Library evaluation considers only a subset of the combinational cells that were analyzed during library analysis. These are categorized into two sets: basic and extended. All cells that do not belong to these two categories are ignored during library evaluation. However, all scalable cells and all discrete cells in the technology library are considered during technology dependent optimization. The cell sets are:
- Basic set of cells:
  - Inverters
  - NAND (2 and 3 inputs)
  - NOR (2 and 3 inputs)
- Extended set of cells:
  - AND (2 and 3 inputs)
  - OR (2 and 3 inputs)
  - XOR (2 and 3 inputs)
  - XNOR (2 and 3 inputs)
  - MUX (2 and 3 inputs)
  - IMUX (2 and 3 inputs)
  - AOI
  - OAI

The complete set of cells used in library evaluation is a combination of the basic and the extended set of cells. Even though buffers have not been included in any of the sets above, they are used by the technology-dependent optimization engine. The only reason why library evaluation does not consider buffers is that they can be derived easily using inverters, as long as the library has a good subset of these elements. The library designer is encouraged to add buffers as they will help in reducing cell count and area during technology-dependent optimization.

- Number of Sizes

Library analysis creates clusters of discrete cells for each single-output combinational logic function in the target technology library. Only the best cluster for each logic function is retained for the optimization algorithms. In order for the library to qualify for optimization, the size of the clusters must be reasonably large.

For library evaluation, the number of sizes of a logic function is the number of discrete cells in the cluster associated with the scalable cell implementing that function. The recommended number of sizes for the logic functions in the basic and extended sets is as follows:

- Basic set of cells:
  There should be an average of at least six discrete cells corresponding to all scalable cells in the basic set. Preferably, the number of sizes of each logic function in the basic set should be greater than six.
- Extended set of cells:
  There should be an average of at least five discrete cells corresponding to all scalable cells in the extended set. Preferably, the size of each logic function in the extended set should be greater than five.

It is important to note that simply having the number of recommended discrete sizes for a particular logic function in the technology library is not sufficient. Since the evaluation checks the number of sizes in a cluster, it is important for the discrete cells to have similar characteristics for them to be included in the same cluster.

- Size Consistency

Some of the metrics that help evaluate whether the different discrete sizes in a logic cluster are well correlated, such that good scalable models (with small errors) can be generated are: input pin consistency and delay consistency. Every scalable cell and all discrete cells in its logic cluster have the same number of input and out-put pins. The recommended ratio of capacitances of corresponding input pins of consecutive sizes be within 10 percent. This input pin consistency will ensure uniform scaling of the discrete sizes. Also the variance of capacitances between different functionally equivalent input pins of a discrete cell should be within 10 percent. This will ensure that the scaling size approximation that is used in library analysis will be less erroneous. Adherence to these metrics is likely to result in rich clusters and good linear plots during library analysis.

In order to check for delay consistency, there is a `delay_max_error` metric defined for each scalable cell. `delay_max_error` is the maximum delay error between the NLDM table of the scalable cell derived by library analysis and the NLDM table of all the corresponding discrete cells. This metric is based on the maximum overall delay arcs and both rise and fall NLDM tables. The error is represented as a percentage and is only computed for gain and input slew values within a permissible range. The average `delay_max_error` is the average of all the max delay errors for all the scalable cells in the complete set of cells.

The value of average `delay_max_error` should be as low as possible. For rich libraries, the acceptable value of average `delay_max_error` is increased. The maximum permissible value of average `delay_max_error` is 15 percent.

Library evaluation also computes the average `delay_max_error` for cells from the basic set alone. This is expected to be significantly lower than that for the complete set of cells. The maximum permissible value for average `delay_max_error` of cells in the basic set is 10 percent.

Library Developers Guidelines

The previous sections described the library analysis and library evaluation techniques used by the DC Ultra algorithms. Library developers should use these techniques and metrics as guidelines for designing libraries that are suitable for the new optimization algorithm. There are other guidelines that can be used to take advantage of the QoR capabilities in Design Compiler. Like library analysis, these guidelines cover only single output combination cells, not sequential cells or multibit cells.

The basic guidelines are:

- There should be at least six cells for each logic function in the basic set of cells.
- There should be at least five cells for each logic function in the extended set of cells.
- Sizes derived using internal buffering will most likely get ruled out during logic clustering as they typically have poor correlation with other non-buffered cells. For example, in Figure 1, most of the cells not included in the cluster have been derived by internal buffering.
- For each discrete cell in the complete set of cells, the variance of capacitances between different input pins of the cell should be within 10 percent. This will ensure that the scaling size approximation that is used in library analysis will be less erroneous.

  The following are the input pin capacitances for three different two-input NAND gates in a library. Cells NAND2 and NAND3 have almost equal input pin capacitance for the two input pins. However, cell NAND1 has significantly different input pin capacitance for the two input pins and the ratio is below 0.9. This cell will not be included within the cluster because of inconsistent input pin capacitances.

Cell	Pin1	Pin2	Ratio
nand1	0.00410	0.00365	0.89
nand2	0.03360	0.03340	0.99
nand3	0.05630	0.05600	0.99

- For corresponding timing arcs of discrete sizes for a particular logic function, if the ratio of the output load to average input capacitance is equal, then the delays of the arcs should be within 10 percent. In the following example, output capacitance is set to three times the input capacitance for a set of three different discrete inverters. Cells inv2 and inv3 have consistent delay arcs and hence might be clustered together. Cell inv1 will be ignored as its rise and fall delays are significantly different from those of the other cells.

Cell	Input Cap.	Output Load	Rise Delay	Fall Delay
inv1	0.0140	0.0420	0.2545	0.3325
inv2	0.0440	0.1320	0.0978	0.0734
inv3	0.0870	0.2610	0.1016	0.0783

- The sizes of cells for each logic function preferably should be geometrically distributed. The scaling factor between consecutive sizes should be less than two, though a smaller scaling factor is desirable. In addition, a hybrid size distribution with linear distribution for small to medium sized cells, and geometric distribution for medium to large size cells is recommended. Such a distribution of sizes will help Design Compiler produce better results. However, the size distribution criterion is not imposed as a metric during clustering or library evaluation.
- For each discrete cell within a logic cluster, it's desirable to have an output maximum capacitance constraint that is linearly proportional to the average input capacitance of the discrete cell.

Current versions of technology libraries might have logic cells that do not meet the above criteria. The developer does not have to remove such cells. Clustering will ignore these cells as long as there are sufficient cells that do meet the above requirements. Developers can also add nonconformant cells for optimization reasons other than delay cost. It is important to note that these cells will be used during technology-dependent optimization and can significantly improve the target cost (design rules or area) that the cell has been designed for.

- Example Library Analysis

The new DC Ultra library analysis and evaluation algorithms were tested with several vendor libraries. Most of the newer libraries were found to be suitable for the new optimization algoritm. However, those libraries with very few sizes were not found suitable. Below are the results of these tests that can act as a guideline for library developers who want to perform similar evaluations.

The details of the columns in the following tables are as follows:

- Num Sizes - This column represents the number of cells in the selected cluster.
- Size Consistency Max Error - Within each cluster the cells are sorted based on their scaling size. Next, size consistency error is computed for each cell by comparing its NLDM to the NLDM of the cell succeeding the current cell after normalizing the NLDM tables based on the scaling size of the two cells. This table represents the maximum of all the size consistency errors computed for all the cells in the cluster.
- Delay Max Error - The maximum delay error between the NLDM table of the scalable cell derived by library analysis and the NLDM table of all the corresponding discrete cells. This metric is based on the maximum overall delay arcs and both rise and fall NLDM tables. The error is represented as a percentage.

Cell	Num Sizes	Size Consistency Max Error	Delay Max Error
and2	7	8.10%	7.80%
aoi4	10	5.90%	7.40%
oai4	9	5.00%	4.90%
inv	21	15.10%	15.50%
imux2	8	10.30%	14.50%
nand2	24	11.00%	14.00%
nand3	19	11.70%	13.00%
nor2	28	17.60%	18.40%
nor3	26	9.00%	13.40%
or2	7	4.70%	5.50%
xnor2	9	6.50%	6.30%
xor2	10	5.90%	10.40%
Average	14.8	9.20%	10.90%

Table 1: Library 1 Results

Table 1 was generated during library evaluation. The library was accepted as a good library because it had many sizes for both basic and extended gates. The error numbers were also within acceptable ranges.

Cell	Num Sizes	Size Consistency Max Error	Delay Max Error
and2	3	9.9%	14.7%
aoi4	4	8.8%	7.0%
inv	15	9.7%	18.8%
imux2	2	6.7%	8.8%
mux2	3	12.9%	13.1%
nand2	9	6.5%	4.5%
nand3	5	6.5%	5.1%
nor2	5	7.2%	9.6%
nor3	3	8.2%	2.9%
oai4	4	8.7%	8.5%
or2	3	12.2%	13.0%
xnor2	2	9.8%	6.7%
xor2	2	10.0%	7.9%
Average	4.6	9.0%	9.3%

Table 2: Library 2 Results

Even though Library 2 has average sizes less than five, it was accepted because it has several good properties. For example, it has many sizes for the basic set of gates and the error numbers are fairly low.

Cell	Num Sizes	Size Consistency Max Error	Delay Max Error
xnor2	2	7.4%	16.5%
inv	5	6.4%	4.5%
nor2	2	11.3%	9.7%
nor3	3	9.9%	13.4%
nand2	3	9.0%	4.3%
nand3	3	10.4%	7.3%
aoi4	2	10.4%	9.6%
oai4	2	10.2%	10.0%
xor2	2	1.7%	16.5%
Average	2.7	8.5%	10.2%

Table 3: Library 3 Results

Library 3 was not found suitable for optimization, because it had too few sizes for both the basic and extended gates, even though size consistency and delay max errors were reasonably low.

Cell	Num Sizes	Size Consistency Max Error	Delay Max Error
and2	2	10.7%	16.4%
inv	7	4.4%	5.6%
mux2	2	26.2%	31.1%
nand2	2	6.5%	6.1%
nor2	2	44.0%	41.8%
nor3	2	31.2%	33.3%
or2	2	4.2%	11.7%
Average	2.7	18.2%	20.9%

Table 4: Library 4 Results

The above library in Table 4 was not found suitable, because it has too few sizes and the error numbers are too high.

Using the New DC Ultra Algorithms

These library analysis, evaluation and optization algorithms are part of DC Ultra and are switched off by default. They can be invoked using the two switches:

- `set_ultra_optimization`
- `compile_new_optimization = true`

If library evaluation determines that the library is not suitable for optimization, it will switch back to the standard Ultra optimization engine and will issue the following message:

```
Information: Switching to base optimization engine for the current compile run. (OPT-220)
```

Experimental results indicate that many modern libraries satisfy the optimization guidelines. Similarly, developers can design and tune their libraries to better use QoR capabilities in Design Compiler. These capabilities are only part of DC Ultra's enhanced feature set targeted to high-performance designs. A more detailed explanation of other DC Ultra features can be found at http://www.synopsys.com/products/logic/dc_ultra_ds.html.

SYNOPSYS®

700 East Middlefield Road, Mountain View, CA 94043 T 650 584 5000 www.synopsys.com
For product related training call 1-800-793-3448 or visit the Web at www.synopsys.com/services