Standard-Cell Placement from Functional Descriptions

Klaus Buchenrieder

Waldparkstraße 60, D-8012 Ottobrunn/R. Germany
Received February 15, 1990.

Abstract

This paper presents a functional language for the unambiguous description of digital circuits, a method and algorithms to obtain a standard-cell layout and a comparative evaluation of the developed functional standard-cell placement technique. The presented placement scheme is different from traditional methods because the complete layout geometry is specified and constructed automatically from a functional description. The construction relies on a translation that combines the simplicity of standard-cells with the elegance of functional programming. An evaluation of the method introduced shows that the quality of the resulting placement is close to the results achieved with simulated annealing while the computation time is significantly less. Furthermore, the evaluation suggests to employ the functional placement method in conjunction with low-temperature simulated annealing for running-time reduction and improved results.

1. Introduction

In recent years, the semiconductor industry has had to speed up the development cycle for low-quantity, high performance ASICs. This caused the silicon foundries to introduce elaborate polycell catalogs allowing for design above the geometric of functional level. The standard-cell approach is currently the most popular implementation technique for randomly connected control and “glue” logic. From a users point of view, standard-cells perform commonly used functions of which implementation details are of no concern. Geometrically, standard-cells are fixed-height building blocks that implement circuit components of various complexity, and thus vary in width. Cell interconnects are located on top and bottom of each cell so that all wiring fits in routing channels along cell-rows. These are the connection points to the routing channel along the cell rows. Channels are relieved from many short distance connects through primary in- and outputs that are available at the sides for cell-to-cell result propagation. Connections between neighboring cells are established with overlaid stubs of conductive material.

Complete layouts are composed of several parallel cell-rows separated by routing channels. Since the problem of optimal standard-cell placement is NP-hard, various placement techniques have been developed for accomplishing area-efficient layouts. The placement strategy proposed in this paper is different from existing methods, because it is based on syntax-directed parsing of a high-level expression.

Unlike other methods it is not based on a heuristic, and furthermore a functional specification is used instead of low-level circuit information obtained by schematic capture. Here, circuits are specified with a functional expression that is parsed, optimized and laid out.

An implementation of a standard-cell placement system based on the functional placement principle explained in this paper has shown that the layout quality is comparable to that of simulated annealing, yet requires only $O(n^2)$-time and $O(n)$-space to compute. The developed method constitutes an area-efficient, technology-independent, standard-cell placement technique that allows the translation of functional descriptions into complete layouts. Even though, many definitions for circuit behavior can be devised, behavior in this context is understood merely as
mapping of signals from input-to-output within specified time-constraints. Mappings are performed by functions realized as standard-cells or cell-blocks that fulfill the constraints imposed.

2. The Approach

The goal of the proposed placement scheme is to provide the basis for a library-independent layout tool that is flexible enough to meet various designer needs. An unambiguous translation of the functional circuit description into placement and netlist information is achieved by syntactical and semantical analysis, parse-tree construction, tree-optimization, netlist generation and layout construction. The placement process is constructive and based on recognized compiler techniques. A physical layout is obtained from the netlist and placement information with standard tools inherent to the supporting design environment.

2.1. The Realization Language

In recent years it was shown that functional languages are most appropriate for digital hardware design. Complete functional programs represent the entire logic and corresponding connectivity in a concise and unambiguous way. The grammar given in Table 1, specifies the syntax of the realization language.

Table 1. BNF of the Realization Language

Size and clarity of the grammar are improved through additional symbols included in the metalanguage. Additionally introduced symbols such as \( \text{\{,\}}, \) \( \text{[,...,]} \), do not lead to a mathematically more powerful BNF, but serve as shorthand notation for more elaborate constructs. Square brackets \( \text{[,...,]} \) indicate optional sequences, and curly braces enclose sequences that can occur any number of times (Kleene Star). \( \epsilon \) denotes the null or empty string and terminal symbols are underlined. Another important extension to the grammar grants access to externally defined functions.

This addition is required because the grammar offers only a limited set of inherent functions. Additional functions can be load from a library as rules for the grammar and as standard-cells for the actual implementation on silicon. For each identifier that is not explicitly specified in the grammar, the number of expressions \( num \), is read from the component library. The sequence of metasymbols \( !\cdots!^{num} \) repeats \( \text{expression} \) exactly \( num \) times producing the desired function.

2.2. Analysis and Parse-Tree Construction

In the lexical analysis, or scanning phase, the functional description is read, and macro