The design process, at various levels, is usually evolutionary in nature. It starts with a given set of requirements. Initial design is developed and tested against the requirements. When requirements are not met, the design has to be improved. If such improvement is either not possible or too costly, then a revision of requirements and an impact analysis must be considered. The Y-chart (first introduced by D. Gajski) shown in Fig. 14.3 illustrates a design flow for most logic chips, using design activities on three different axes (domains) which resemble the letter "Y."

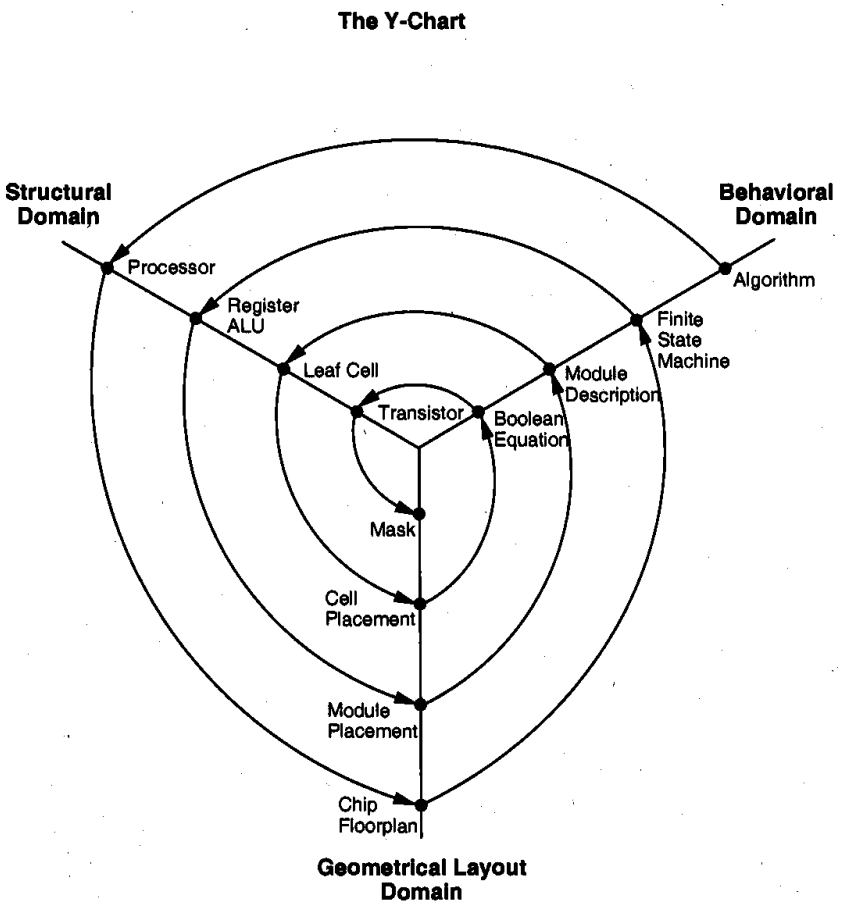

Figure 14.3. Typical VLSI design flow in three domains (Y-chart representation).

The Y-chart consists of three domains of representation, namely (i) behavioral domain, (ii) structural domain, and (iii) geometrical layout domain. The design flow starts from the *algorithm* that describes the behavior of the target chip. The corresponding *architecture of the processor* is first defined. It is mapped onto the chip surface by *floorplanning*. The next design evolution in the behavioral domain defines *finite state machines* (FSMs) which are structurally implemented with *functional modules* such as registers and arithmetic logic units (ALUs). These modules are then geometrically placed

onto the chip surface using CAD tools for automatic *module placement* followed by routing, with a goal of minimizing the interconnects area and signal delays. The third evolution starts with a behavioral *module description*. Individual modules are then implemented with *leaf cells*. At this stage the chip is described in terms of logic gates (leaf cells), which can be placed and interconnected by using a *cell placement and routing* program. The last evolution involves a detailed *Boolean description* of leaf cells followed by a transistor level implementation of leaf cells and *mask generation*. In the standard-cell based design style, leaf cells are pre-designed (at the transistor level) and stored in a library for logic implementation, effectively eliminating the need for the transistor level design.

Figure 14.4 provides a more simplified view of the VLSI design flow, taking into account the various representations, or abstractions of design: behavioral, logic, circuit and mask layout. Note that the *verification* of design plays a very important role in every step during this process. The failure to properly verify a design in its early phases typically causes significant and expensive re-design at a later stage, which ultimately increases the time-to-market.

Although the design process has been described in linear fashion for simplicity, in reality there are many iterations, especially between any two neighboring steps, and occasionally even remotely separated pairs. Although top-down design flow provides an excellent design process control, in reality, there is no truly uni-directional top-down design flow. Both *top-down* and *bottom-up* approaches have to be combined for a successful design. For instance, if a chip designer defines an architecture without close estimation of the corresponding chip area, then it is very likely that the resulting chip layout exceeds the area limit of the available technology. In such a case, in order to fit the architecture into the allowable chip area, some functions may have to be removed and the design process must be repeated. Such changes may require significant modification of the original requirements. Thus, it is very important to feed forward low-level information to higher levels (bottom-up) as early as possible.

In the following, we will examine design methodologies and structured approaches which have been developed over the years to deal with both complex hardware and software projects. Regardless of the actual size of the project, the basic principles of structured design will improve the prospects of success. Some of the classical techniques for reducing the complexity of IC design are: Hierarchy, regularity, modularity and locality.

14.3. Design Hierarchy

The use of the hierarchy, or "divide and conquer" technique involves dividing a module into sub-modules and then repeating this operation on the sub-modules until the complexity of the smaller parts becomes manageable. This approach is very similar to software development wherein large programs are split into smaller and smaller sections until simple subroutines, with well-defined functions and interfaces, can be written. In Section 14.2, we have seen that the design of a VLSI chip can be represented in three domains. Correspondingly, a hierarchy structure can be described in each domain separately. However, it is important for the simplicity of design that the hierarchies in different domains be mapped into each other easily.