C H A P T E R 5

INTEGRATION OF TEST PLANNING SYSTEM

This chapter describes the integration of the modules of the test planning system and the Test Planning Graphical User Interface.

5.1 System Integration

The development of the test planning system involves the use of various design tools and programming languages, and creation of files of different formats. For instance, the specification repository is developed using SGE [19], the simulation models of the test bench primitives are created in VHDL using SPW [18] and SGE, the requirements

interface and the goal tree engine are designed in C, the goal tree editor and the graphical user interfaces are implemented in C and X Window libraries of different layers, the goal tree data files are stored in a binary format, VHDL simulation control files are created in Synopsys script language, the requirements extraction tool is developed in C shell language, the VHDL code is analyzed and simulated using Synopsys tools, and the output data are displayed using Matlab toolboxes [62]. It is required that all these modules be integrated to form an a seamless and automated test planning system.

A C program has been developed to integrate the modules of the test planning system. This software performs data format conversion required from one module to another, and provides a single platform for the control and execution of the tasks of test planning, test bench generation, model simulation, database management, etc. Figure 5.1 depicts the integrated test planning system. The use of the integrated test planning system for these tasks is twofold. First, the test planning system works as an interactive query system where the goal tree system accepts from the user a query in the form of a test goal and commands that the test bench generator create and execute a set of test benches to answer the query. Second, the test planning system works as an automated batch system where the whole test plan is exercised and the model under test validated [48].

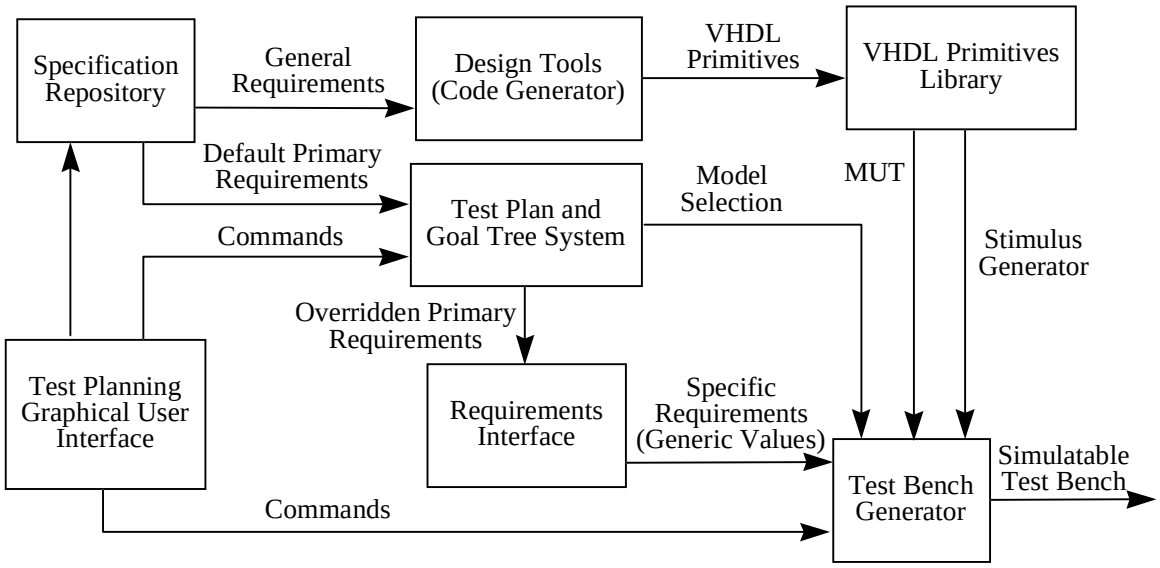

Figure 5.1 Integrated Test Planning System

5.2 Test Planning Graphical User Interface

The Test Planning Graphical User Interface (TPGUI) has been developed in C using XView layer libraries of the X-11 protocol [63]. It provides the user with a friendly environment to operate the test planning system for the tasks of test planning, test bench generation, model simulation, and results display. It fully utilizes the three types of test bench configurations, i.e., off-line, online nonadaptive, and online adaptive as described in Section 2.2.

The Test Planning Graphical User Interface is a menu-driven system with a set of menus and input fields showing up one by one per the selections made by the user. The

user is thus prompted and led all the way through the test configuration session. Figure 5.2 shows the flow of menus provided by TPGUI. The menus are described as follows.

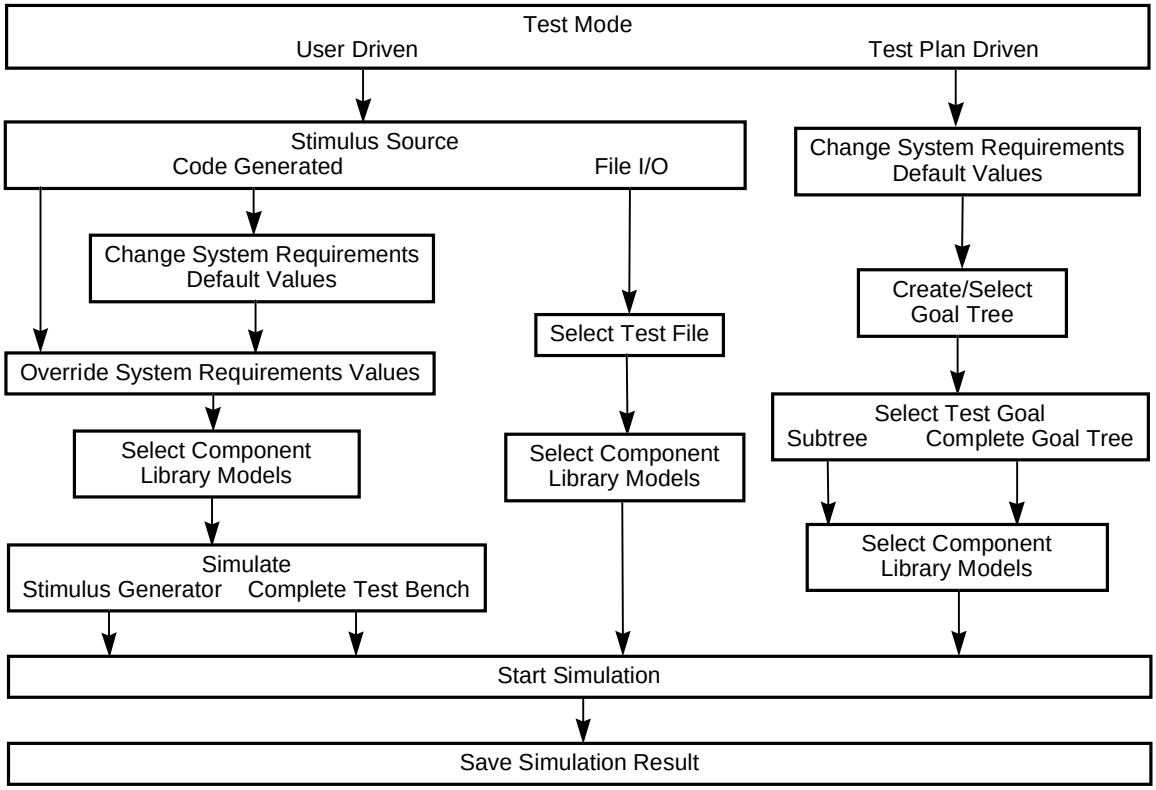

Figure 5.2 Menu Structure of TPGUI

Test mode: The test mode leads to the two major paths of the test bench generator: the user driven mode and the test plan driven mode. The user driven mode allows the user to set up and simulate one single test at one time. In the test plan driven mode, the goal tree editor pops up where the user can create or load a goal tree. Test benches are configured iteratively per the goal tree and all tests are executed in a batch.

Both the stimulus generator and the model under test of the test bench are simulated. This is an example of the online adaptive test bench configuration type since the test results are fed back to the goal tree system and thus affect the configurations of the following tests.

Stimulus source: The menu allows the user to select where the stimulus comes from. There are two stimulus source options: code generated and file I/O which lead to the two branches under the user driven test mode. The code generated option uses the online nonadaptive test bench configuration type and allows the test vectors to be generated during simulation. Instead of generating test vectors during simulation, the file I/O option employs the off-line test bench configuration type where the test bench retrieves test vectors from the test vector database through file I/O and drives the MUT with them.

Change system requirements default values: This button shows up in both the code generated option of the user driven mode and the test plan driven mode. It invokes the specification repository from which the user can change the default values of the system requirements, specifically, the primary requirements. Once the default values are settled, this button is seldom used during normal testing session unless there is a need to change the specification.

Override system requirements values: This menu allows the user to override the system requirements for the current test. These overridden system requirements and those

unmodified system requirements which accept default values from the specification repository compose the primary requirements and are sent to the requirements interface.

Select component library models: This button is available for both user driven mode and test plan driven mode. It invokes a menu from which the user has the option to change library models from the VHDL primitives library and have them bound to the instantiated components of the test bench. If the user does not press this button, the default library models are used.

Simulate menu: This menu is available in the code generated option of the user driven mode. The user can choose to either simulate the stimulus generator only or simulate the complete test bench. If the user chooses to simulate the stimulus generator only, the stimulus generator is enabled while the MUT and the comparator are disabled. After simulation, the generated test vectors can be saved to the test vector database. They can be reused in the file I/O option of the user driven mode, or in the test plan through the file I/O strategy. If the user selects to simulate the complete test bench, the test vectors are generated on-line and are sent to the MUT for processing in the same simulation.

Select test file: This menu is only available for the file I/O option of the user driven mode. It allows the user to select a particular test file from the test vector database.

Create/Select goal tree: This button is available with the test plan driven mode. It invokes the goal tree editor from which the user can either create a new goal tree, accept or modify an existing goal tree from the goal tree library.

Select test goal: This menu allows the user to choose which goal(s) to be tested. If the subtree option is selected, the subtree rooted at the current node is traversed and tested. If the complete goal tree option is chosen, the complete goal tree is traversed. This menu is available when the test plan driven mode is chosen.

Figure 5.3 depicts the base window of TPGUI under the code generated option of the user driven mode. It shows how the system requirements are specified. The values can be entered either by typing or by using the slide bars. Once the adjustable requirements are specified, the derived requirements are computed accordingly by the requirements interface, and the changes are reflected immediately on the base window.

Figure 5.4 gives the base window of TPGUI under the file I/O option of the user driven mode, which shows the selection of a test file from the test vector database. When navigating through the list of test files, the primary and derived system requirements associated with the current test file are automatically displayed to help the user understand what is being simulated.

Figure 5.5 shows the window under the test plan driven mode, which displays the simulation results in the text pane in the right half of the window. Figure 5.6 shows the pop-up window for library model selection.

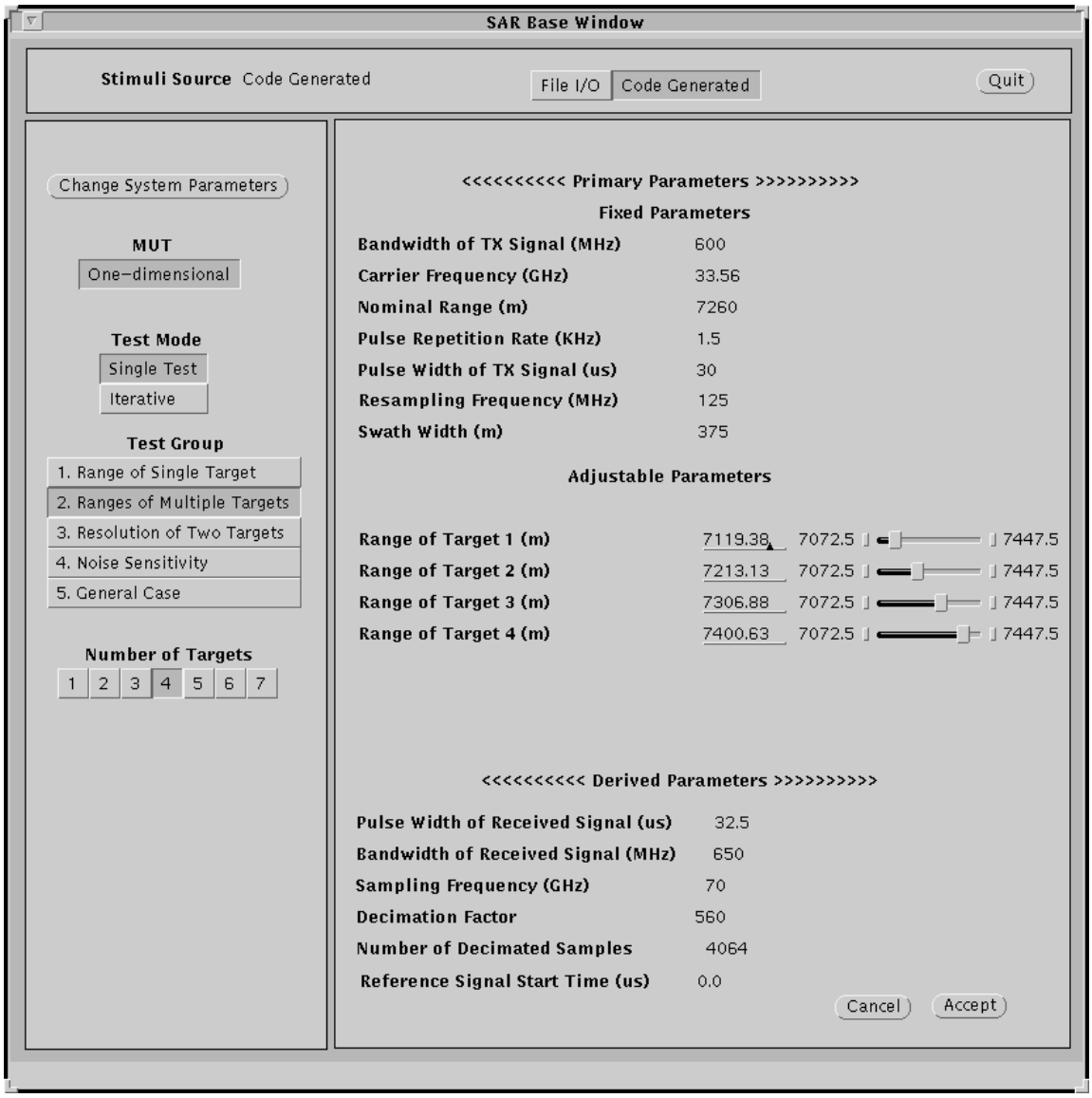

Figure 5.3 TPGUI Showing Code Generated Option of User Driven Mode

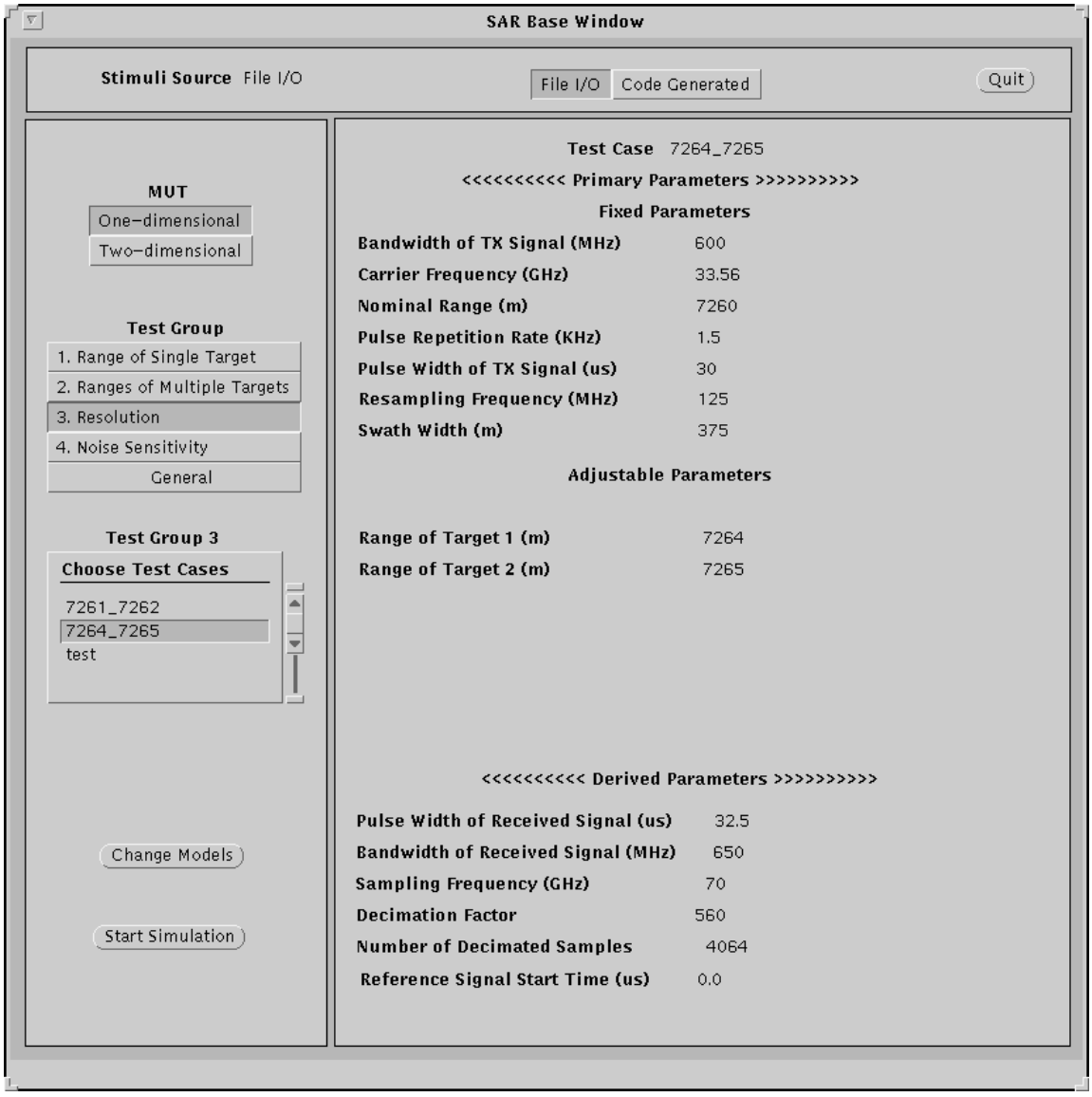

Figure 5.4 TPGUI Showing File I/O Option of User Driven Mode

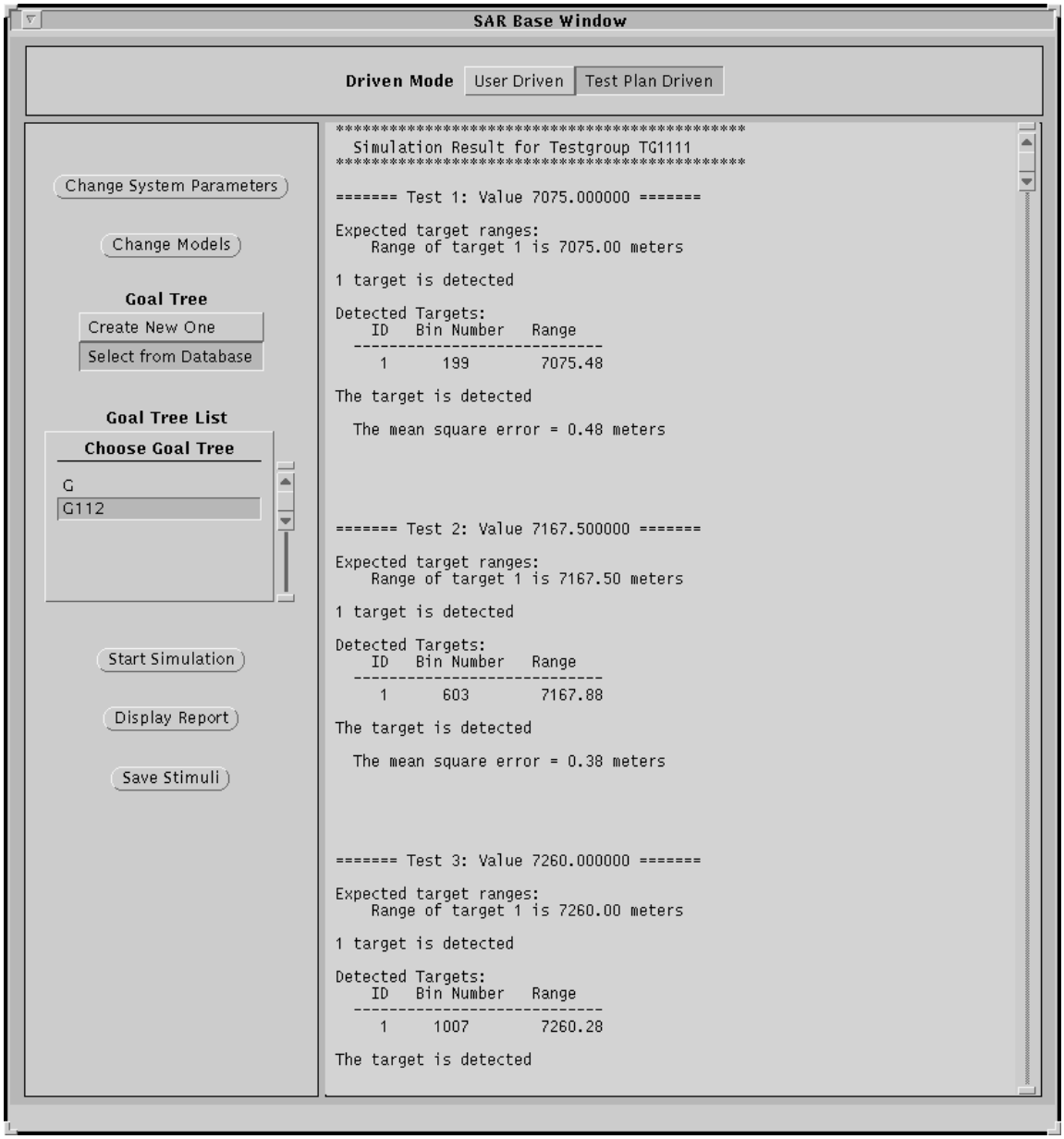

Figure 5.5 TPGUI under Test Plan Driven Mode

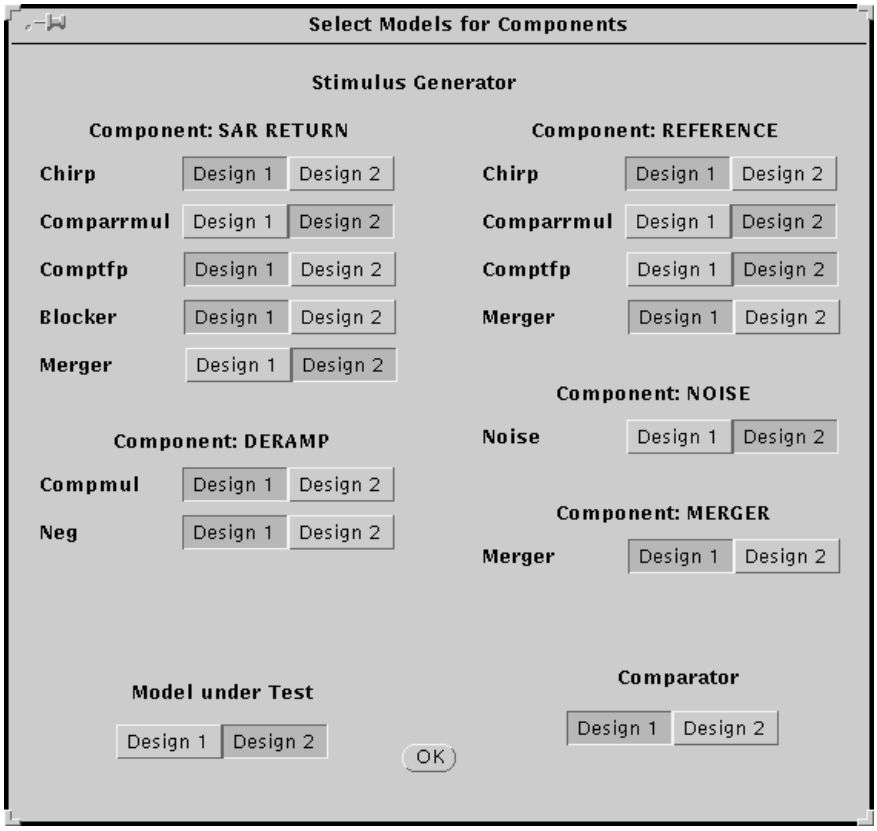

Figure 5.6 Window for Library Model Selection