A graphical user interface (GUI) which is used to describe chip design and verification infrastructure is disclosed in this document. The GUI is used to capture the following chip design and verification infrastructure specification, check the specification, and generate the following elements:

- Design hierarchy and interconnect
- Test benches
- Instruction set architectures (ISA)
- Constants
- Networks on chip
- Components in functional units
    - Registers/register files
    - FIFO's
    - Memory maps

Design hierarchy and interconnect

The GUI is used to enter the design hierarchy and interconnect specification using the following steps.

1. Create the design hierarchy using hierarchical block diagrams or a tree like structure.

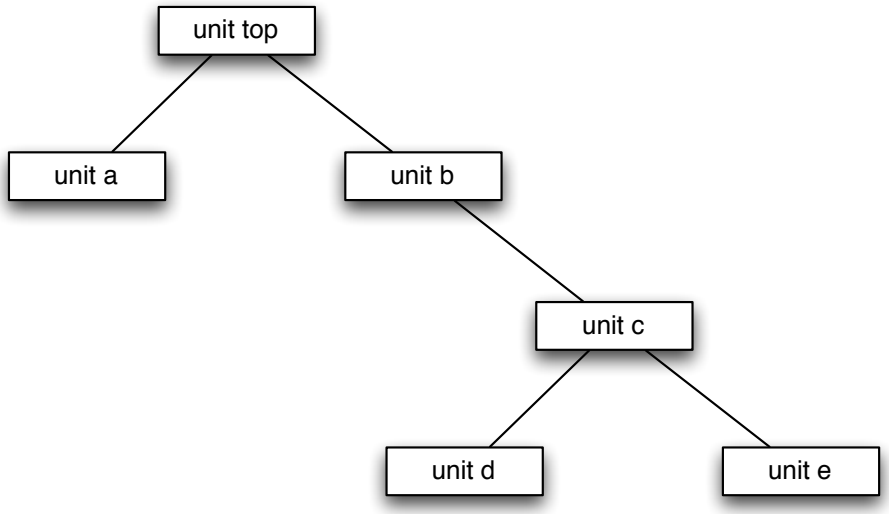

Figure 1. Design hierarchy tree

In figure 1 there are three levels of hierarchy. The first level is unit top. Unit top contains unit a and unit b (level 1). Unit b contains unit c (level 2). Unit c contains units d and e (level 3).

2.Create the connections between the modules using either the Fastpath Logic GUI (e.g. the blue lines in figure 2.) or using the Fastpath Logic spreadsheet entry system.

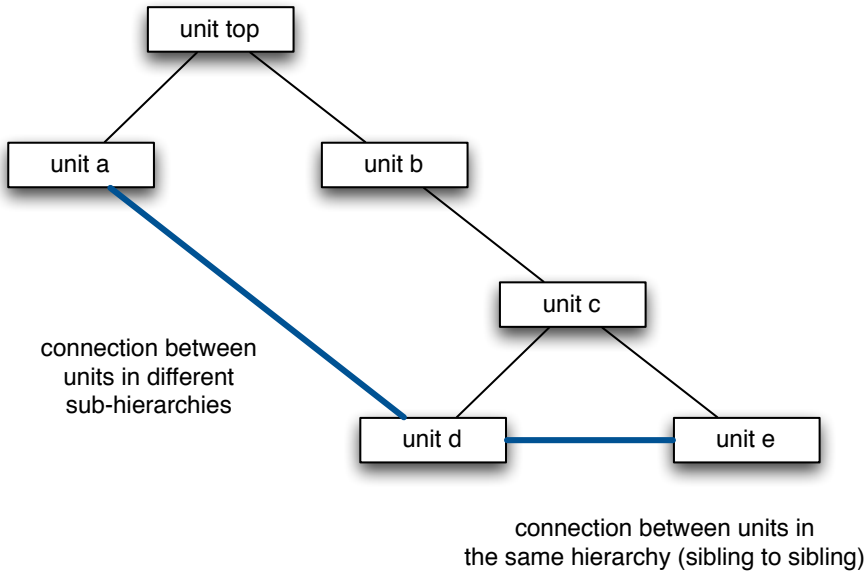

Figure 2. Point to point connections between modules

Figure 2 shows connections between different units in the design hierarchy.

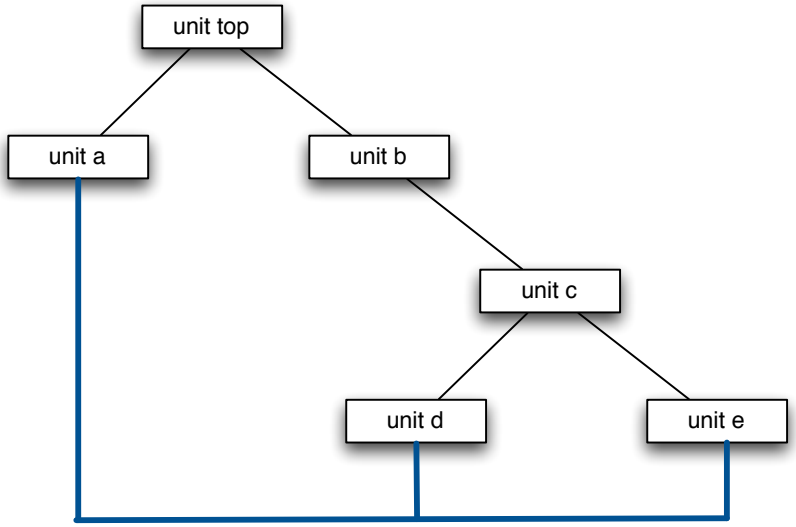

Figure 3. Connections between more than two modules

Figure 3 shows the connections between three modules in the design hierarchy

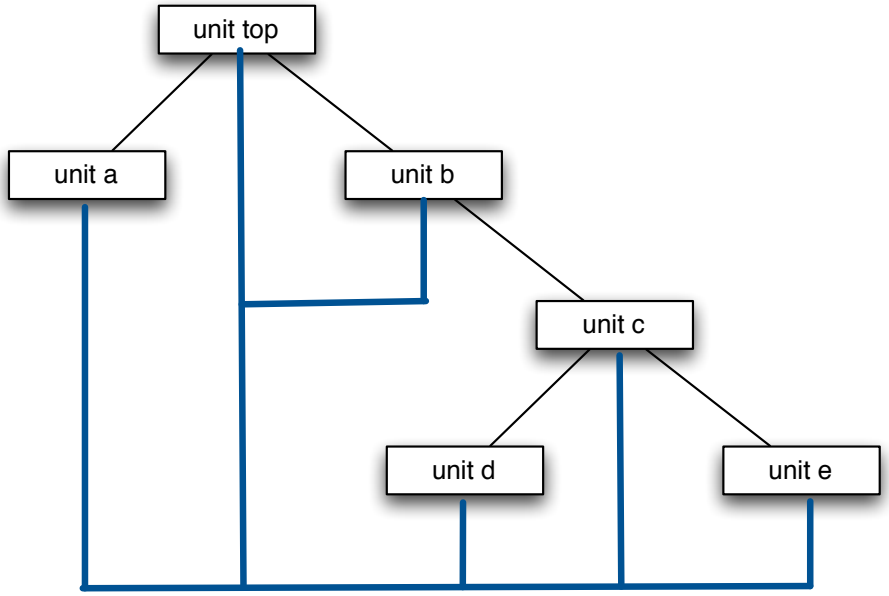

Figure 4 shows a signal such as a clock or reset which is connected to all modules in a sub-hierarchy.

A signal can be added to all units in a sub-hierarchy by right clicking on the top unit in the sub-hierarchy and then selecting add_connection_to_all_units. This will add the specified connection to all units in the sub-hierarchy.

Clicking on the connection between the units should allow the user to select an existing interface or define a new interface that will be automatically added to all endpoints of the connections.

Creating an interface library
There is a create_interface menu item which the user selects in the GUI. A table or dialog box appears which the user then enters individual ports or selects other pre-existing interfaces to add to the current interface being defined. The interface being defined is then saved for use in connections in the design or in other interfaces. Existing interfaces can be viewed and edited in the GUI. Alternatively the interface library can be created in a design language (e.g. C++, Verilog, etc.) and loaded into the GUI.

Adding connections to a design
An example is described below.

1. Select the interface library(s) to load into the GUI.
2. A design hierarchy is entered into the GUI or read in from a file.
3. The user selects an add_new_connection (add new connection) menu option in the GUI.
    a. A spreadsheet interface appears on the screen with the following columns
        i. Driver unit(s) -the name(s) of the unit(s) the signal is driven by

ii. Receiver unit(s) - the name(s) of the unit(s) the signal is received by
iii. connection object type - the connection object type inserted
iv. actual name-the name(s)/prefix name(s) of the connections

driver unit/ instance(s)	receiver unit/ instance(s)	connectivity object type	actual name
unit a	unit d	ad_ifc	a_d
unit d	unit e	de_ifc	d_e

Figure 5. Example spreadsheet entry

Figure 5 Shows an example interconnect spreadsheet entry form. After the design hierarchy and the connection library is entered using the GUI the user then selects the unit instance from a pull down menu in the driver and receiver unit instance cells. More than one unit instance can be a driver or receiver depending on the connection type. The user then selects the connection type from the connectivity object type. The connectivity object type can be a basic type such as a port or a signal or an interface from the connectivity library. If the user selects a basic type then the user will have to provide the specification for the basic type (e.g. port direction and width). The user then selects the actual name for the connectivity object. If the connectivity type is a signal group of an interface actual name is prefixed to the names in the connectivity object (see the Fastpath Logic interconnect application).

Connecting endpoints
After entering the design hierarchy, interfaces, and connections the user then calls the auto router from within the GUI which processes each connection. The connection type (e.g. signal, signal groups, interface, port) are automatically added to the endpoint units and if present to the intermediate modules. The ports in the unit instances are then connected by the auto router using the actual name which is either either the name of the port/signal or as a prefix for the ports/signal in the interfaces/signal groups. The resulting design hierarchy and connections can then be viewed in the GUI. Figure 5, row 1 shows a connection from unit a to unit d. The connections can be modified as necessary.

Adding elements
Once the design hierarchy has been created the user can add other elements such as registers, register files, FIFO's and other objects to the units using the GUI. The user can right click on the state element, set the addressable object type to true (add the object to the memory map), and then optionally set the objects address in the property dialog box.

A chip design contains objects which can be assigned an address. The address is used by software and hardware to write/read the objects. A user can optionally set the address or unit id for each unit in the design. Alternatively the user can instruct the GUI/

compiler to call the auto mapper which will automatically assign the ID's to the units. A symbolic name is used for the unit ID which the users can incorporate in their code. The addresses can be viewed by invoking the view address map menu option in the GUI. These addresses can be viewed by invoking the view address map menu option in the GUI.

ISA GUI

A GUI can be built that the users enters the ISA instruction format tree. The GUI has an entry mode which facilitates building the tree by placing instruction formats and fields within those instruction formats and associating enumerated types with the instruction fields much like the tree shown in figure 2. In addition the GUI checks that the user only selects opcodes from fields that have a parent set of opcodes. The GUI tracks the opcodes that are selected in the descendants of the instruction format that contains the opcode set. Only one opcode may be selected by one instruction format in the set of off spring. In other words the selection of the individual opcodes and association of the individual opcodes with a specific off spring instruction format is mutually exclusive.

The following ISA objects can be drawn in the GUI.

1. The user creates a root instruction format field by selecting the "add new instruction format" menu item. The type of the instruction format will be "root format". The user then adds the following information:
    a. Name of the new instruction format.
    b. Instruction format type.
    c. Width of the instruction format.

    Once the root instruction format is created the add_root_instruction is grayed out in the menu.
2. After the root format is created, the user can right click on any visible instruction format (including the root instruction format) and selects the "create child instruction format" menu item. A new instruction format appears in the GUI which inherits the parent format information (width and existing fields) and a connection is shown from the parent to the child instruction format. The user then adds the following information:
    a. Name of the new instruction format.
    b. Optionally sets the instruction format type.
    c. Optionally adds new fields to the instruction format in unused positions by right clicking on the instruction format which brings up a dialog box.
        i. The user names those fields, and sets the width of the fields in the field specification dialog box.
        ii. Optionally the user sets the field type in the dialog box.
3. Adding enumerated types to fields and selecting enumerated item in fields
    a. The user can right click on a field to add an enumerated type if the field is either:
        i. defined in the current instruction format.
        ii. inherited from an ancestor instruction format and the field does not already have an associated enumerated type in any of the ancestor instruction formats.

b. If the field already has an enumerated type associated with it in an ancestor format, the user can right click on a field and select an enumerated item value to associate with it.

4. The user can set the position of the fields within the instruction format where the field is added by right clicking on the instruction format using one of two different methods.
   a. A dialog box with the list of fields in the instruction format appears. The user can set the upper and lower field position index (Fields can have their position specified in the instruction format set either using absolute indices) using the field position table. Note that the fields need to have contiguous positions.
   b. Alternatively the user can change the field width and position within the instruction format by dragging the left or right edge of a field. When a field edge position changes, the index value of the field changes automatically above the field in the instruction format.
   c. The GUI prevents fields dragged a field edge into an overlapping position with another field in the instruction format.
5. GUI checker rules (not exhaustive)
   a. Field widths and types can only be modified in the instruction format in which the field was added.
   b. The GUI does not allow the modification of the width, position or name of a field inherited from a parent in an off spring format.
   c. When a field is created in a parent instruction format, the field will appear in all of the offspring of the parent instruction format.
   d. Any conflicts (e.g. overlapping fields) between fields in the same instruction format or parent off spring instruction formats is shown by highlighting the overlapping fields.
   e. The overlapping fields are then corrected by the user by moving the offending fields to the left or right. This mechanism provides instant feedback to the user when there are problems in the inheritance of fields or overlapping fields in the same instruction. The user is able to immediately correct the problems in the GUI.
6. Visual display of instruction formats and fields
   a. The enumerated type or enumerated item value associated with a field is shown either in or next to the associated field.
   b. The upper and lower indices of each instruction format are shown next to the instruction format
   c. The upper and lower indices of the fields are shown next to the field
   d. The unused and unused bits of the instruction format are shown in the instruction format.
   e. The GUI may contain a pipeline drawing which allows the user to specify which signals are used in specific pipes.
   f. Lines linking instruction formats are used to construct the instruction format tree.

Compiling the ISA

In addition as the ISA is constructed the user or an automatic mechanism can compile the ISA format into various formats. The compilation will check the ISA for correctness and report any errors to the user.

ISA GUI Advantages

The advantage of describing an ISA via a GUI or context sensitive editor is as follows:

•The input to each instruction format can be checked for correctness with respect to the other instruction formats in the tree using a list of rules and errors can be flagged visually.
•The ISA designer can see the relationships between the different parts of the tree.
•The ISA designer can drag and drop various parts of the ISA instruction format tree.
•The GUI will force the ISA designer to select valid values at the lower levels of the tree (e.g. a field with an enumerated type is inherited by lower level instruction formats which can only use the value in the upper level enumerated type once).
•The GUI representation is more compact than the code representation.
•Information only needs to be entered once using the GUI as opposed to a hand written specification which requires that information be entered in parallel in different classes and the data is not checked.
•Ease of use.

GUI Interactive Checker

The user's GUI input which represents the design and verification specification is checked by the GUI interactive checker. The checker will generate warnings or errors in certain cases and show the result within the GUI enabling easy debug. Examples of warnings are - In a 20 bit bus, if only 16 of those bits connected, it will generate a warning indicating that 4 of the bits remain unconnected. Examples of errors - if a connection is created with the name "x" and the user attempts to create another connection with the name "x" in the same scope the new "x" will be highlighted in red. If a connection is created between two signals that are of different interface types, it will generate an error and highlight the specific connection within the GUI. In some cases the errors and warnings will be flagged interactively as the user is entering the input specfication.

ISA tree of instruction formats

ISA's are described using a hierarchy of instruction formats. The hierarchy is effectively a tree of instruction formats-see figure 2. Parent instruction formats are inherited by the off spring instruction formats. Off Spring instruction formats add information to the parent instruction formats.

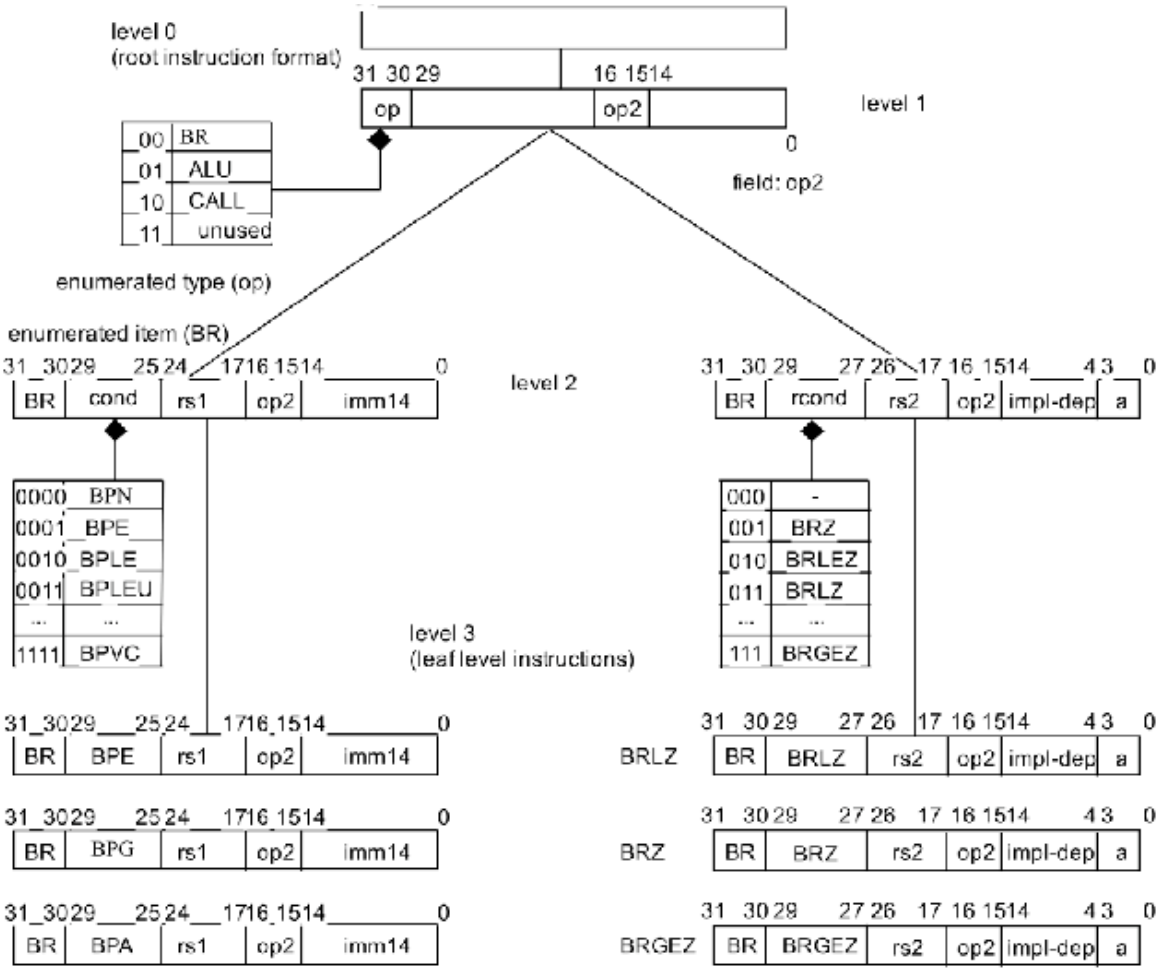

Figure 2. Instruction format tree

Each node in the tree is an instruction format which contains fields and optionally associated enumerated types. The leaf level nodes in the tree have a single major opcode (left hand) associated with the instruction format. The tree may have intermediate levels which add fields and associate enumerated items with fields.

There are three types of nodes in the tree. The root which sets the width of the instruction format and the left most opcode enumerated type, the intermediate level instruction formats, and the leaf level instruction formats.

Visualizing the Design Specification

After the design is specification is compiled, checked, and the infrastructure elements are generated the user can view the design in the GUI. Different views of the design include: the unit instance block diagram, the interconnect schematic, the table view can show memory maps

Networks on chip

The GUI shows schematic, table, and other views of different networks in a design.
•DRAM controller and DRAM client network

•Testability network- JTAG, BIST, MBIST, IO loopback, test clock bypass.
•On chip communication network generated from the memory map and addressable state elements (e.g. registers, register files, memories, and FIFO's that can be written/ read from software or hardware).

Reuse of Design Specifications
Designs may be reused by loading them, saving a new project and then modifying the new project to create a proliferation design.

GUI Infrastructure Specification Compilation, Checker, Code Generator
The GUI interfaces with a software tool or compiler which parses the specification entered using the GUI. The GUI contains a dialog box to configure the compiler options. The GUI passes the options to the compiler which controls the compilation and code generation process. The compiler will compile and check the specification. The GUI displays the compiler warnings and errors in a window. The user can click on a warning or error and a text editor or dialog box with the specification will appear and the source of the warning or error will be highlighted within the GUI. The user can then fix the violation and recompile the modified specification. The GUI contains a menu item to generate the code.