

USING VHDL BEYOND SYNTHESIS

Krishna Kumar
Maria Tovey
Sanjay Sawant
Phil George
Cadence Design Systems
555, River Oaks Parkway
San Jose, CA 95134

1 Abstract

This paper outlines our experience from the investigation of the applicability of VHDL to communicate physical properties to and from the logic design and physical design tools. This information transfer is implemented using a combination of VHDL modeling and design methodologies and the present-day tools. The information flow between existing tools is described first and then the impact of the proposed methodology on these tools are discussed. The area of application chosen for this study is board designs using standard off-the-shelf components.

2 Introduction

In the design cycle of a digital device, the internal organization of various components is obtained after the logic design phase. This internal organization is represented either as a schematic or a netlist. The design process continues as this information is passed on to the physical design tools, which are used to package, place and route the various components. These components may be off-the-shelf parts or ASIC cells. Information is back-annotated to the schematic from the physical design tools, to maintain the logic to physical instance relationships and to perform analysis using actual delays. For rules-driven designs, the designer can specify some requirements and constraints to the physical design tool resulting in information flow to and from the logic and physical design tools. After physical design, simulations are performed on the back-annotated schematic to ascertain the functional correctness of the design. Physical design tools also perform analysis of the design, like thermal, reliability and sig-noise analysis, to check the manufacturability of the design.

The VHSIC hardware description language (VHDL) is used for the description of electronic circuits. The multi-level descriptive capability and the flexibility of having different configurations and multiple architectures, have made VHDL a viable language for use in the design of digital devices. Various commercial simulators and synthesis tools have accelerated this use of VHDL.

VHDL is being used for the design of systems, boards, ASICs and ICs. It is used to describe the desired device at a high level of abstraction, behaviorally, and is synthesized until the final architecture is realized. Based on the complexity of the behavior and the level of abstraction, the synthesis is either automatic or manual decomposition. The output of the the synthesis tools

is the netlist of the design. This netlist can be described in VHDL . The methodology presented here uses the VHDL netlist to communicate with the physical design system. A large portion of this effort is the mapping of attributes and properties described in a non VHDL vendor-dependent format to VHDL. Section 3 describes the interaction between today's commercial tools. The proposed methodology is described in the section 4. Section 5 contains a comparative study with the proposed EIA modeling standard (EIA-567B), and future trends.

3 Present day methodology

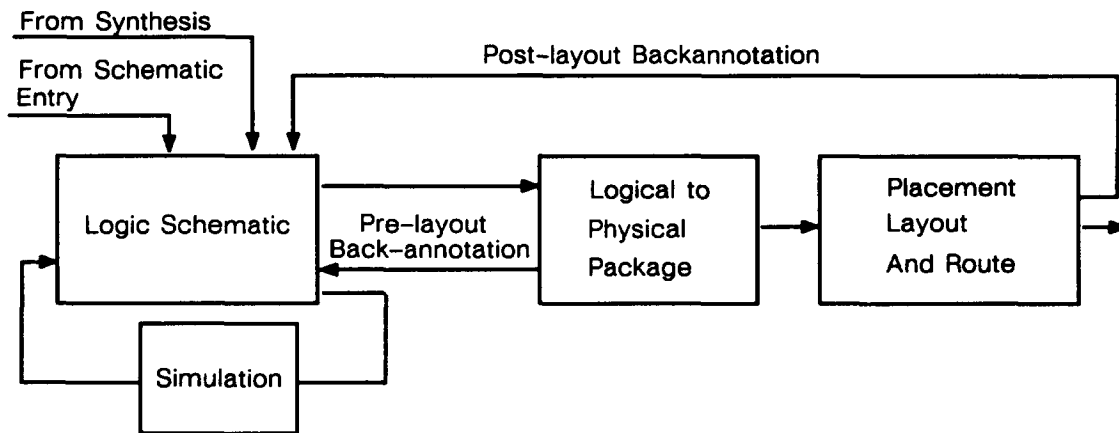


Figure 1 : Information Flow Between Design Phases

In general the overall design process can be distinctly divided into seven steps.

- Creation Of Logic Schematic
- Functional Simulation and Timing Analysis
- Packaging
- Pre-layout Backannotation
- Placement, Layout and Route
- Post-layout Backannotation
- Post-layout Simulation

For digital designers, the goal is to create the logic schematic that represents their design. Due to the increasing complexity and size of the design, many engineers are using some HDLs to describe the devices, before creating the schematics. The hardware description is then synthesized to get a logic schematic. The logic synthesis has improved the design quality and time to market. It acts as a catalyst for the top-down methodology and technology independent designs. However, small designs are still created using off-the-shelf schematic capture tools from the EDA vendors. The logic schematics are simulated to verify their functionality. Typically these simulations are performed in test fixtures to check both the functionality and interoperability with other parts of the design. Once the design is verified, separate timing verification is performed to determine the critical paths.

The design is packaged prior to sending the netlist to the physical design system. Primary function of the packager is to assign the gates to the individual packages and to create a netlist describing physical information necessary by the board layout system. This primarily includes physical reference designators, pin numbers and the user-defined attributes/properties. After successful packaging and before sending the netlist to physical design, the schematic is

back-annotated to get the physical reference designators, pin numbers and attributes are placed on the schematic.

There are several state-of-the-art physical design systems available from the EDA vendors. Typically, all of them use a netlist created by a "packager". Most of the systems allow a designer to describe his/her rules at the schematic stage and uses those rules during placement and routing. The physical design analysis is also performed. Post-layout back-annotation is carried out to extract the accurate wire delays and parasitics from the physical design.

To complete the design process the information extracted from the physical design is used to re-simulate the design. Throughout the design process, the designer can influence the tools by specifying rules, and by selecting various options through the command files.

4 Proposed methodology

The rules specified by the logic designers, the back-annotated delays and pin numbers are either attached to the schematics as properties or specified as options and flags in the command files. VHDL can be used to represent the netlist, by including all the properties and rules into the description. But for this description to be used by various tools, some methodologies have to be followed. In general all the properties and flags can be modeled in VHDL using a framework of generics, attributes and constants. Though all these three constructs hold static values, the values are specified in different ways.

In this methodology, all these three constructs are used. Generics are used to model values that are used during VHDL simulations, like delays and delay flags. The attributes are used to model all the other values. These values are not used by the VHDL simulators, and are solely used for the information transfer between the various tools. Global flags are modeled using deferred constants. The constructs were chosen also based on the flexibility of back-annotation.

The interface objects (ports and generics) are standardized for inter-operability and portability. Things enforced are a single logic-type for all the ports and a set of generics for all the standard components, user-defined modules, and commercially procured models. The VLT, a sixteen state type defined in the `valid_logic_package` is used. This package has the type VLT defined with all the overloaded logic operators and three forms of resolved types. This type was chosen for compatibility with the existing gate-level simulator. Every model has to have a specified set of generics. These generics help the user to select the various delays and modes during the design simulations (figure 2). Generics are used to choose between propagation delay or pin-to-pin delays, with a choice of min, typ and max values. Wire delays are lumped into input wire delays, and the pin number generics are used for the back-annotation after packaging.

The attributes used are declared in the package `attribute_decls` (figure 3) in the library `common`. The attributes can be categorized as containing pre-layout information or post-layout information. The pre-layout information exists as rules and properties between the logic schematic and the packaging/layout tool. These attributes are mostly instance specific. In VHDL the attributes are specified for the labels of the instances. The post-layout information is described as signal-specific attributes and these exist as rules and properties between the placement tool and the logic schematic. Designers can add design-specific properties as attributes.

```

library valid, common ;
use valid.valid_logic_package.all ;
use common.timing_package.all ;
entity F112 is
  generic (p_delay : boolean ;          edge_dep : boolean ;          delay_mode : delay_mode_type ;
           i_j_wire_delay : time ;      i_k_wire_delay : time ;      i_clk_wire_delay : time ;
           i_clr_wire_delay : time ;     i_pr_wire_delay : time ;     pin_number_j : natural ;
           pin_number_k : natural ;      pin_number_clk : natural ;   pin_number_clr : natural ;
           pin_number_pr : natural ;     pin_number_q : natural ;    pin_number_q_bar : natural) ;
  port (j, k, clk, clr, pr : in vlt ;
        q, q_bar : out vlt) ;
  attribute in_load_factor of j : signal is 1 ;
  attribute in_load_factor of k : signal is 1 ;
  attribute in_load_factor of clk : signal is 1 ;
  attribute in_load_factor of clr : signal is 1 ;
  attribute in_load_factor of pr : signal is 1 ;
  attribute out_load_factor of q : signal is 1 ;
  attribute out_load_factor of q_bar : signal is 1 ;
end ;

```

Figure 2 : Example Interface Of a Library Component

```

package attribute_decls is
-----
-----  ATTRIBUTES USED FOR INFORMATION TRANSFER FROM/TO PACKAGER
-----  These attributes are associated to the component instance (label)
--      and are used (if present) by the packager
-----
type jedec_type is (dip, solc, plcc) ;
attribute group : string(1 to 16) ;   attribute jedec : jedec_type ;
attribute location : string(1 to 16) ; attribute no_load_check : boolean ;
attribute pin_group : string(1 to 4) ; attribute section : positive ;
-----
--  ATTRIBUTES USED FOR INFORMATION TRANSFER FROM/TO PLACEMENT & ROUTE TOOL
--  These attributes are associated to the component instance (label) or the signal and are used (if present) by
--  the placement, layout and route tool
-----
-- Instance specific
attribute cost : real ;                attribute room : string(1 to 16) ;
attribute tolerance : positive ;      attribute value : positive ;
-- signal specific
type distance is 0 to 1E20
units
  -- units A, nm, um, mm, cm m
end units ;
type resistance is 0 to 1E20
units
  ohm ; Kohm ; Mohm ;
end units ;
type delay_rule_type is record
  source_ins : string(1 to 16) ;      source_pin : string(1 to 16) ;
  dest_ins : string(1 to 16) ;       dest_pin : string(1 to 16) ;
  delay_in_time : boolean ;          delay : time ;          length : distance ;
end record ;
attribute DRC_class : string(1 to 16) ;   attribute delay_rule : delay_rule_type ;
attribute driver_term_val : resistance ;   attribute ecl : boolean ;
attribute load_term_val : resistance ;     attribute no_gloss : boolean ;
attribute no_pin_escape : boolean ;       attribute no_rip_up : boolean ;
attribute no_route : boolean ;           attribute no_swap_gate : boolean ;
attribute no_swap_gate_ext : boolean ;    attribute no_swap_pin : boolean ;
attribute route_priority : positive ;     attribute route_line_width : distance ;
attribute route_to_shape : boolean ;     attribute stub_length : distance ;
attribute weight : positive ;
end ;

```

Figure 3 : Attribute Declarations

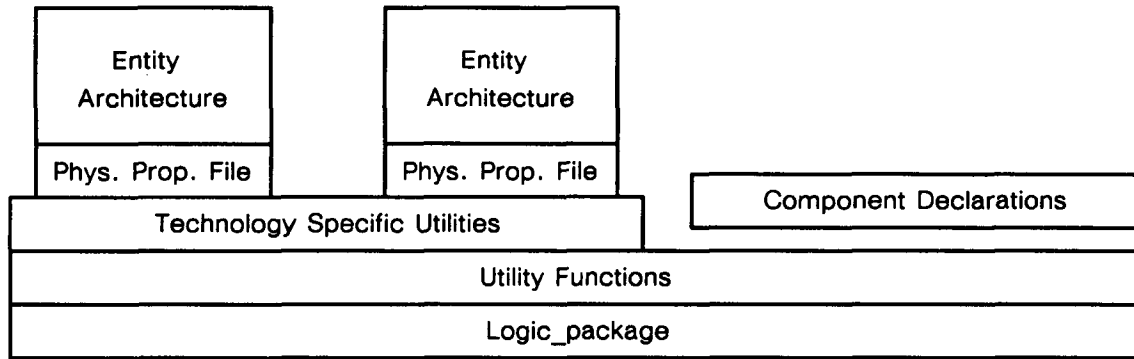


Figure 4 : Design Units In Model Library

Global flags are usually specified in the form of command files, in the tools. Here, these flags can be documented with the design, as values of deferred constants.

Figure 4 shows the relationship between the various VHDL design units in a model library. The `logic_package` is used by all the models. The library models can also use one or more packages with utility functions. Along with any library-specific or technology-specific utilities package, a package with component declarations is also present. Each model has an entity and an architectures and a physical property file. The entity is all the ports and the required generics. The architecture contains statements that are used to describe the function of the device. All the generics should be used for the calculation of the delays. The physical property file contains information regarding the physical package/s of the device. The packager is presently set-up to use the physical properties and pass on those to the placement tool. The format of the physical property file is vendor specific, in this case specific to Valid.

The schematic entry tool is used to create the schematic. The flattened schematic is compiled, using a translator/compiler. The translator/compiler is an enhanced compiler, that can translate the schematic into the equivalent VHDL structural architecture. The entity/architecture pair is stored in the design directory with the ASCII and binary files needed to save the schematic. The packaging tool takes as input the VHDL entity architecture or the schematic, and packages the design and produces the new architecture with the packaged information including the three files used to communicate to the physical design tool, Allegro. The placement, route and layout is completed, and is saved as a new architecture in the design directory. The schematic cannot be saved into the design directory without going through the translator/compiler. This is very essential to maintain information consistency between the two views of the design. Figure 5 shows the various files created during design process. New architectures represent the modified design in the process.

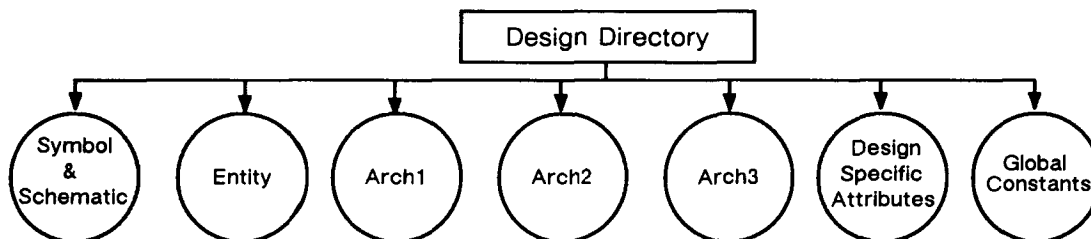


Figure 5 : Files In Design Directory

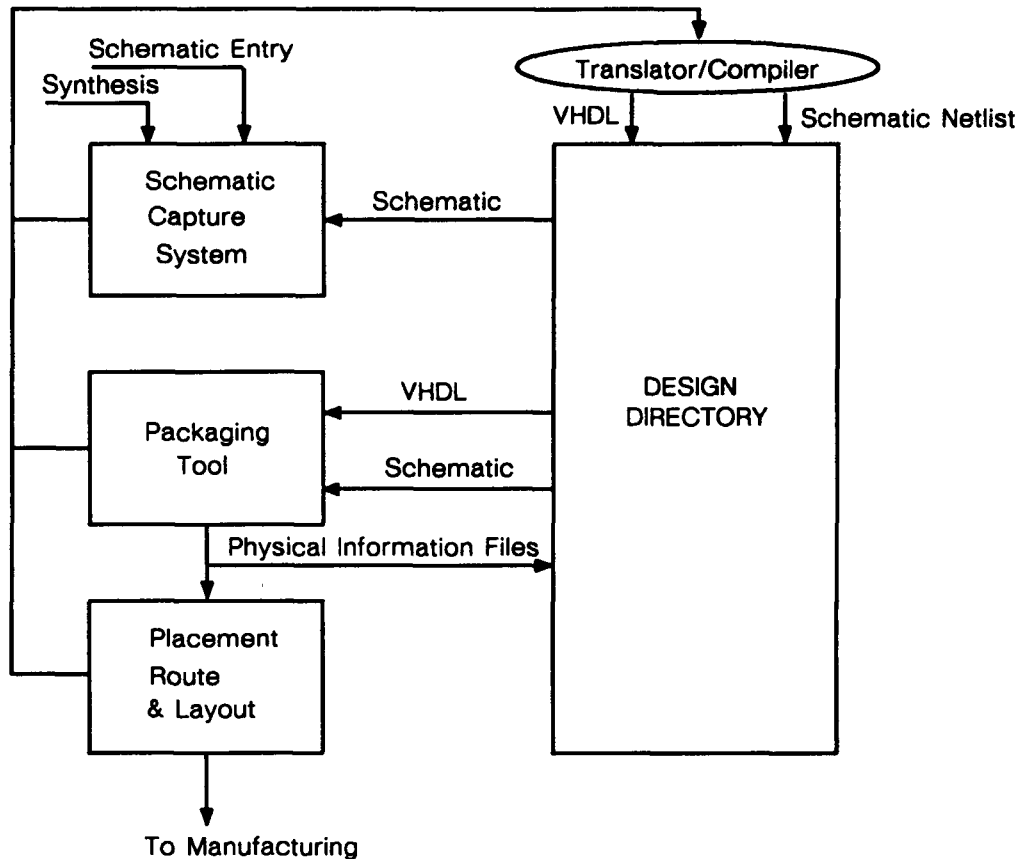


Figure 6 : Process Flow

Figure 6 shows the various software components needed for the incorporation of VHDL. Geometric information about the schematics are not stored in the VHDL descriptions. Hence, the binary and ASCII files are also stored in the design directory. Since the schematic capture tool and the packaging tool already work off these files, these tools can still be used with slight enhancements. The important new component is the translator/compiler. The translator/compiler can also be incorporated with design framework software, for better data management and version control.

The schematic capture system, is used to create and view the logic schematic. The schematic can be drawn or translated from VHDL or EDIF netlist representations. When a new schematic is saved in to the design directory, the translator/compiler checks the schematic and then creates the VHDL entity (if not present) and the architecture corresponding to the schematic. Other VHDL files, like packages for design-specific attributes and global flags are also created. The packaging tool takes as input the VHDL entity architecture or the schematic, and packages the design and produces the new architecture with the packaged information. Three ASCII files are created with the connectivity and physical properties of the design. These files are stored in the design directory, and are used by the placement, route and layout tool for physical design. The physical delays and other information are back-annotated through the translator/compiler and stored as new versions of schematics and new architectures.

5 Related topics and future trends

The proposed EIA-567 VHDL modelling standard effort tries to define standard interfaces and requirements for commercial VHDL component models. This methodology is to facilitate the inter-operability of models procured from various sources. The significant difference between the approach given here and the EIA's approach, is that this methodology was developed with a target set of tools in mind. The scope of this study was to map the existing properties, options and features in VHDL. The EIA-567 is much more general, and the information requirement and interface constraints are independent of any suite of tools. The EIA-567 requirement is much more formal, though both the approaches do use generics, attributes and deferred constants. The important deviation is the electronic data book (in EIA-567, containing the electrical view, timing view and physical views) is a set of packages, and the physical property files in this methodology is specific to a vendor. The EIA requires the models to be described in IEEE 1164 9-state type. Most of the delays, extrinsic and intrinsic, are handled similarly in both methodologies. While EIA-567 requires flags to turn on or off the 'X' generation during any timing violations, our models generate 'X's for all such violations. The validation and verification using accompanying test benches are no requirement in this methodology, since most of the models would be developed internally.

Our study was concentrated on the physical design tools used for system design. However, for the physical design of ASIC/ICs, most of the issues are quite similar. This methodology is applicable for their physical design since the information flow for the design of ASIC/IC is similar to that of systems. The important aspect, is that the present day ASIC cells are described, such that the calculations of the load dependent delays are coded internal to the model. In this methodology, the delays can be calculated based on the load information that is provided statically.

The proposed changes in VHDL92 can have significant impact on this methodology. The enhancements of User-defined attributes and the availability of Private types will make VHDL more viable. The availability of attribute like 'pathname will assist in making back-annotation much simpler. Component instantiations without component declarations will be very suitable in this domain, as the entity/architecture pairs are known during these stages.

Our methodology was targeted on a particular solution. A major enhancement would be the addition of framework data management software and translators to/from schematics and VHDL. These tools would help include hierarchical and structured schematics. Many of the properties that are described as attributes can become generics, and controlled by configurations.

6 Conclusion

This study shows that VHDL can be used for the information transfer between the logical and physical design phases. There are several advantages in utilizing VHDL for such information transfer. The incorporation of models from various sources is very easy, and the portability of the design data between different tool, is available. This facility promotes a better concurrent design environment, with little or no translations during design. This approach promotes the idea of a single simulator for all functional simulations from a high-level of abstraction to gate-level. In general simulations to check the functionality, can be performed during any design phase. All this is assuming that the VHDL simulator is tremendously fast and back-annotation is automatic.


```

library valid ;
use valid.valid_logic_package.all ;
entity decade_counter is
  port (enable : in vlt ;
        clear : in vlt ;
        clk : in vlt ;
        carry : out vlt ;
        q3 : buffer vlt ;
        q2 : buffer vlt ;
        q1 : buffer vlt ;
        q0 : buffer vlt ) ;
end ;

library common, fast;
use common.attribute_decls.all ;
use fast.component_decls.all ;
use common.global_signals.all ;
architecture ver_1 of decade_counter is
  -- Internal signal declarations
  .....

  -- attribute specifications
  attribute group of U1 : label is "      A" ;
  attribute group of U3 : label is "      A" ;
  attribute group of U5 : label is "      B" ;
  attribute group of U10 : label is "     B" ;
  attribute value of R1 : label is 1 Kohm ;
  attribute tolerance of R1 : label is 1 ;
  attribute value of R2 : label is 1 Kohm ;
  attribute tolerance of R2 : label is 10 ;
  attribute value of R3 : label is 2 Kohm ;
  attribute value of R4 : label is 2 Kohm ;

  for all : F112 use entity fast.F112(simple) ;
  for all : F11 use entity fast.F11(simple) ;
  for all : F00 use entity fast.F10(simple) ;
  -- configuration specifications for all instances
  -- ....

begin
  U1 : F112 generic map
    (p_delay => true,
     i_j_wire_delay => 0 ns,
     i_clr_wire_delay => 0 ns,
     pin_number_k => 0,
     pin_number_pr => 0,
     port map (j => enable,
               clr => clear,
               q_bar => open) ;
     edge_dep => false,
     i_k_wire_delay => 0 ns,
     i_pr_wire_delay => 0 ns,
     pin_number_clk => 0,
     pin_number_q => 0,
     k => enable,
     pr => sig_0001,
     delay_mode => max,
     i_clk_wire_delay => 0 ns,
     pin_number_j => 0,
     pin_number_clr => 0,
     pin_number_q_bar => 0)

  R2 : resistor generic map (pin_number_node_1 => 0, pin_number_node_2 => 0)
    port map (node_1 => vcc,
              node_2 => sig_0001) ;

  U2 : F11 generic map
    (p_delay => true,
     i_in1_wire_delay => 0 ns,
     pin_number_in1 => 0,
     pin_number_out1 => 0 )
    port map (in1 => enable,
              out1 => sig_0003) ;
     edge_dep => false ,
     i_in2_wire_delay => 0 ns,
     pin_number_in2 => 0,
     in2 => q0,
     in3 => sig_0002,
     delay_mode => max,
     i_in3_wire_delay => 0 ns,
     pin_number_in3 => 0,
     pin_number_in3 => 0,

end ;

```

The pre-layout instance, back-annotated with the pin numbers :

 -- Example of a back-annotated instance (after packager)

```

U6_U1 : F112 generic map
  (p_delay => true,
   i_j_wire_delay => 0 ns,
   i_clr_wire_delay => 0 ns,
   pin_number_k => 12,
   pin_number_pr => 10,
   port map (j => enable,
             clr => clear,
             q_bar => open) ;
   edge_dep => false,
   i_k_wire_delay => 0 ns,
   i_pr_wire_delay => 0 ns,
   pin_number_clk => 13,
   pin_number_q => 9,
   k => enable,
   pr => sig_0001,
   delay_mode => max,
   i_clk_wire_delay => 0 ns,
   pin_number_j => 11,
   pin_number_clr => 14,
   pin_number_q_bar => 7)
   clk => clk,
   q => q0,

```