

AUTOMATED VHDL MODEL GENERATION FOR PROGRAMMABLE LOGIC

Neal S. Stollon, P.E.
Texas Instruments - Defense Systems and
Electronics Group, Dallas, Texas.

ABSTRACT: This paper will discuss VHDL modeling techniques for programmable logic design and a suite of translation tools developed for the automated generation of VHDL models from some typical programmable logic design formats. The application of PLD VHDL models to systems level integration in VHDL design automation environments will also be addressed.

INTRODUCTION:

Programmable logic devices (PLDs) have become a widely used and performance effective approach to implementing many diverse types of electronics systems. An increasingly wide range of programmable parts have become available in the marketplace to fill application niches formerly occupied by TTL MSI logic standard parts. Programmable parts can be roughly drawn into 3 families:

- PALs: 50-500 gates complexity with logic matrix structure
- EPLDs: 500-5K gates complexity with multiple logic matrices
- FPGAs: 1K+ gates complexity with logic block array structure

Modeling capabilities in VHDL converge well to applications and design approaches in programmable logic. The majority of PLD usage is based on boolean equation and state machine designs that map to standard structures in VHDL. This paper addresses a suite of VHDL model generation tools developed at Texas Instruments Defense Systems and Electronics Group (TI-DSEG) to support programmable logic design. The tools support generation of VHDL models from standard PLD design representations such as table formats, PLD specific HDLs (PALASM), and programmable logic architectural descriptions (PAL JEDEC fusemaps and FPGA netlists).

VHDL MODELING OF PLD DESIGNS

The PLD design tool arena has historically been driven by modeling languages. PALASM (the first PLD HDL) was released concurrently with the initial PAL devices from Monolithic Memories (now AMD) in 1982. PALASM has been followed by other industry driven programmable logic HDLs (ABEL, AHLG, MINC, CUPL, etc.) that continue to address specific programmable logic design approaches and issues. VHDL itself has had limited support from mainstream PLD design tool vendors. PLD design automation has historically been PC-based. The recent introduction of PC based VHDL capabilities, adequate for simulation of PLD designs at both the system specification and device model level, has allowed both PLD vendors and users to seriously consider the applicability of VHDL for PLD design.

Modeling of PLDs using VHDL has been discussed previously [1,2] from a viewpoint of importing PLD fusemap information into the VHDL model.

This work builds on, but differs from, the referenced approaches in the way that the VHDL model accesses the PLD design information and in the impact of VHDL application to design methodologies at TI-DSEG:

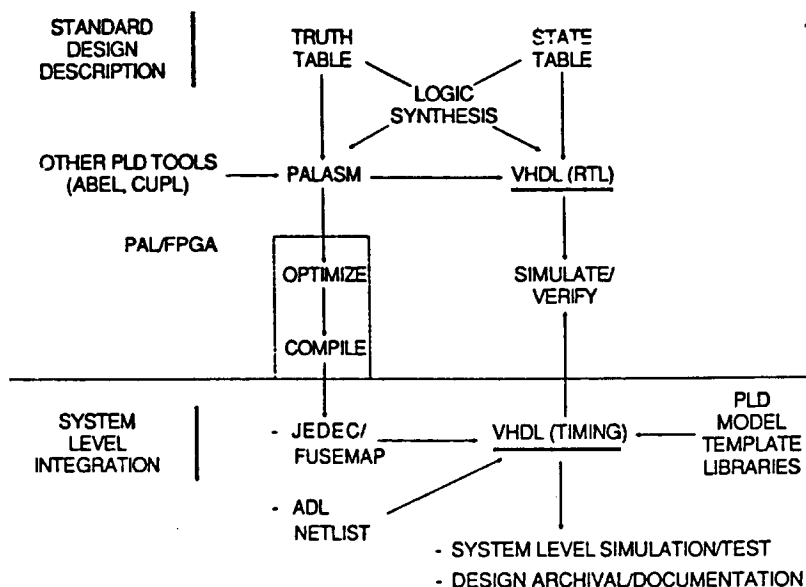
1. An attempt was made to provide some VHDL modeling capability at all of the major phases of a PLD design flow (abstracted behavior, equation level, physical implementation). The capability to address multiple modeling views helps to insure cohesive top down modeling documentation and allows the design models to be logically compared with each other using a single testbench.

2. The generated VHDL models have all equations and variables from a designated design implicitly embedded; as opposed to having the VHDL read the design data from external formats (using a data interpreter as part of the VHDL model) during model operation. The incorporation of the design specific information as part of the model generation effort helps to support stand-alone VHDL design model usage and was considered to be key in ensuring VHDL model readability.

3. The generated VHDL models are essentially stand-alone; the only vendor supplied packages needed are logic supporting the '0', '1', 'X', and 'Z' states. All the generation programs allow logic systems to be user defined. Given the multiplicity of VHDL tools in complex design environments and the need for inter-operability of models between the VHDL tools, significant dependency on application or vendor specific packages was considered a risk, compromising VHDL model usability.

VHDL application has still not achieved mainstream acceptance by many design engineers. Model generators support VHDL use without relying exclusively on the engineer's VHDL modeling expertise. One major advantage of this approach is that a consistent set of VHDL models can be created easily and concurrently within a top down design methodology (Fig. 1). The ability to integrate PLD models for a board or system level simulation originally justified this VHDL modeling tools development effort.

FIG. 1 : A PLD DESIGN FLOW WITH VHDL DATA TRANSLATION



PLD SPECIFICATION MODELING IN VHDL

As seen in Fig. 1, initial PLD design can be performed using a range of input formats. Schematic capture can be used in PLD design, but can often result in an inefficient design representation during logic optimization and compilation. More effective logic design for PLDs uses synthesizable formats, such as table based entry (applicable for both combinational and state machine logic) and equation based HDLs.

Table Entry: Table based combinational logic and finite state machine (FSM) design synthesis have been extensively researched as design automation topics. Many of the widely used synthesis algorithms and tools in IC and PLD design are outgrowths of UC-Berkeley research in this area during the 1980's.

Berkeley format truth tables (PLA) and state machine tables (KIS) are widely accepted by most synthesis tool vendors. The ability at TI-DSEG to model these tabular formats in VHDL was originally developed for VHDL driven synthesis tool evaluation, but was soon recognized as as forming a desirable option for system level simulation of both pre and post-synthesis models. Approaches for logic and FSM modeling in VHDL have been discussed in depth, and development of automated truth and state table format translation to VHDL description proved straight forward, using vendor recommended PLA and FSM model templates [3]. An example of VHDL generated from a state table is shown in Listing 1.

LISTING 1 : FSM TABLE and STATE MACHINE VHDL MODEL

```
# lion fsm
.i 2
.o 1
.p 11
.s 4
-0 st0 st0 0
11 st0 st0 0
01 st0 st1 -
0- st1 st1 1
11 st1 st0 0
10 st1 st2 1
1- st2 st2 1
00 st2 st1 1
01 st2 st3 1
0- st3 st3 1
11 st3 st2 1
.end

entity LION is -- FSM to VHDL Translation
port ( CLK, in1, in2 : IN BIT; out1 : OUT BIT );
end; -- entity LION
architecture behavior of LION is
type STATE_TYPE is (st0, st1, st2, st3);
signal CS, NS : STATE_TYPE; begin
process begin
wait until CLK='1'; CS<=NS; end process;
process ( CS, in1, in2 ) begin
CASE CS is
WHEN st0 =>
if (not in2)='1' then out1<='0'; NS<=st0; end if;
...
WHEN st3 =>
if (not in1)='1' then out1<='1'; NS<=st3; end if;
if in1 and in2='1' then out1<='1'; NS<=st2;
end if; end case; end process;
end behavior; -- model: LION
```

HDL driven design: As previously discussed, numerous PLD HDLs exist and are supported as commercial products. PALASM is generally accepted as among the more standardized HDLs for PLD design. PALASM has vendor supported translation paths from other HDLs (ABEL, CUPL) and can be applied as an input format into FPGA synthesis tools (ie. ACTEL ALES and Xilinx XNFOP optimization). In many design scenarios, a VHDL model was desired for systems modeling before the final device selection was implemented. To support the VHDL tools environment in TI-DSEG, a portable and generic PALASM to VHDL translation capability (with minimal package dependencies) was developed in 1991 to address VHDL model generation of PLD specifications.

As with most PLD HDLs, PALASM's model structure supports generalized one for one replacement with VHDL constructs. With some heuristics on PALASM language structure, VHDL entity generation (input, output, and inout port declarations) can be derived from the PALASM I/O signal list. Building the VHDL architecture is similarly direct. With an understanding of PALASM's logic precedences, PALASM combinational equations remap into concurrent VHDL statements and PALASM sequential and state machine operations remap into clocked VHDL processes. Some of the more typical PALASM to VHDL correspondences are shown in Table 2. An example of VHDL generated from PALASM is given in Listing 2.

TABLE 1: Some PALASM to VHDL Correspondences

CHIP DECLARATION	ENTITY DECLARATION
SIGNAL LIST	PORT MAP
EQUATIONS	ARCHITECTURE
X = /A * B + C	X <= ((NOT A) AND B) OR C;
Y.CLKF = CLK	CLOCKED PROCESS (CLK)
Y := X	WAIT FOR CLK='1'; Y <= X;

LISTING 2: PALASM EQUATION FILE and VHDL EQUATION (RTL) MODEL

```

; 4 states, 2 state bits
CHIP LION generic
CLK in1 in2 out sbt0 sbt1

EQUATIONS
out1 = (/in1 * sbt0)
      + (in1 * /sbt_0 * sbt1)
      + (/in1 * sbt_1)
sbt0 := (/in1 * /sbt1)
      + (in1 * /sbt0 * sbt1)
      + (/in1 * sbt0 * sbt1)
sbt0.CLKF = CLK
sbt1 := ...
sbt1.CLKF = CLK

entity LION is -- PALASM TO VHDL TRANSLATE
port ( CLK, in1, in2 : IN std_logic;
      out : OUT std_logic;
      sbt0, sbt1 : INOUT std_logic );
end; -- entity LION
architecture rtl of LION is begin
out1 <= ((not in1) and sbt0)
      or (in1 and (not sbt0) and sbt1)
      or ((not in1) and sbt1);
process (CLK) begin
if CLK'event and CLK='1' then
sbt0 <= ((not in1) and (not sbt1))
      or (in1 and (not sbt0) and sbt1)
      or ((not in1) and sbt0 and sbt1);
sbt1 <= ...
end if; end process;
end rtl; -- model: LION

```

PLD DEVICE MODELING IN VHDL

The previous section discusses options of VHDL model generation during design entry and analysis stages in a PLD design. Tabular and equation based approaches to generating VHDL models of a PLD function produce detailed Register Transfer Level (RTL) model descriptions. RTL modeling, while appropriate for functional simulation, does not include the physical information to support defining and embedding of path delay and timing information and/or checking of device level performance. As shown in Fig. 1, the transformation of a design into a programmable device requires compiler/programmer (PALs/EPLDs) or routing (FPGA) steps. Modeling of PLD timing information before this design transformation step is typically not viable with a high degree of confidence due to lack of physical implementation information.

The data file standard for compiling and programming of PAL and EPLD designs is via a JEDEC (Joint Electronic Devices Engineering Council)

format. The JEDEC file defines fuse location and status of the device programming fusemap. As an immediate precursor to a programmed part, the JEDEC file supports creation of a VHDL model based on the fusemap structure with the physical information needed for generating timing complete VHDL models. Timing models of PAL/EPLD devices are typically straightforward, since the device timing is highly deterministic due to the characteristic sum-of-products architecture of PALs and EPLDs.

A JEDEC file to VHDL model generation program was written in 1991 for PAL devices in the 16xxx thru 22Vxxx families (most widely used by Texas Instruments) to support timing based modeling of these parts at the pcb or system level. Extracting PAL design information for VHDL modeling can encompass fuse mapping of optional features in device input, output, and feedback architecture sections, as well as the AND/OR matrix that typically defines the logic function. In addition to fusemap data, specifications on the device structure (AND/OR matrix configuration, IO and feedback options, nodal timing) are used for model building. The modeling program handles this by accessing device specific files extracted from the PAL vendor's data book. Due to the highly regular nature of PAL/EPLD timing, device delay can be modeled in VHDL using generics. An example of VHDL generated from a PAL JEDEC format is given in Listing 3.

LISTING 3: JEDEC FUSEMAP and PAL DEVICE VHDL MODEL

```

PALASM90 PAL ASSEMBLER
PAL16R4
PINS CLK:1 IN1:2 IN2:3
PINS OE:11 OUT:13
L0000 0000000..000000..*
L0992 0000000..000000..*
L1024 1111101..101110..*
L1056 0111111..111111..*
L1088 0000000..000000..*
L1248 0000000..000000..*
L1280 1111101..101110..*
L1312 0111111..111111..*
L1344 0000000..000000..*
L1504 0000000..000000..*
L1536 1111111..111111..*
L1568 0111011..101111..*
L1600 0111101..011101..*
L1632 1111111..101110..*
L1664 0000000..000000..*

entity LION is -- JEDEC to VHDL translation
generic (tpd : TIME := 15 ns; -- in to comb.
         tsu : TIME := 25 ns; -- clk set up
         tdcl : TIME := 10 ns); -- clk to reg
port ( CLK, IN1, IN2, OE : IN std_logic;
      OUT : INOUT std_logic );
end; -- entity LION mapped to PAL16R4
architecture pal of LION is
SIGNAL s5, qb_5, s6, qb_6, s7 : std_logic := '0';
SIGNAL q_5, q_6 : std_logic := '1';
begin -- COMBINATIONAL ARRAY (AND PLANE)
s5 <= ((not IN2) and (not qb_5)) or (IN1);
s6 <= ((not IN2) and (not qb_6)) or (IN1);
s7 <= (IN1 and IN2 and (not qb_5))
or ((not qb_5) and (not qb_6));
process (CLK, s5, s6) begin
if CLK'event and CLK='1' then
ASSERT(s5'stable(tsu) and s6'stable(tsu))
REPORT "FF setup timing" SEVERITY warning;
q_5 <= s5 after tdcl; qb_5 <= NOT s5 after tpcl;
q_6 <= s6 after tdcl; qb_6 <= NOT s6 after tpcl;
end if; end process;
OUT <= NOT s7 after tpd;
end pal; -- model: LION mapped to PAL16R4

```

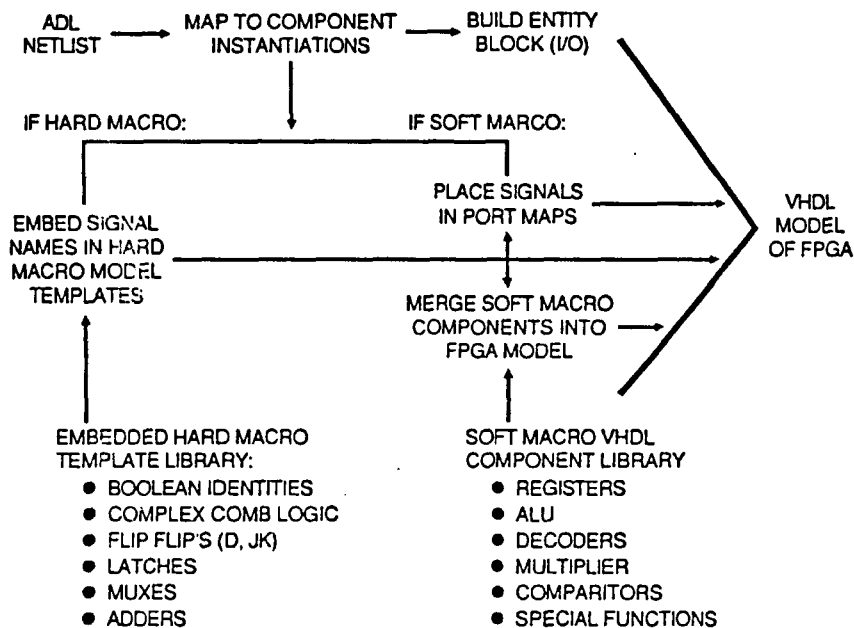
A flow diagram for the VHDL PAL model generation from JEDEC files is shown in Fig. 2. The approach and algorithms used are extendable to EPLD devices which share the same JEDEC fusemap format definition for programming. A somewhat different approach to creating a VHDL model from JEDEC files for EPLD devices has been discussed previously [2].

FPGAs rely on vendor specific (non-standard) formats for transferring design data to device programmers. FPGA design provides a significant

FPGA netlists consist of multiple instances of pre-defined functions. Functions are characterized as using one logic block (hard macros) or being implemented using multiple hard macros in a sub-netlist format (soft macros). Hierarchical libraries were developed for both hard soft FPGA macro functions (Fig. 3) in the VHDL modeling program. VHDL model templates are accessed from a look-up file for each of the 170+ hard macro names allowed in an ADL netlist. This VHDL macro look-up table is referenced at each logic block instance in the ADL netlist.

If an ADL function name does not map to a hard macro template, it is assumed to be part of a soft macro. ADL soft macros are modeled in VHDL as stand-alone models (pre-generated and stored in a VHDL soft macro library) which are mapped hierarchically into the primary FPGA design model file using VHDL port map functions. Fanout driven timing is built into all the model blocks by reference to built-in tables of databook statistical timing data. This approach has proved viable for modeling the large netlists found in FPGA design. ADL files with over 500 logic function instances have been modeled in VHDL. Xilinx based design has also been modeled in VHDL (using Exemplar Logic Synthesis netlist translation). Xilinx XNF files were converted to ADL netlists and modeled in VHDL using the above modeling process.

FIG. 3: FPGA NETLIST TO VHDL MODEL FLOW



DEVICE TIMING ISSUES IN VHDL

The use of typical timing data in the FPGA models provides a useful approximation, but detailed PLD (as in ASICs in general) timing for the VHDL model requires backannotation of inter-block timing from the routing step. These delay timing numbers are typically asymmetrical for rising and falling logic transitions, which presents significant issues in handling the VHDL representation of the timing information. Since desired timing numbers are not known until after logic block operation is evaluated by a VHDL simulator, some method of allowing timing corrections after the logic operation is completed must be

addressed. Some options being investigated for addressing this are:

1. Assigning an intermediate variable for each node name in the FPGA description (lengthy for a several thousand gate design) to act as a buffer for correcting the timing based on the logic resolution
2. Using some timing function package to determine the proper timing value to be assigned to the logic transition based on previous states
3. Using the worst case post-routing timing number (min to max post-routing timing values are typically within a 20% worst case window) for modeling the inter-block timing.

Optimal approaches (from both computational and performance viewpoints) to address asymmetric delay timing found in FPGAs remains an area under investigation.

SUMMARY

An approach of generating VHDL models for different points in the PLD design flow has been presented. VHDL generators are discussed for translating truth and state table, PALASM HDL, JEDEC fusemap, and FPGA netlist data into VHDL. These provide a method of access to PLD design information at various stages in a VHDL oriented design flow with minimal user VHDL expertise. VHDL model availability provides a vendor independent method of device and board level modeling for use in simulation, analysis, and data archival. The VHDL model generation tools address inter-operability concerns by using VHDL constructs that comply with typical VHDL synthesis subsets. VHDL models created are also designed to be generally vendor and application package independent.

Automation of VHDL modeling for PLDs has proven useful in enhancing PLD synthesis, simulation, and documentation efficiency for several TI-DSEG project design phase(s). The availability of VHDL models is expected to prove beneficial during any future design-fitting and technology migrations that are typical in product design life cycles.

ADL is a trademark of Actel Inc. ABEL is a trademark of DATA I/O Corp.
MINC is a trademark of MINC Inc. AHDL is a trademark of Altera Inc.
PALASM is a trademark of AMD Inc. XNF is a trademark of Xilinx Inc.
Exemplar Logic Synthesis is a trademark of Exemplar Logic Inc.
CUPL is a trademark of Logical Devices Inc.

- [1] Coelho D. "The VHDL Handbook" Chap. 5.4, Kluwer Academic Publishers
- [2] Lee C.H. "Modeling of EPLD in VHDL", Fall 1989 VHDL User Group
- [3] Carlson S. "Introduction to HDL-Based Design using VHDL", Synopsys Inc.