AFRREV STECH
An International Journal of Science and Technology
Bahir Dar, Ethiopia

Vol.1 (2) April-July, 2012:85-100

ISSN 2225-8612 (Print) ISSN 2227-5444 (Online)

Hardware Descriptive Languages: An Efficient Approach to Device Independent Designs with Complex Programmable Logic Devices and Field Programmable Gate Arrays

Ezekwe, C. Gene
Electronic Development Institute, Awka

&

Okide, Samuel O.
Department of Computer Science,
Nnamdi Azikiwe University, Awka
E-mail: samuelokide@yahoo.com

Abstract

Contemporarily, owing to astronomical advancements in the very large scale integration (VLSI) market segments, hardware engineers are now focusing on how to develop their new digital system designs in programmable languages like very high speed integrated circuit hardware *description language (VHDL) and Verilog and consequently target it to Complex programmable logic devices (CPLDs) and Field programmable gate arrays (FPGAs). If this is properly implemented, it will reduce bulkiness of most of the presently used electronic and electrical devices in our technology This paper will focus on using VHDL to design an application specific integrated circuit (ASIC) liquid dispenser controller system while targeting the device independent architecture (Ultra 3700 CPLD series) for synthesis, optimization and fitting to realize the design. The ASIC controller will have two bin cans to dispense regular and diet drinks. The system will dispense a drink if the user activates a button for that drink and at least one can is available. A refill signal appears when both bins are empty. Activating a reset signal informs the system that the machine has been refilled and the bins are full. The design methodology is presented with other details in the body of this paper.*

Key words: Very high speed integrated circuit hardware descriptive language (VHDL), Application Specific Integrated Circuit, Synthesis (ASIC), Complex programmable logic devices (CPLDs), field programmable gate arrays (FPGAs),]

Introduction

The VHSIC Hardware Descriptive Language (Robert, 2005) has gained wide acceptance as a tool for hardware design and synthesis. VHDL is frequently used for two different goals: simulation of electronic designs and synthesis of such designs.

Synthesis is a process where a VHDL is compiled and mapped into an implementation technology such as an FPGA or an ASIC (Nasri, 2009). An ASIC is a custom- designed fixed-function device; an example of an ASIC would be a specially designed chip for a fax/modem (Robert, 2005);. The key advantage of VHDL, when used for systems design, is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires).

Another benefit is that VHDL allows the description of a concurrent system. VHDL is a dataflow language, unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially, one instruction at a time. There are three levels of abstraction in VHDL description as depicted in figure 1 below.

VHDL project is multipurpose and portable. Being created once, a calculation block can be used in many other projects. However, many formational and functional block parameters can be tuned (capacity parameters, memory size, element base, block composition and interconnection structure). Being created for one element base, a computing device project can be ported on another element base, for example VLSI with various technologies (Wikipedia, 2011).

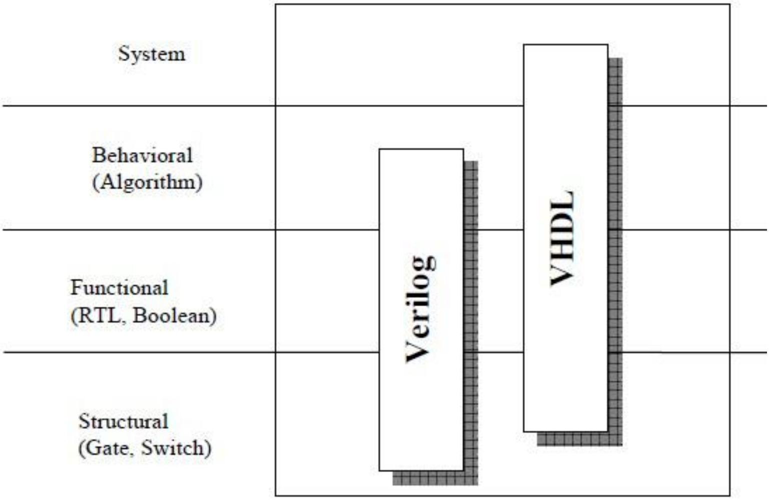


Figure 1: Levels of abstraction. **(Source:** (Nasri , 2009 **)**

Design process framework (flowchart design)

Modern design of logic circuits depends heavily on Computer aided design tools (). Computer aided design tools in the VLSI market segments are produced by CAD vendors like Altera, Cadence, Cypress, Mentor Graphics, Synopsys, Synplicity, and Xilinx. However, this paper developed the liquid dispenser controller (LDC) with the Cypress Warp EDA tool. After the design description in the EDA environment, the CAD tool performs the automatic tasks of optimizing the design logic circuitry to meet the objectives. Synthesis tools support VHDL, although some problems can arise with its use for synthesis (Michel and Alan, 1991). Difficulties such as modeling low-level devices, understanding if timing information is part of specification or if it belongs to the simulation model, defining the equivalence between logic levels and electrical levels are some of the problems. Thus, synthesizing a circuit from a VHDL description is possible, but some limitations are imposed to that description.

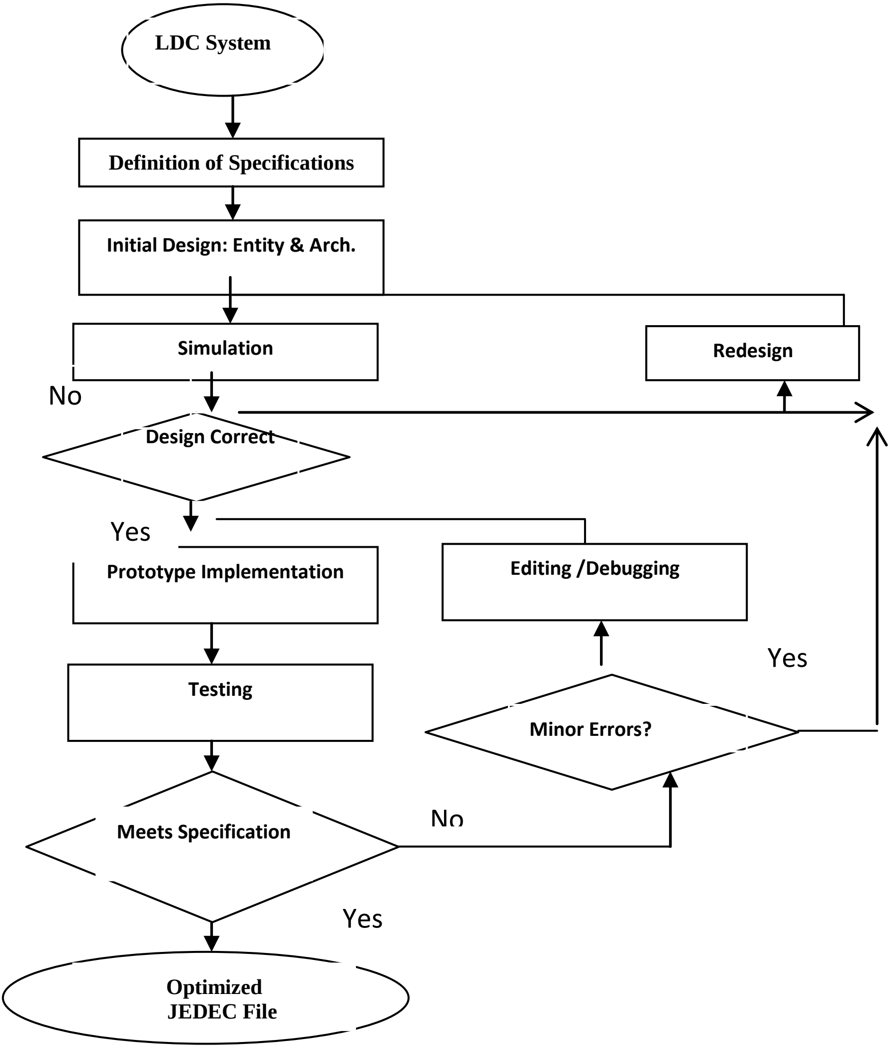
The availability of computer-based tools has greatly influenced the design process in various design environments (Stephen and Zvonko 2005). The flowchart in figure 2 depicts a framework for the design developmental process adopted in this work. Basically, it is the intension of the authors to develop an ASIC customized controller for a liquid dispenser system since the design can be optimized for a specific task; hence leading to better performance.

The process begins with the definition of the liquid dispenser controller system specifications. Then the general structure of the initial design is developed. The Cypress CAD tool (WARP) was now used for simulation of the behavioral description. Any error is rectified before generating the prototype for a complete successful design.

Device Independent Design Concept

Since VHDL supports multiple methods of design description, with one description for the liquid dispenser controller (LDC), a targeted device architecture (CPLD Ultra 37256-CY37256P208-154 NC) was selected to optimize the LDC system for resource utilization. The same VHDL code (design) can be used with any synthesis tool (compiler) and targeted to any vendor device (PLD, CPLD, or FPGA). Hence, with VHDL, device-independent design and portability allows for benchmarking a design using different device architectures and different synthesis tool. This however, facilitates speedy design process and guarantees quick time-to-market at low cost.

Figure 2: LDC Design Process Framework



Complex Programmable Logic Devices (CPLDS)

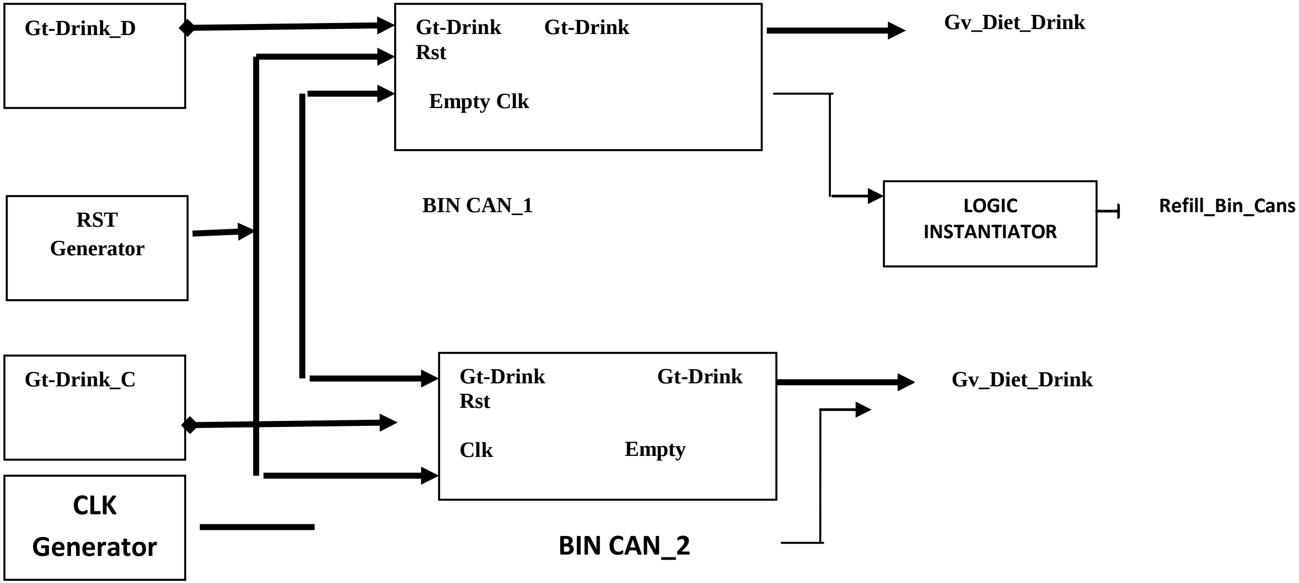
The CPLD architecture, (Ultra 37256-CY37256P208-154 NC) was used in this work. It is In-system Reprogrammable (TM), with 256 microcells, 154MHz maximum frequency and a Temperature range 0^0C to $+70^0$C on a plastic Quad Flat package (PQFP) . Because they offer high speeds and a range of capacities, CPLDs are useful for a very wide assortment of applications, from implementing random glue logic to prototyping small gate arrays. One of the most common uses in industry at this time, and a strong reason for the large growth of the CPLD market, is the conversion of designs that consist of multiple SPLDs into a smaller number of CPLDs.

CPLDs can realize reasonably complex designs, such as graphics controller, LAN controllers, UARTs, cache control, and many others. As a general rule-of-thumb, circuits that can exploit wide AND/OR gates, and do not need a very large number of flip-flops are good candidates for implementation in CPLDs. A significant advantage of CPLDs is that they provide simple design changes through re-programming (all commercial CPLD products are re-programmable). Within system programmable CPLDs it is even possible to re-configure hardware (an example might be to change a protocol for a communications circuit) without power-down.

Liquid Dispenser Controller (LDC) Model

This model adopts a step-by-step approach of using a low-level VHDL behavioral description and a high-level VHDL file to instantiate the component contained in the low-level VHDL file. **Figure 3** shows the LDC model implemented in this work.

Figure 3: A VHDL LDC Model



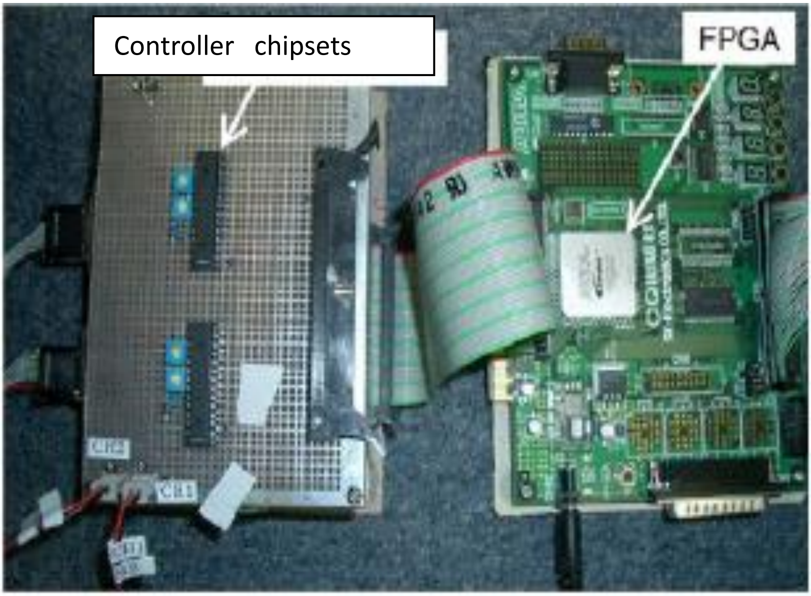


Figure 4: System Development Board for Cypress CPLD.

Discussion

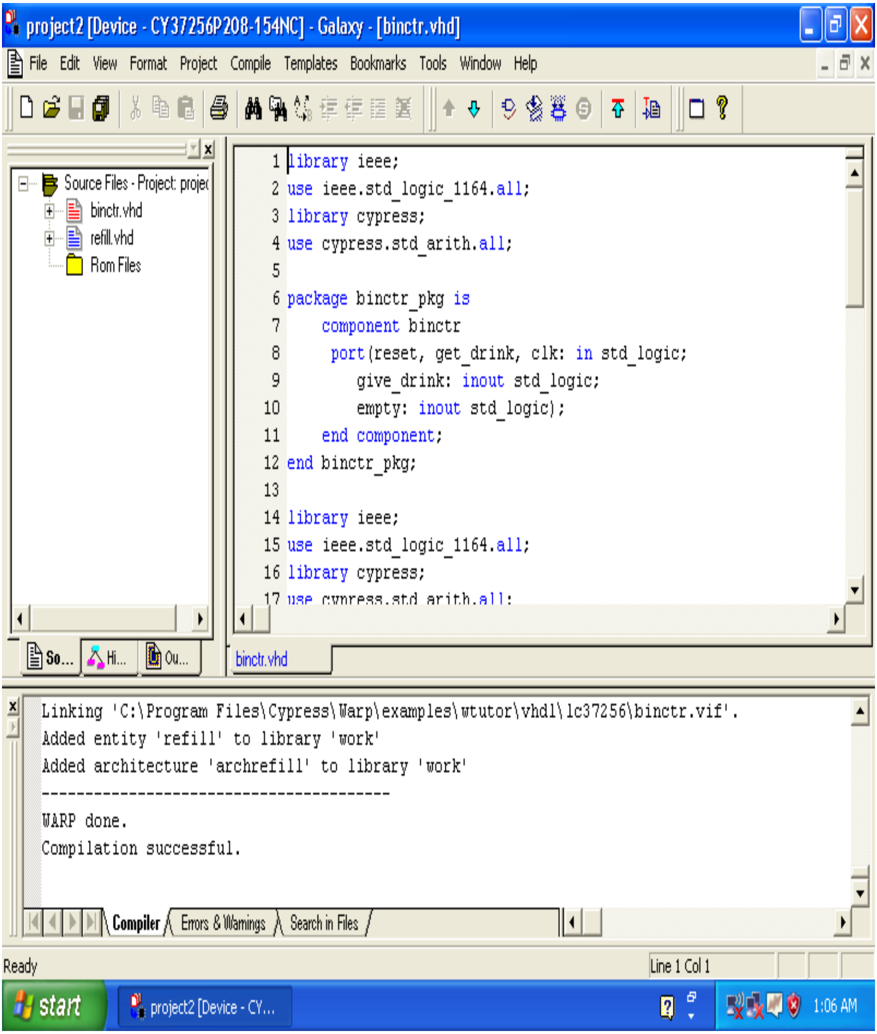
In this research, a VHDL LDC model is presented in figure 3 above. The controller comprises of two bin cans to dispense two drink brands (diet and coke). Each bin can holds three cans of drinks or more. The system dispenses a drink if the user activates a button for that drink. When both bin cans are empty, a refill signal appears. Pressing a reset signal tells the circuit that the machine has recycled and the bin cans are full again. The programming and modeling of the top level instantiation with the low level bin cans component explores completely the full capability of VHDL tool.

Programming Metrics and Results

The device-independent architecture used for this work is CPLD Ultra 37256-CY37256P208-154 NC as stated earlier and programmed with VHDL. Synthesizing the VHDL description for the LDC system facilitates the development of the ASIC version of the system. For the ASIC LDC system, the programming methodology used includes:

1. Writing the VHDL code for Entity, Architecture, Component and package declarations for the LDC system behavioral description.
2. Writing VHDL Entity declaration and Architecture that instantiates the two bin cans into a top level VHDL file via the logic Instantiator.
3. Running WARP (Wikipedia, 2011) on windows 7 platform to compile and synthesize the LDC system design description.

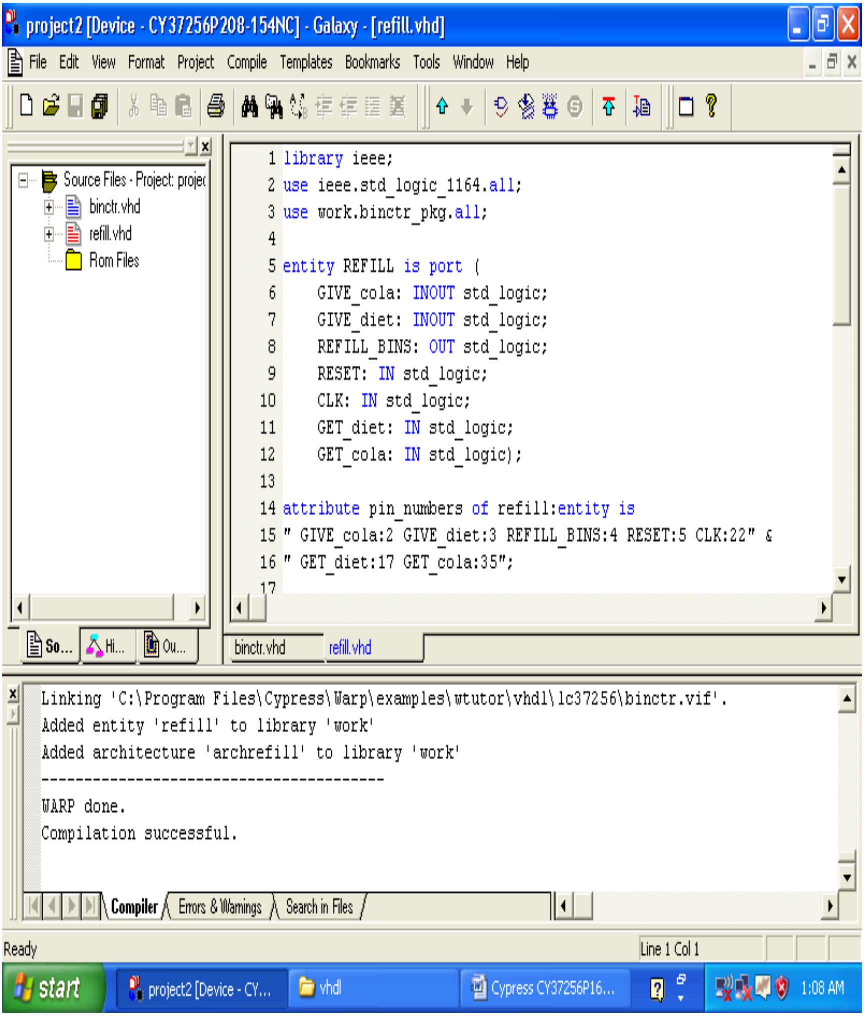
Essentially, there are two major bin can circuits for VHDL description. These circuits were declared as components and their corresponding packages defined. Consequently, a top level description that instantiates the two bin can circuits was implemented via the logic instantiator. After that, the low level bin can circuits and the high level logic instantiator were compiled and synthesized into the device independent Ultra CY37256 JEDEC file. Figures 4, 5, 6, 7, 8 show the results of synthesis and optimization processes for the LDC system in WARP. Warp is a VHDL low cost development system for CPLD by Cypress Semiconductor Corporation. Warp contains an interactive simulator (Aldec) and a compiler (Galaxy).


project2 [Device - CY37256P208-154NC] - Galaxy - [binctr.vhd]
File Edit View Format Project Compile Templates Bookmarks Tools Window Help
Source Files - Project: projec
binctr.vhd
refill.vhd
Rom Files
So...
Hi...
Ou...
binctr.vhd
Compiler
Errors & Warnings
Search in Files
Ready
Line 1 Col 1
start
project2 [Device - CY...
1:06 AM

```
library ieee;
use ieee.std_logic_1164.all;
library cypress;
use cypress.std_arith.all;

package binctr_pkg is
    component binctr
     port(reset, get_drink, clk: in std_logic;
        give_drink: inout std_logic;
        empty: inout std_logic);
    end component;
end binctr_pkg;

library ieee;
use ieee.std_logic_1164.all;
library cypress;
use cypress.std_arith.all;
```

```
Linking 'C:\Program Files\Cypress\Warp\examples\wtutor\vhdl\lc37256\binctr.vif'.
Added entity 'refill' to library 'work'
Added architecture 'archrefill' to library 'work'
----------------------------------------
WARP done.
Compilation successful.
```

Figure 4: Low level BIN_CAN.vhd Synthesis code, (JEDEC File Generated for CPLD).



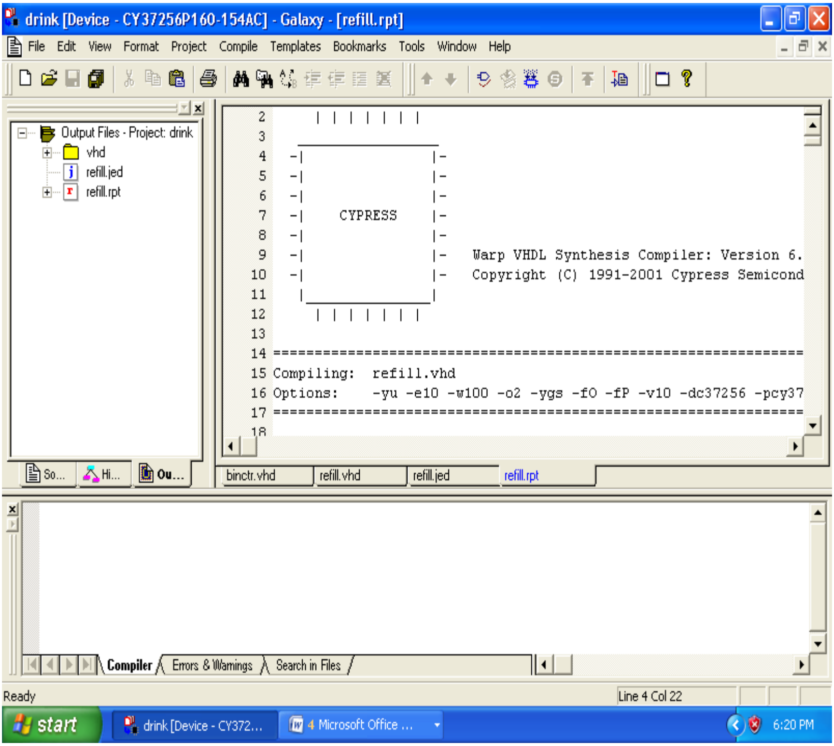
```
library ieee;
use ieee.std_logic_1164.all;
use work.binctr_pkg.all;

entity REFILL is port (
    GIVE_cola: INOUT std_logic;
    GIVE_diet: INOUT std_logic;
    REFILL_BINS: OUT std_logic;
    RESET: IN std_logic;
    CLK: IN std_logic;
    GET_diet: IN std_logic;
    GET_cola: IN std_logic);

attribute pin_numbers of refill:entity is
" GIVE_cola:2 GIVE_diet:3 REFILL_BINS:4 RESET:5 CLK:22" &
" GET_diet:17 GET_cola:35";
```

```
Linking 'C:\Program Files\Cypress\Warp\examples\wtutor\vhdl\lc37256\binctr.vif'.
Added entity 'refill' to library 'work'
Added architecture 'archrefill' to library 'work'
----------------------------------------
WARP done.
Compilation successful.
```

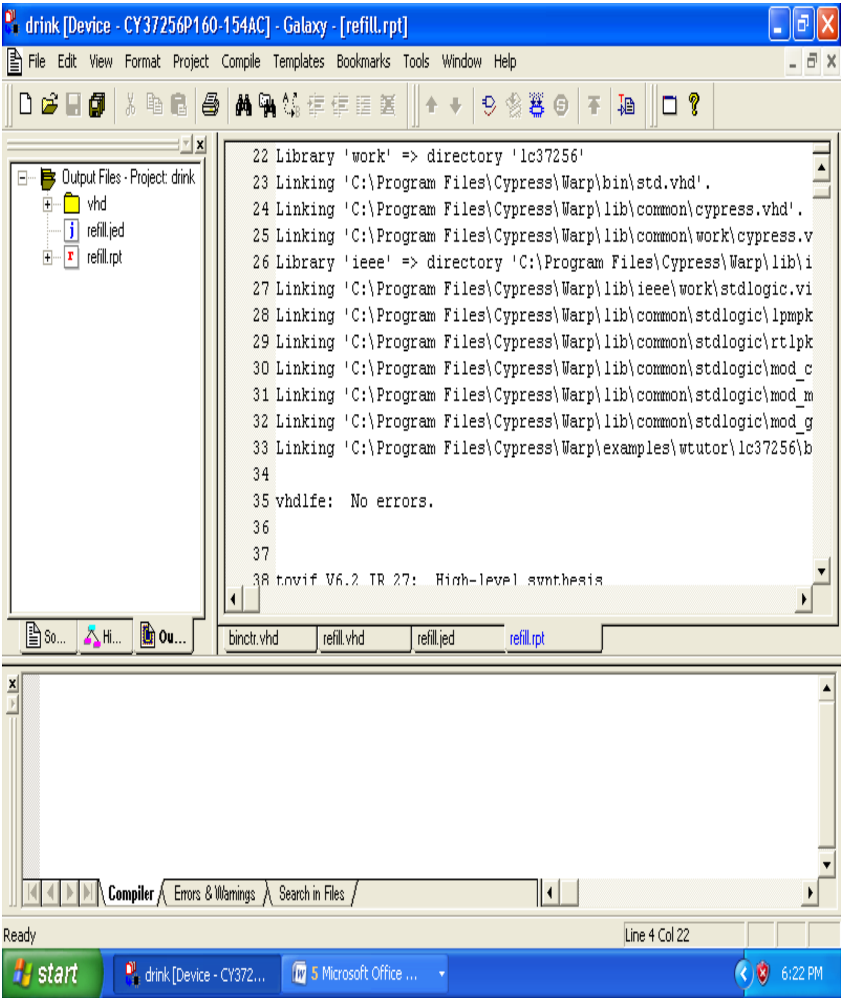
Figure 5: High level REFILL.vhd Synthesis Code, (Jedec File Generated for CPLD)

Following the programming methodology explained above, Figure 4 shows a successful compilation of the low level BIN_CAN description. In this regard, a description of a bin can circuit in IEEE standard VHDL code that controls the operation of bin cans as components with predefined packages is made before the top level instantiation for the LDC system.

Following the programming methodology explained above, Figure 5 shows a successful compilation of the high level BIN_CAN description. Essentially, a description of a top level circuit in IEEE standard VHDL code that instantiates the two bin cans for the LDC system was realized while figure 6 shows the linking process and report file generation from the logic instantiator.

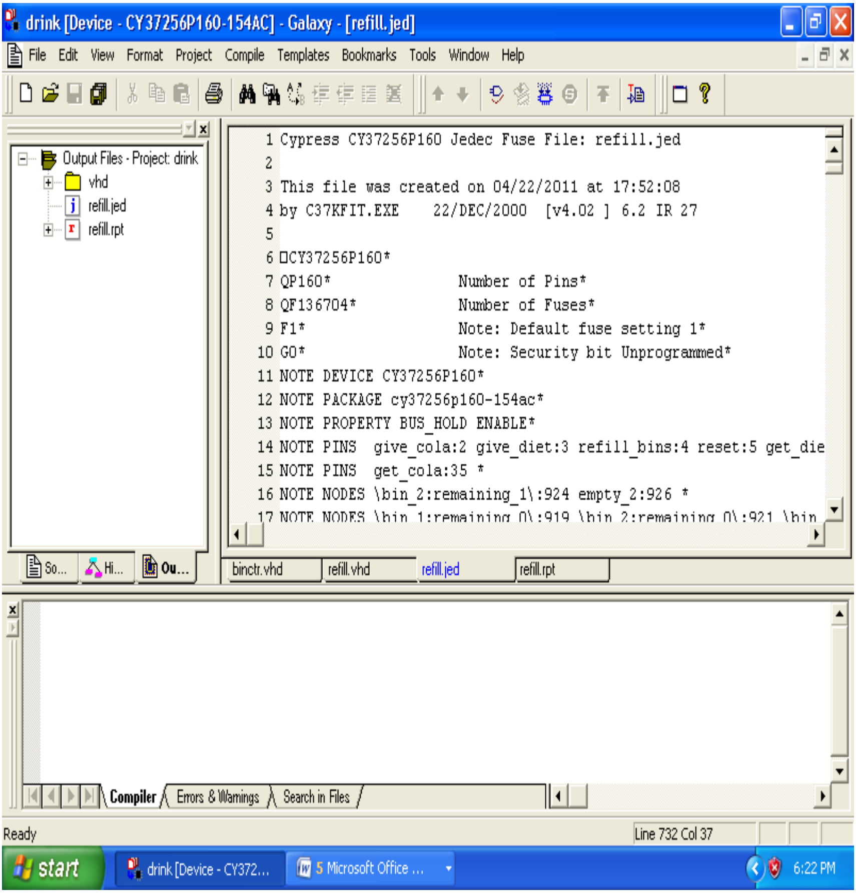


```
      | | | | | | |
   __________________
  -|                 |-
  -|                 |-
  -|                 |-
  -|    CYPRESS      |-
  -|                 |-
  -|                 |-   Warp VHDL Synthesis Compiler: Version 6.
  -|                 |-   Copyright (C) 1991-2001 Cypress Semicond
   |_________________|
      | | | | | | |

================================================================
Compiling:  refill.vhd
Options:    -yu -e10 -w100 -o2 -ygs -f0 -fP -v10 -dc37256 -pcy37
================================================================
```



```
Library 'work' => directory 'lc37256'
Linking 'C:\Program Files\Cypress\Warp\bin\std.vhd'.
Linking 'C:\Program Files\Cypress\Warp\lib\common\cypress.vhd'.
Linking 'C:\Program Files\Cypress\Warp\lib\common\work\cypress.v
Library 'ieee' => directory 'C:\Program Files\Cypress\Warp\lib\i
Linking 'C:\Program Files\Cypress\Warp\lib\ieee\work\stdlogic.vi
Linking 'C:\Program Files\Cypress\Warp\lib\common\stdlogic\lpmpk
Linking 'C:\Program Files\Cypress\Warp\lib\common\stdlogic\rtlpk
Linking 'C:\Program Files\Cypress\Warp\lib\common\stdlogic\mod_c
Linking 'C:\Program Files\Cypress\Warp\lib\common\stdlogic\mod_m
Linking 'C:\Program Files\Cypress\Warp\lib\common\stdlogic\mod_g
Linking 'C:\Program Files\Cypress\Warp\examples\wtutor\lc37256\b

vhdlfe:  No errors.


tovif V6.2 IR 27:  High-level synthesis
```

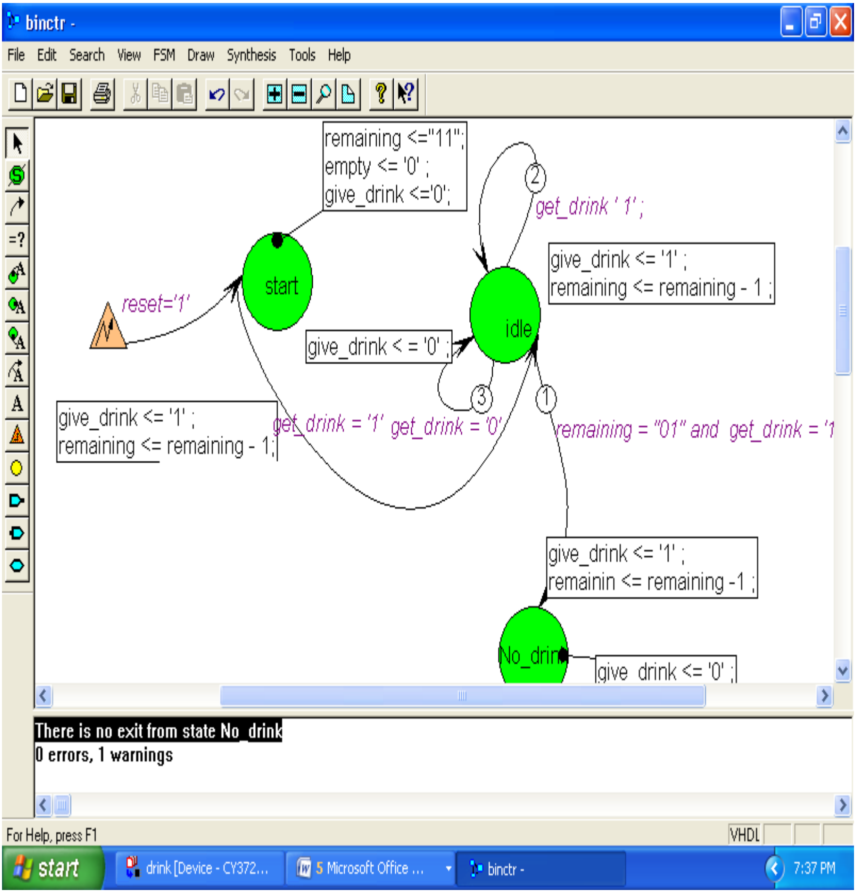
Figure 6: Compilation and Optimization report file for Top level Instantiator



```
Cypress CY37256P160 Jedec Fuse File: refill.jed

This file was created on 04/22/2011 at 17:52:08
by C37KFIT.EXE    22/DEC/2000  [v4.02 ] 6.2 IR 27

□CY37256P160*
QP160*                    Number of Pins*
QF136704*                 Number of Fuses*
F1*                       Note: Default fuse setting 1*
G0*                       Note: Security bit Unprogrammed*
NOTE DEVICE CY37256P160*
NOTE PACKAGE cy37256p160-154ac*
NOTE PROPERTY BUS_HOLD ENABLE*
NOTE PINS  give_cola:2 give_diet:3 refill_bins:4 reset:5 get_die
NOTE PINS  get_cola:35 *
NOTE NODES \bin_2:remaining_1\:924 empty_2:926 *
NOTE NODES \bin_1:remaining_0\:919 \bin_2:remaining_0\:921 \bin
```

Figure 7: Cypress CY37256P160 Jedec Fuse File: refill.jed

Figure 7 shows the output jedec fuse file to programmed into the device Cypress CY37256P160.The compilation, synthesis, optimization and fitting must successfully execute before the fuse file is generated as shown above.



```
There is no exit from state No_drink
0 errors, 1 warnings
```

Figure 8: Finite State Machine for the LDC system

Figure 8 shows the FSM realized with Active-HDL FSM while in Galaxy text editor of Cypress WARP. By selecting Tools option-Active-HDL FSM, a new project is created via the State Editor dialog box. Using the design wizard, the final state machine was created for the LDC system as depicted above.

Conclusion

This paper has presented HDL approach as an efficient design technique for an ASIC LDC system considering the very large scale integration (VLSI) market segments. It outlined an efficient design framework for the proposed system, (LDC system). Unlike the traditional digital system development approaches, this technique offers several advantages to the VLSI market segments particularly quick time to market, ASIC migration, vendor independent compilation and synthesis among others. This is the future of digital revolution for hardware modeling of complex digital systems. Future work will focus on the time diagram of the LDC system and creation of test benches.

References

Nasri S. (2009). "Design and Implementation of FPGA-Based Systems - A Review": *Australian Journal of Basic and Applied Sciences, 3(4):* 3575-3596,

Robert K. Dueck (2005). "Digital design with CPLD applications and VHDL" NY

Stephen B. and Zvonko V., (2005). "Fundamentals of Digital Logic with VHDL Design", pg 6, 2nd Edition,

Wikipedia (2011). the free encyclopedia, [online] http:en.wikipedia.org/wiki/Warp (Cypress).