HDL Windows Designer Compatible with the PALASM Platform

ALIN BADEA, LORAND BOGDANFFY
Department of Control Engineering, Computers, Electrical and Power Engineering
University of Petrosani
20 University Street, 332006, Petrosani
ROMANIA
ghost_0100@yahoo.com, lorand1985@yahoo.com, www.upet.ro

Abstract:- This paper presents a high level view of circuit design and how it can be achieved using hardware description languages as well as the different levels of abstractions that are available to the designer using one of these languages. The paper also presents the characteristicss that describe a PAL circuit. A short description of PALASM HDL is presented and then the paper goes on to describe how a PALASM Windows designer was designed and implemented using the OOP paradigm and the C# 2010 language. Finally the paper presents an example of using the designed application to implement a left-right drive control for a motor assembly.

Key-Words:- HDL, PALASM, Windows Designer, PAL, PLD, WPF, C#

1 Introduction

Circuit design using HDL is achieved by creating a textual description of a circuit or logic digital system. Systems and hardware circuits are described at different levels of abstraction using HDL. Starting at the highest level using the concept of system or conceptually representing a high level description. The algorithm level describes the behavior of the project in mathematical terms.

Neither the concept of system nor the algorithm describe how the project behavior will be implemented. The algorithm structure in the hardware is described by architecture. The architecture identifies the high level functional blocks that will be used and the interconnections between these blocks. The algorithm level and the architecture describe the behavior of the project well enough for it to be verified and simulated.

The next level of the architecture is the register transfer level. This level describes the register storage and the transfer of data in the project along with the logic operations on the data. This level is used mostly by synthesis tools that describe the structure of the project. Logic gates are implemented using transistors. The HDL can offer a comutation description level that models the transistor like a switch. When working with HDLs, the designer chooses the language and the desired abstraction level. When choosing an HDL the following things need to be considered;

- The availability of process automation tools to offer support for the selected language;
- Prior knowledge;
- Personal preferences;
- Availability of simulation tools;
- Sinthesys capabilities;
- Reusing the design;
- The need to learn the language and its capabilities;
- The existence of standards for the language;
- Access to these standards;
- The code produces is easy to read

Because programming PLD circuits is relatively easy to do the PALASM language was designed to desccribe the PAL circuits, create the AND matrix connections, simulate the circuit and in the end program them.
Each PAL circuit is described using the following parameters:

- Number of input pins;
- Number of output pins;
- Number of I/O pins;
- The number of product terms of each output;
- Number of flip-flops;
- Active logic.

PALASM is a hardware description language used to translate logic functions and state transition tables to fuse maps in order to program PAL devices. The last version of this software platform ran under the DOS operating system and was developed by Advanced Micro Devices.

2 Problem formulation

PALASM is a HDL used to program PAL devices. These contain a programable AND gate input matrix and an OR gate output matrix. There exists a large variety of PAL devices designed for implementing relatively simple logic functions with combinatorial and sequential inputs and outputs.

Beeing available only under the DOS operating system the application has many inconvenients when running on modern computers.

The goal of this paper is to present the design and implementation of a software platform that runs on the Windows operating system and is based on the DOS version of the PALASM platform created by AMD. The paper also presents a practical application implemented using this software platform.

3 Problem solution

In order to solve some of the problems that the old PALASM software has when running on modern computers this section proposes a new application that is designed to run under the Windows operating system.

This application is designed to be compatible with the older PALASM version that runs on the DOS operating system and offers a user friendly interface that is easier to use.

3.1 Configuration file

The application is designed and implemented using the OOP paradigm in the C# language and has a friendly user interface built using WPF. The GUI uses the PALASM software compiler developed by AMD and a custom configuration file.

The configuration file contains compilation settings, simulation settings, logic synthesis settings and data regarding the current project beeing developed.

All of these are stored using custom configuration sections. When the PALASM compiler and simulator are run the settings in the configuration file are passed to them as parameters.

The configuration settings are stored in three classes. These classes are necesary in order to use the custom configuration sections inside the App.config file of the application.

These settings are read from the XML file using APIs provided by the .NET platform. The class diagrams for these clases are presented next.

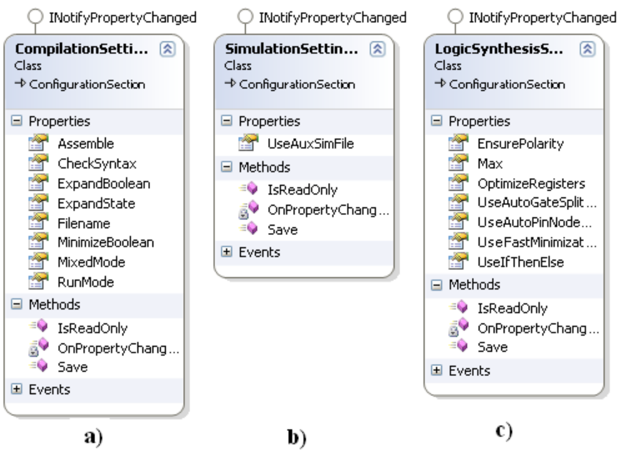
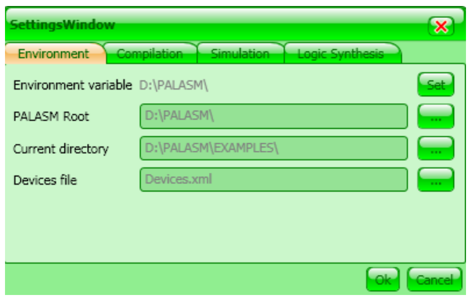


Fig.1 Configuration sections a) Compilation b) Simulation c)Logic synthesis

These settings can be accessed and changed in the application using the settings window. The GUI for this window is presented next.



Environment variable	D:\PALASM\
PALASM Root	D:\PALASM\
Current directory	D:\PALASM\EXAMPLES\
Devices file	Devices.xml

Fig.2 Settings window

As can be seen from the above figure this window also allows the user to also change environment settings. In order to read the settings from the configuration file the custom configuration section classes are used.

The structure of the configuration file is presented next.

```
<myConfigSettings>
  <compilationSettings runMode="Auto"
                       filename="palasm2.log"
                       checkSyntax="true"
                       expandBoolean="true"
                       expandState="true"
                       mixedMode="true"
                       minimizeBoolean="false"
                       assemble="true"/>
  <simulationSettings useAuxSimFile="false" />
  <logicSynthesisSettings max="4" useAutoPinNodePairing="true"
                          useAutoGateSplitting="false"
                          useFastMinimization="false"
                          optimizeRegisters="AsSpecifiedInDesignFile"
                          ensurePolarity="BestForDevice"
                          useIfThenElse="DontCare"/>
</myConfigSettings>
```

3.2 General project data

The information regarding the current project are stored in the ProjectData class. The structure of this class is presented next.

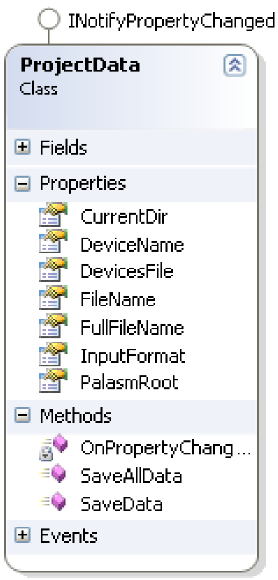


Fig.3 ProjectData class

As can be seen from the above figure this class stores information about the current working directory, the device name, the project file name and the file that holds a list of all available PAL and GAL devices. Some of this information (DeviceName, FileName, InputFormat) is set when a new project is created and cannot be changed after that. The rest of the information can be changed using the Environment tab of the settings window.

3.3 Pin and node information

When a new project is created the information regarding pin and node settings are stored in the PinData class. The structure of this class is presented next.

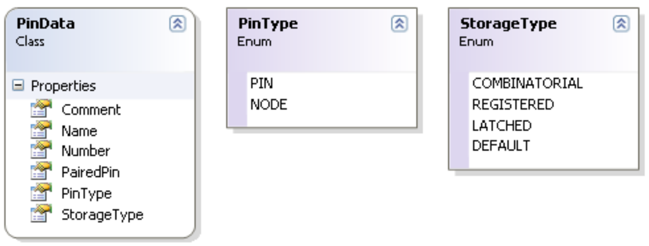
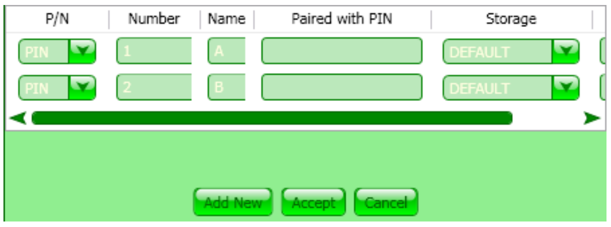


Fig.4 PinData class

This class exposes some properties that allow a pin or node declaration to be changed. The PinType property is used to declare either a pin or a node. The StorageType property can be used to define the pin or node as a registered, combinatorial, latched or default (combinatorial). In order to declare a new pin or node when the NewProject window is open the user can press the AddNew button. When this button is pressed a new pin having a default configuration will be added to the list of pins. Pin settings can be changed by entering the new data in the available textboxes of by selecting available values from the comboboxes. The GUI of the NewProject window is presented next.



P/N	Number	Name	Paired with PIN	Storage
PIN	1	A		DEFAULT
PIN	2	B		DEFAULT

Fig.5 NewProject window

3.4 The Controller class

This class cotains properties that provide access to the application settings as well as some methods that are used to compile and simulate the current project. The structure of this class is presented next.

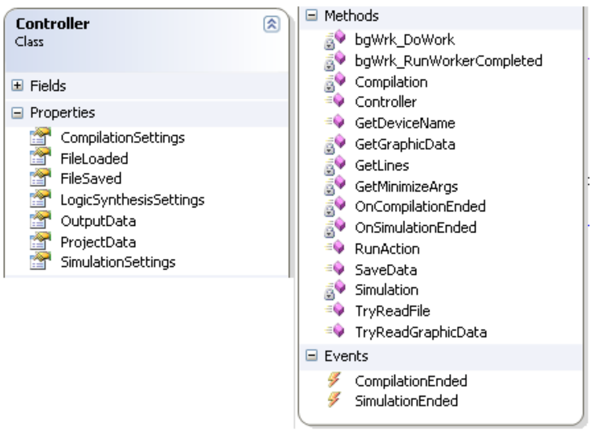


Fig.6 The Controller class

The compilation and simulation of the current project is done using the Compilation and Simulation methods. These methods are called in a separate thread using the BackgroundWorker component available in C# in orded not to cause the UI to block. The code that starts the compilation and simulation processes is presented in the following snippet.

```
void bgWrk_DoWork(object sender, DoWorkEventArgs e)
{
    if (action == Action.Compilation)
        Compilation();
    else
        Simulation();
}
```

When the compilation or the simulation is complete the following code is executed which fires one of two events.

```
if (action == Action.Compilation)
    OnCompilationEnded();
else
    OnSimulationEnded();
```

The CompilationEnded and SimulationEnded events are used to display the results of the compilation and simulation processes. The compilation and simulation of the current project is achieved using the PALASM compiler and simulator created by AMD which are available in the DOS version.

In order to start the compiler and the simulator the Process class (available in the .NET platform) is used as follows.

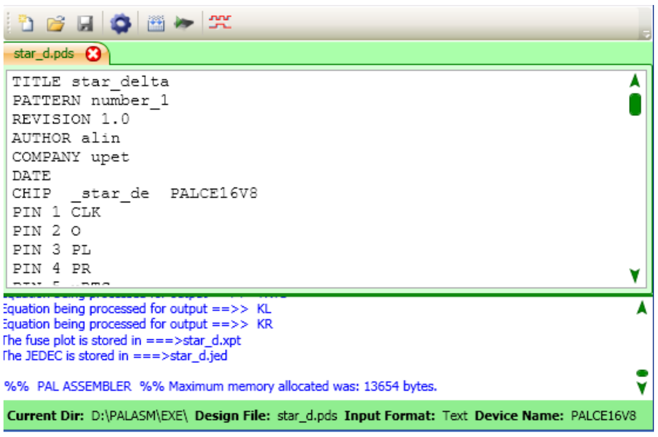
- Create an instance of the Process class
- Initialize this class by setting the StartInfo property

  For StartInfo we set the following properties
  - The name of the compiler file, using the FileName property
  - The compiler parameters using the Arguments property
  - The Directory from which the file will launched using the WorkingDirectory property
- The Start method of the Process class is called
- The WaitForExit method is called in order to wait for the current process to complet before starting a new process
- Close the process.

The compilation and simulation processes create an intermediary file and a log file. After each process is complete the log file is read and displayed. This file contains the results of the compilation or simulation and may contain errors.

3.5 The G.U.I.

The designed application is used to create, edit, compile and simulate PALASM design files. The main application window is divided into 2 sections: an editor section at the top and an output section at the bottom that displays log messages from the compiler and simulator. The main window also has a status bar with details about the current project and a toolbar that contains common menu shortcuts. This window is presented in the following figure. The parser created by AMD was used.



```
TITLE star_delta
PATTERN number_1
REVISION 1.0
AUTHOR alin
COMPANY upet
DATE
CHIP  _star_de  PALCE16V8
PIN 1 CLK
PIN 2 O
PIN 3 PL
PIN 4 PR
```

```
Equation being processed for output ==>>  KL
Equation being processed for output ==>>  KR
The fuse plot is stored in ===>star_d.xpt
The JEDEC is stored in ===>star_d.jed

%%  PAL ASSEMBLER  %% Maximum memory allocated was: 13654 bytes.
```

Current Dir: D:\PALASM\EXE\ **Design File:** star_d.pds **Input Format:** Text **Device Name:** PALCE16V8

Fig.7 Application main window

As can be seen from teh above figure the main window of the application is divided into two sections. The upper section is used to edit the current project file and the bottom section is used to display the results of the compilation and simulation processes. The botttom section also contains details about the current project.

The application can open multiple files related to the current project where each open file will appear in its own tab. When the user opens a file other thatn the current design by using the View menu options the content of this file will be displayed in a new tab as can be seen from the following figure.

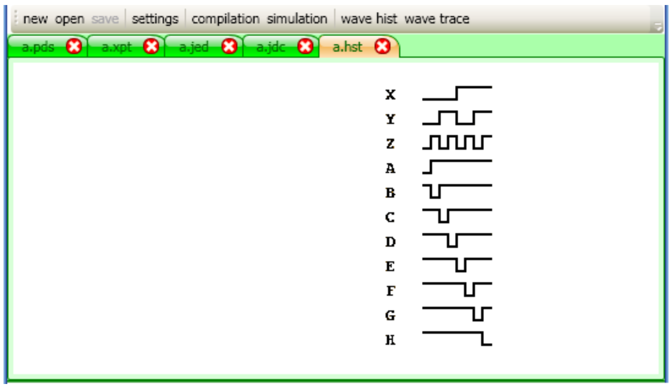


Fig.8 Simulation process results

In order to create a new design the New option of the File menu is used. The NewProject window that is shown allows the user to specify initial project data like the chip that is used, the list of pins and nodes and others.

3.6 Practical application

In order to demonstrate the usefulness and ease of use of the designed platform a controller was implemented that allows reversible starting of a motor assembly.

The diagram of the controller for the reversible starting is presented in the following figure:

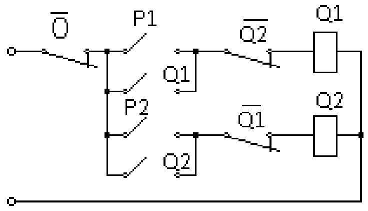
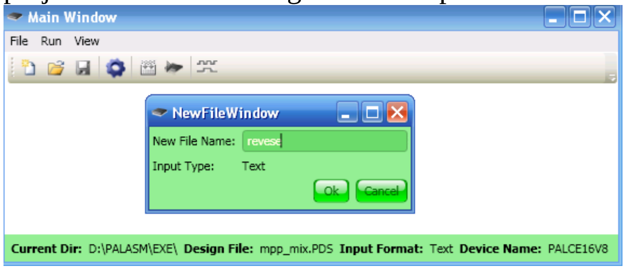


Fig.9 controller diagram

Based on the above diagram the logic functions are determined.

$$Q_1 = \bar{O} * (P_1 + Q_1) * \overline{Q_2} \qquad (1)$$
$$Q_2 = \bar{O} * (P_2 + Q_2) * \overline{Q_1} \qquad (2)$$

The logic functions were then programmed into a PAL device using the designed application. A new project was created using the menu options.

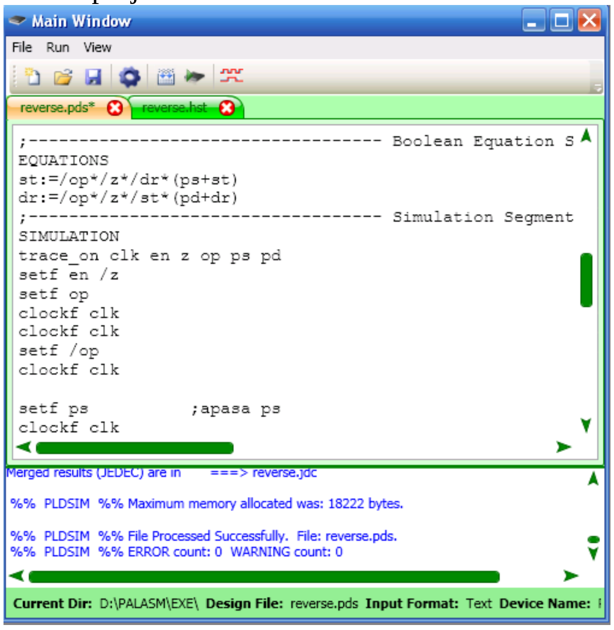


New File Name: revese
Input Type: Text

Current Dir: D:\PALASM\EXE\ Design File: mpp_mix.PDS Input Format: Text Device Name: PALCE16V8

Fig. 10 Create new project

The logic eqquations were them programmed into the project.



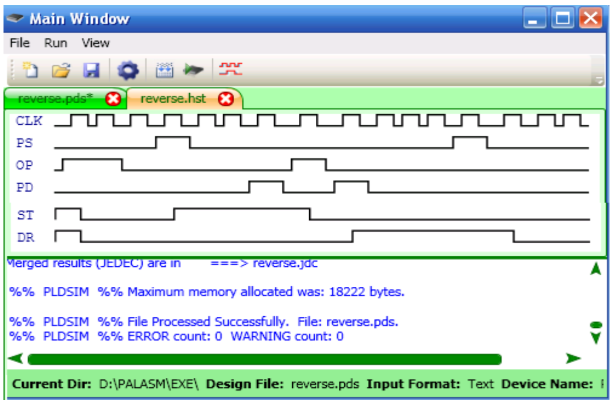
```
;----------------------------------- Boolean Equation S
EQUATIONS
st:=/op*/z*/dr*(ps+st)
dr:=/op*/z*/st*(pd+dr)
;----------------------------------- Simulation Segment
SIMULATION
trace_on clk en z op ps pd
setf en /z
setf op
clockf clk
clockf clk
setf /op
clockf clk

setf ps           ;apasa ps
clockf clk
```

```
Merged results (JEDEC) are in      ===> reverse.jdc

%% PLDSIM %% Maximum memory allocated was: 18222 bytes.

%% PLDSIM %% File Processed Successfully.  File: reverse.pds.
%% PLDSIM %% ERROR count: 0  WARNING count: 0
```

Current Dir: D:\PALASM\EXE\ Design File: reverse.pds Input Format: Text Device Name:

Fig.11 Code window

The project was then compiled and simulated. The results of the simulation are shown in the following figure.



```
Merged results (JEDEC) are in      ===> reverse.jdc

%% PLDSIM %% Maximum memory allocated was: 18222 bytes.

%% PLDSIM %% File Processed Successfully.  File: reverse.pds.
%% PLDSIM %% ERROR count: 0  WARNING count: 0
```

Current Dir: D:\PALASM\EXE\ Design File: reverse.pds Input Format: Text Device Name:

Fig.12 Simulation results

4 Conclusion

Even though the PALASM language is not used as much as it was in the early 90s it is still used to program simple logic functions into PAL devices. The software platform that was presented in this paper runs under the Windows operating systems offering a user friendly and intuitive interface with relatively few menu options. In case of errors durring the simulation and compilation processes the bad lines of code can be easily identified since the application displayes both the code window and the log window on the screen at the same time. Also since the application can open multiple files, related project files can be viewed more easily.

References:

[1] *** PALASM User Guide.

[2] Pop E., *Automatizari in industria miniera*. Editura Didactica si Pedagogica, Bucuresti, 1983

[3] Torelsen A., *Pro C# 2010 and the .NET 4 Platform 5th* ed., Apress, 2010.

[4]B. Cohen, *VHDL Coding Styles and Methodologies 2nd* ed. Kluwer Academic Publishers, 2002.

[5] F. Badea, E. Pop, Approache to Designing the Graphic User Interface Components for Mobile SCADA HMIs and Application, *Proceedings of the 11th WSEAS International Conference DNCOCO'12, Sliema, Malta, ISBN: 978-1-61804-118-0, 2012.*

[6] O.Hachour, N.Mastorakis, IAV: A VHDL Methodology for FPGA implementation, *WSEAS Transactions on Circuits and Systems, 2004.*

[7] A. Avram, E. Pop, C. Barbu, VLSI Embedded Solution for Multi-Drive Conveyors Control, *Proceedings of the International Conference on Applied Computer Science (ACS), WSEAS, Malta, September 15-17, 2010.*