ISSN 2394-7314

International Journal of Novel Research in Computer Science and Software Engineering
Vol. 3, Issue 1, pp: (140-158), Month: January-April 2016, Available at: www.noveltyjournals.com

Review of FPD'S Languages, Compilers, Interpreters and Tools

[1]Amr Rashed, [2]Bedir Yousif, [3]Ahmed Shaban Samra

[1]Higher studies Deanship, Taif university, Taif, Saudi Arabia
[2]Communication and Electronics Department, Faculty of engineering, Kafrelsheikh University, Egypt
[3]Communication and Electronics Department, Faculty of engineering, Mansoura University, Egypt

Abstract: **FPGAs have achieved quick acceptance, spread and growth over the past years because they can be applied to a variety of applications. Some of these applications includes: random logic, bioinformatics, video and image processing, device controllers, communication encoding, modulation, and filtering, limited size systems with RAM blocks, and many more. For example, for video and image processing application it is very difficult and time consuming to use traditional HDL languages, so it's obligatory to search for other efficient, synthesis tools to implement your design. The question is what is the best comparable language or tool to implement desired application. Also this research is very helpful for language developers to know strength points, weakness points, ease of use and efficiency of each tool or language. This research faced many challenges one of them is that there is no complete reference of all FPGA languages and tools, also available references and guides are few and almost not good. Searching for a simple example to learn some of these tools or languages would be a time consuming. This paper represents a review study or guide of almost all PLD's languages, interpreters and tools that can be used for programming, simulating and synthesizing PLD's for analog, digital & mixed signals and systems supported with simple examples. In addition, their features and applications will be discussed. In Addition to the aforementioned summary, the paper presents a new classification for FPGA languages and Tools as well. Interpreters like PERL and MYHHDL (python based) also described here. At the end one can summarize all what's needed to know about FPGA languages, Tools, Compilers, and Interpreters in one research paper.**

Keywords: **HDL languages, C to HDL tools, Labview, Matlab HDL coder.**

1. INTRODUCTION

FPGAs have achieved quick acceptance, spread and growth over the past years because they can be applied to a variety of applications. Some of these applications includes: random logic, bioinformatics, video and image processing, device controllers, communication encoding, modulation, and filtering, limited size systems with RAM blocks, and many more. In these paper FPGA's languages, compilers and tools would be presented. These can be used for programming, simulating, synthesizing and more FPGA, PLA, and CPLD for analog, digital and mixed signals and systems. In addition, their features and applications will be discussed as well. All C based languages will be widely discussed while FPGA, LABVIEW system will be briefly discussed. The importance of each language is declared with simple example and its applications. Some languages does not declared here for two reasons first there is no good references describing it or there is no good and simple example to explain this language. Finally a proposed languages category table is presented.

The paper is structured as follows. Section 1: Introduction. Section 2: present Related Work. Section 3 discusses different Field-Programmable Device (FPD) types and programming techniques. Section 4 explains FPD languages, tools and compilers used for implementation, simulation and synthesis. Section 5 summarizes our observations and discusses potential future extensions of the research.

This paper makes the following contributions (1) Paper represents a collection of almost all FPGA languages and tools by names, (2) Paper represents a brief description of some of these languages and tools, (3) Paper represents new classification for FPGA languages and tools, (4) Paper represents a collection of comparative studies between FPGA languages and tools with respect to simple calibration or evaluation test application. This research is very helpful for language developers to know strength points, weakness points, ease of use and efficiency of each tool or language. The best way to clear describe of a language is to explain its functionality through a simple example. A 4-bit counter is chosen to be mentioned in each language description.

2. RELATED WORK

As mentioned above there is no complete (in my knowledge) reference or guide describes all FPGA languages, Tools, Compilers, Interpreters. Also there is no clear classification of their categories, so this research paper depends on many references each one presents one or more tool. For no or inadequate reference tools or languages we depend on available data in their home pages.

Basic Categories of Hardware description Languages (HDLs) are (1) *High Level Synthesis (HLS)* tools also called behavioral and architectural-level synthesis .HLS tools in general (some of them doesn't generate HDLs) uses the existing programming language like C as an input design language and produce or generate the related HDL language, (2) *Alternative syntax*: Some tools suggest a substitution syntax to VHDL or Verilog code. The substitution or alternative syntax approach keeps the control of the generated hardware, but gives the advantage of an easier to master syntax and sometimes of ease of debugging,(3)*Basic HDL languages*: basic or traditional HDL languages like VHDL, Verilog HDL.

Esam El-ArabyI , and others in 2007[1], characterize a comparative analysis methodology and results depend on empirical study of high level programming for reconfigurable computers . This research is focused on three representative high-level tools Impulse-C, DSPlogic , and Mitrion-C in the Cray XD1 environment. Also paper represents imperative programming, functional programming and graphical programming comparison are made with respect to their performance and ease of use. This research depend on classifying Fpga languages in two categories HDL languages and HLS such as Impulse-C, DSPlogic , and Mitrion-C .Table 1 shows comparison adopted from this research.

Table 1 Discusses Tools Efficiency and Ease-of-Use.

	Average difficulty factor (d_x)	**Normalized average difficulty factor (δ_x)**	**Normalized average ease-of-use factor (λ_x)(%)**	**Average efficiency (E_X)**	**Normalized average efficiency (η_x)**	**efficiency**	**Ease-of -use**
Impulse-C	3.9727×10^{-3}	0.4640	53.60	2.8196×10^{3}	5.72	low	easy
Mitrion-C	7.9112×10^{-3}	0.9240	7.60	3.9122×10^{3}	7.94	low	difficult
DSPlogic	4.0383×10^{-3}	0.4717	52.83	2.2975×10^{4}	46.625	moderate	easy
VHDL	8.5615×10^{-3}	1	0	4.9277×10^{4}	100	High	difficult

Donald G. Bailey presents alternative classification for FPGA languages in his book [2] .this classification is depend on dividing FPGA programming languages for image processing application into three categories (1) Hardware Description Languages,(2) Software-Based Languages ,and (3)Visual Languages .Table 2summarized in table 2.

Table 2.programming languages and tools (second approach)

Basic Classification	Secondary Classification	Examples
1-Hardware Description Languages	--	VHDL, Verilog HDL
2-Software-Based Languages	a-Structural Approaches	JHDL, Quartz, HIDE
	b-Augmented Languages	SystemC, ASC,SA-C, A|RT, Mitrion-C, Handel-C, Handel-C Derivatives
	c-Native Compilation Techniques	Transmogrifier C, SUIF-Based Compilers, Impulse C, Dime-C and DimeTalk , Catapault C, MATLAB Based
3-Visual Languages	a-Behavioral	vVHDL,
	b-Dataflow	Khoros Based, Simulink Based, others (like labview, PixelStreams)
	c-Hybrid	VERTIPH,

Third classification for programming languages and tools presented by Richard Wain ,and others [3] can be summarized in table 3 .

Table 3. programming language and tools (third approach)

Basic classification	Examples
VHDL &Verilog	a-Xilinx tools b-FPGA Advantage Mentor Graphics tools
Pseudo-C High level languages	a-Mitrion-C and Mitrion-IDE b- Handel-C and Celoxica DK Design Suite. c-Nallatech DIME-C and DIMETalk
Other languages and tools	Like : SystemC, Catapult C, Impulse C, Carte, Streams C, AccelChip, Starbridge, NAPA-C, SA-C, CoreFire, Trident compiler, DSPlogic, CHiMPS

For analog mixed signal modelling languages. Design and Reuse web site [4] presents an article[5] titled with Analog and Mixed Signal Modelling Approaches comparative study between basic analog mixed signal approaches VHDL-AMS,VERILOG-A,MATLAB SIMULINK,SPICE and system C-AMS. Also describe Challenges in AMS Design and Modelling. Results taken from this article would be presents later.

Brian Holland, and others presented a Survey of C-based Application Mapping Tools based Application Mapping Tools for Reconfigurable Computing for Reconfigurable Computing[6] .Table 4 discuss five basic categories of C based languages discussed in this survey. Results taken from this study will be described in C to RTL tools in this research.

Table 4. Five Basic Categories of C based Languages

Open Standard	Generic HDL Multiple Platforms	Generic HDL (Optimize for Manufacturer’s Hardware)	Targets a Specific Platform/Configuration	RISC/FPGA Hybrid Only
System C	Mitrion C, Impulse C, Catapult C	Handel C, DIME-C	Streams C, SA-C, Carte	Napa C

Embedded system design with FPGA book [7] in chapter 9 introduced CAPH Language for Implementing Stream-Processing Applications on FPGAs .this research describes drawbacks of C to RTL languages or solutions. Also describes CAPH language capabilities and advantages over traditional VHDL.CAPH language does not represented here .

A Comparative Evaluation study [8] is made between some of Automated HDL Generation tools or languages like SPARK, ROCCC, and DWARV based on a dataset of roughly 140 C functions. The number of functions in each application domain for DWARV and SPARK under consideration is depicted. These numbers represent the number of kernels who's generated VHDL was successfully synthesized. Total number of functions for all application is equal to 158. Also results taken from this research will be discussed later

This research [9]discusses the history of HLS, categorizes it into three time generation and identifies reasons why the current or third generation HLS tools may be successful and proceeded where earlier generations(first and second generations) tools failed. Also research discussed failure causes.

This research [10] presents a comparative study between twelve HLS languages based on number of features like ease of implementation, test-bench generation, verification, and so on. Results will be discussed later.

The Design warrior's guide to FPGAs [11] describes in chapter 9 HDL design flow with comparison between VHDL and VERILOG ,also chapter 11 discussed problems with traditional HDL based flows. Author prefers to divide HLS into three categories: (A) System C based flows; (b) Augmented C/C++ based flows; and (c) Pure C/C++ based flows. Advantages of using mixed language design and verification environments also discussed.

3. S/W AND H/W BASED LANGUAGES

In this section a comparison between hardware based languages (HDLs) and Software based languages(HLS Tools) is made based on number of features like sequential/concurrent processing. Table 5 shows difference between software and hardware based languages [2].

Table 5.difference between S/W and H/W based languages for FPGA

	Hardware based languages(HDLs)	Software based languages(HLS Tools)
processing	Inherently concurrent	Inherently sequential
Word width	Data word width is optimized	Fixed data word width (8,16,32 bits)
Time taken	Time taken for task is depend on timing constraints	Time taken is not considered
Memory management	Memory is divided into a large number of small blocks. Local variables are not implemented in memory.	Software treats memory as single large block, compiler may map local variables to registers
Processes communication	Parallel processes can communicate directly. Communication relies on hardware	It uses shared memory (including mailboxes) to communicate between processes. Processes cannot communicate directly.
Algorithm representation	The algorithm is demonstrated by hardware connection between blocks. Saving current state is difficult. Recursive algorithms are impractical.	An algorithm is demonstrated as a sequence of instructions stored in memory.
Other advantages	Available for high level languages and high capacity FPGA	Support floating point computation

4. FPD LANGUAGES

In this section our proposed categories for FPD languages will be discussed here with simple example. Basic FPD language categories: (1)Early HDL languages,(2)No common use language,(3)Traditional and Widely Used Languages,(4)Verification languages,(5)Control languages,(6)Rarely used and Alternative Languages,(7)Tools,(8)Compilers .

4.1 Early HDL languages

Five languages will be discussed here in brief (A)PALASM, (B)CUPL-HDL, (C)ABEL-HDL, (D)LOLA, (E)AHDL.

4.1. 1. PALASM:

PALASM Stands for Programmable Array Logic Assembler. PALASM is an early hardware description language, used to convert Logic functions and state transition tables to a fuse map for use with programmable array logic (PAL) devices.

PALASM provides more than just a logic description. It also provides, in the function table section, a set of tests to be applied to the device after programming. The language was developed in the early 1980s. It is not case-sensitive language [12].

This was an important language feature because device programming was not always successful. The devices could have faulty programmable fuse elements, the programming equipment could be out of spec, or the logic equations themselves could be incorrect. Providing hardware test data in the form of test vectors right in the design itself eliminated the vast majority of later in-system problems [13].See a PALASM code for 4 bit counter with synchronous clear [14]

4.1.2. CUPL-HDL:

CUPL stands for Common Universal tool for Programmable Logic compiler. Assisted Technology released CUPL in September 1983. It was the first high level language that supported multiple PLD families. First it was written in the C language so it could be transferred to additional platforms.

The CUPL language supports any combination of state machine, truth table and Boolean entry methods .CUPL has powerful features like the ability to compare a set of bits to a fixed binary, decimal and hexadecimal. CUPL is also available as an integrated development package (IDP) for Microsoft Windows.

CUPL includes simulator called CSIM. It uses for verifying logic specification files written in CUPL, it also allows the creation of test vectors which may be down-loaded to a PLD programmer/tester. A PALASM-to-CUPL translator is included to allow easy conversion of existing logic specification files written in PALASM [15].See a CUPL-HDL code for 4 bit counter [16]

4.1.3. ABEL-HDL:

ABEL stands for Advanced Boolean Expression Language. The ABEL concept and basic compiler were created by Data I/O's Applied Research Group in 1981 .Then it was continued by ABEL product development team. ABEL is now owned by Xilinx Inc. ABEL is a Hardware description language and it includes a related set of design tools for programming PLDs.

ABEL includes concurrent Boolean equations and truth table logic formats as well as a sequential state machine description format. A pre-processor with syntax loosely based on DEC's Macro-11 is also included.

The Xilinx ABEL version of the ABEL-HDL compiler is supporting software functionally verifies ABEL-HDL designs through simulation. The compiler then implements the designs in CPLDs [17].See A ABEL-HDL code for 4 bit counter with look ahead carry [18].

4.1.4 LOLA:

Lola was created in 1992.LOLA is a simple and easy to learn hardware description language for describing synchronous, digital circuits ,general hardware designs and particular coprocessor applications [19].

The Lola System is a toolbox containing of modules whose commands serve to specify, implement, and test digital circuits. These notes represent its structure to the user of Lola and to the implementer of additional tools .LSB is the Lola system base module which contains the definition of the central data structure used to describe digital circuits. Typically, such a data structure is produced from a Lola text by the compiler, and thereafter used as argument for next processing steps, such as simplification, analysis, comparison, simulation, and layout generation [20].LOLA homepage site is [21].See LOLA code for N –bit binary counter [22].

4.1.5 AHDL:

AHDL stands for Altera Hardware Description Language. It may also refer to analog hardware description language. AHDL is a high-level, modular language that is embedded in the Quartus II system. AHDL Text Design Files (.tdf) can be created using the Quartus II Text Editor .AHDL file can be integrated with other types of design files in a hierarchical design. AHDL is applicable for complex combinational logic, group operations, state machines, truth tables, and parameterized logic design [23].

All language constructs are synthesizable.it is very easy to convert AHDL to Verilog code. In this language, there is more control but less high-level support. A disadvantage of AHDL is that it is proprietary .See a simple AHDL up Counter code [24].

4.2 No Common use languages:

In this section we will describe in brief rarely used languages. Table 6 describe No Common use languages and tools.

Table 6. No Common use languages and tools

Language/Tool	Description
Confluence	A functional HDL; has been discontinued
ELLA	No longer in common use
ISPS	Original HDL from CMU, no longer in common use
KARL	KARlsruhe Language, a Pascalish hardware descriptive language, no longer in common use.
M	A HDL from Mentor Graphics
CoWareC	A C-based HDL by CoWare. Now discontinued in favor of SystemC
AHPL	HDL Language developed at University of Arizona for representation of hardware for academics.

4.3 Traditional and Widely Used Languages:

In this section we will divide traditional and widely used languages in two parts. The first part is analog mixed signal languages (AMS) or analog languages, and the second part is digital languages.

4.3.1 Analog and Mixed Signal (AMS) languages:

VHDL-AMS, Verilog-AMS and System C-AMS allow modelling of discrete and continuous-time signals or a combination of discrete and continuous signals. So it is clear that HDL languages can represent AMS systems at a higher level of abstraction by bringing down the simulation time while providing the intended functionality of the design. A comparison between AMS languages or modelling approaches (VHDL-AMS, Verilog-A, System C-AMS, and SPICE) are made in [25].

VHDL-AMS is derived from VHDL language It includes analog and mixed-signal extensions (AMS) in order to define the behaviour of AMS systems (IEEE 1076.1-1999).It is an industry standard modelling language for mixed signal circuits. It provides both continuous-time and event-driven modelling semantics, and so is suitable for analog, digital, and mixed-signal circuits. It is particularly well suited for verification of very complex analog, mixed-signal and RF integrated circuits [26].

System C-AMS is an extension of System C that uses an open and layered approach. The base layer is the existing System C 2.0 kernel.

Verilog-AMS is a derived from Verilog language. It extends the event-based simulator loops of Verilog/System Verilog/VHDL, by a continuous-time simulator, which solves the differential equations in analog-domain. Analog events can trigger digital actions and vice versa. It was created with the intent of enabling designers of AMS systems and integrated circuits to create and use modules that encapsulate high-level behavioral descriptions as well as structural descriptions of systems and components [27].

SAMSA is a new tool for the VHDL-AMS system's simulation in Matlab .This tool is developed to make possible VHDL-AMS simulations in Matlab, a powerful scientific tool for numerical analysis, along with its associated complete set of toolboxes. The purpose of SAMSA is to have a sole VHDL framework where analog, digital systems can be designed and simulated [28].MATLAB Simulink tool can be used to model AMS systems if supporting libraries and functions are available. Simulink has an inbuilt analog tool set which can be used for AMS modelling and the accuracy results are comparable to that of spice simulation results.

4.3.2 DIGITAL LANGUAGES:

Digital languages consists of two types visual languages (as vVHDL) and non-visual languages. Here we would describe examples of non-visual languages. For more information about visual languages see [11].

A. VHDL, THDL, VITAL:

In 1981, the US DoD began writing the specifications for the Very-High-Speed Integrated Circuit (VHSIC) program. The intent was to be able to accurately describe the behaviour of circuits for documentation, simulation, and later for synthesis purposes. This program was known as VHDL. After two years of developing VHDL by IBM , Intermetics, and Texas instruments ,VHDL was finally released in August of 1985.

Due to the popularity of the language, the IEEE convened a committee to formalize the language. The current standard IEEE 1076-2008 is known informally as VHDL-2008.

VHDL is a large, complex, and powerful language. Although the language is feature-rich, many designs only require a small subset of the total capabilities[29].

The advantage of VHDL are : (1) It allows the behaviour of the required system to be modelled and verified before synthesis tools translate the design into real hardware circuits,(2)VHDL allows the description of a concurrent system. VHDL is a dataflow or concurrent language, unlike other computer languages such as C which runs sequentially (3)VHDL is multipurpose , portable language. Here is VHDL code for 4 bit up/down counter [31].

VITAL (VHDL Initiative towards ASIC Libraries) is an initiative to accelerate the development of sign-off quality ASIC

Novelty Journals

ISSN 2394-7314

International Journal of Novel Research in Computer Science and Software Engineering
Vol. 3, Issue 1, pp: (140-158), Month: January-April 2016, Available at: www.noveltyjournals.com

macro-cell simulation libraries written in VHDL by Leveraging existing methodologies of model development [30].

THDL (Templated HDL inspired by C++) an extension of VHDL with inheritance, advanced templates and policy classes. Table 7 shows comparison between Verilog &VHDL+VITAL [11].

Table 7. Comparison between Verilog &VHDL+VITAL

	Verilog	VHDL+VITAL
Source	C-based	Ada based
Learn	Relatively easy to learn	Difficult to learn
Data types	Fixed data types	Abstract data types
Constructs	Interpreted constructs	Compiled constructs
Gate-level timing	Good	Less good
Design reusability	limited	Good
Design management	limited	Good
Structure replication	No	Supported
Wide acceptance in	West Coast	East Coast

B. *VERILOG-HDL, System VERILOG:*

Verilog HDL originated at Automated Integrated Design Systems (later renamed as Gateway Design Automation) in 1985. Verilog HDL was designed by Phil Moorby, who was later to become the Chief Designer for Verilog-XL and the first Corporate Fellow at Cadence Design Systems. Gateway Design Automation grew rapidly with the success of Verilog-XL and was finally acquired by Cadence Design Systems, San Jose, CA in 1989. Verilog was invented as simulation language. Use of Verilog for synthesis was a complete afterthought.

Cadence Design Systems decided to open the Verilog language to the public in 1990, and thus OVI (Open Verilog International) was born. Until that time, Verilog HDL was a proprietary language, being the property of Cadence Design Systems. When OVI was formed in 1991, a number of small companies began working on Verilog simulators, including Chronologic Simulation, Frontline Design Automation, and others. Now there is variety of Verilog simulators available from many sources. For more details see Verilog homepage [32].A simple Verilog counter code could be seen [33].

Icarus Verilog is an open source compiler implemented for the IEEE-1364 Verilog HDL. It uses the Verilog to develop the source code for the design. The developed design can be targeted to FPGA and code modification is possible. Icarus Verilog V 0.8 supports the synthesis and V0.9 is only intended for simulation process. Official web site [34].

System Verilog is a superset of Verilog-2005, with many new features and capabilities to aid design verification and design modeling. As of 2009, the System Verilog and Verilog language standards were merged into System Verilog 2009 (IEEE Standard 1800-2009). OVM stands for open-source System Verilog verification methodology [35].A simple up/down counter example is here [36].

Many of the System Verilog and Verilog 2001 enhancements are already available in the VHDL language. There is also a new VHDL enhancement effort underway that will add test bench and expanded assertions capabilities to the language (the two areas where System Verilog will provide value over VHDL 2002).

The advent of hardware verification languages such as OpenVera, and Verisity's e language encouraged the development of Superlog by Co-Design Automation Inc. Co-Design Automation Inc was later purchased by Synopsys. The foundations of Superlog and Vera were donated to Accellera , which later became the IEEE standard P1800-2005: System Verilog. Overview comparisons between VHDL, Verilog and System Verilog can be seen here [37].

4.4. Verification languages (ex: PSL):

Ensuring that a design's implementation satisfies its specification is the foundation of hardware verification .The Accellera Property Specification Language (PSL) was developed to address these shortcomings in verification process. It gives the design architect a standard means of specifying design properties using a concise syntax with clearly-defined formal semantics. Also, it enables the RTL implementer to capture design intent in a verifiable form, while enabling the verification engineer to validate that the implementation satisfies its specification through dynamic (means, simulation) and static (means,, formal) verification means. Furthermore, it provides a means to measure the quality of the verification process through the creation of functional coverage models built on formally specified properties. Plus, it provides a standard means for hardware designers and verification engineers to rigorously document the design specification (machine-executable) [38].See an example of Verilog up-down counter code and related PSL assertions for it here [39].

PSL was specifically developed to fulfil the following general hardware functional specification requirements:(1)Easy to learn, write, and read ,(2)Concise syntax ,(3)Rigorously well-defined formal semantics ,(4)Expressive power, permitting the specification for a large class of real world design properties ,(5)Known efficient underlying algorithms in simulation, as well as formal verification.

PSL works alongside a design written in VHDL or Verilog. In future it may be extended to work with other languages. Properties written in PSL may be embedded within the HDL code or placed in different files [38].

4.5 Control Languages (ex: Tcl):

TCL (Tool Command Language) is a very powerful and easy to learn dynamic programming language[40]. It is suitable for a very wide range of uses, including web and desktop applications, networking, administration, testing and many more. Open source and business-friendly, Tcl is a mature yet evolving language that is truly cross platform, easily deployed and highly extensible. Tcl can apply in a typical FPGA tool flow :(1)Xilinx ISE and Tcl [41],(2)Quartus II and Tcl[42],(3)Microsemi Libero and Tcl [43].

TCL homepage is [44]

4.6 Rarely used and Alternative Languages :

In this section we will divide rarely used and Alternative languages in two categories first is digital software based languages and the second is PCB-HDL based languages. First category can be divided in two types structural and non-structural languages. Second category only contains PHDL language.

4.6.1 Digital Software Based Languages:

A. Non Structural Languages:

1. RHDL, HVL:

RHDL (Ruby Hardware Description Language) is an HDL based on the Ruby programming language. Developer idea in developing RHDL was to build an HDL on an object oriented programming language to allow HDL features (concurrent processes, signals, parallelism etc.) in addition to features which come with a modern, object oriented, and agile programming language like Ruby (www.ruby-lang.org). The intent is to allow simulation, verification and test bench creation features. RHDL is based on Ruby should allow modelling at a higher level of abstraction than is possible with VHDL or Verilog so it is not easy to translate RHDL to VHDL and/or Verilog.

A Hardware Verification Language, or HVL, is a programming language used to verify the designs of electronic circuits written in a HDLs. HVLs typically include features of a high-level programming language as well as features for bit-level manipulation similar to those found in HDLs.

RHDL users don't need to know much Ruby only small knowledge is enough in order to benefit from RHDL .RHDL is not a new language. It is a set of modules (code libraries) that allow Ruby to act like an HDL. Ruby has a concept called code blocks; these blocks are made into lexically scoped closures in order to define a domain specific language like RHDL without having to write a separate parser. RHDL can be used as HVL with a few more additions [*http://rhdl.rubyforge.org/*].See RHDL counter code here [45] .

2. Bluespec, Lava (based on Haskell):

Bluespec, Inc. is a semiconductor tool design company co-founded by Prof. Arvind of MIT . Arvind had developed the Bluespec language, a high-level functional hardware description programming language which was essentially Haskell extended to handle chip design and electronic design automation in general. Bluespec is partially evaluated to convert the Haskell parts and compiled it to the term rewriting system (TRS). It can be found in a System Verilog frontend. Bluespec has two product lines mainly for ASIC and FPGA hardware designers and architects. Bluespec supplies high-level synthesis (ESL logic synthesis) with RTL. The first Bluespec workshop was held on in 2007 at MIT. Counter example using Bluespec [46]. Bluespec is basically Haskell with some extra syntactic constructs for the term rewriting system (TRS) that describes what the hardware does. The type system has been extended with types of numeric kind [47].

Lava is an experimental, visual object-oriented, interpreter-based on Haskell programming language with an associated programming environment called Lava Programming Environment (Lava PE) that uses structure editors instead of text editors. Only comments, constants, and new identifiers can be entered as text.Declarations are represented in LavaPE as tree structures whose subtrees may be collapsed or expanded. The properties of the declared Lava entities can be edited through pop-up dialogs.Although executable code has a traditional text representation in LavaPE, it can be edited only as complete syntactic units, rather than character by character. [48].there is no available examples here.

3. MyHDL(based on python):

MyHDL is developed by Jan Decaluwe.It is a Python based hardware description language (HDL). MyHDL include many features like the ability to generate a test bench with test vectors in VHDL or Verilog ,the ability to convert lists of signals, the ability to convert output verification, the ability to do Co-simulation with Verilog, and An advanced data type system. MyHDL's translator tool automatically writes conversion functions when the target language requires them.A MyHDL design can be automatically converted to equivalent Verilog or VHDL code . MyHDL and conversion is probably the best solution available for language-neutral design. It is obviously superior to manual conversion. Moreover, direct convertors between Verilog and VHDL don't work very well. Using a syntactically Python language as the starting point is an easier way to a create a high-quality conversion tool [49].An example of bi-directional 4-bit Johnson counter with stop control could be seen in MYHDL homepage [49].

4. PSHDL:

PSHDL is an open source , board independent, and plain and simple hardware description language. Beta version web interface that was written in Dart with Web UI include the following New features the page scales are much better with lower resolutions, fast simulation, LEDs are dimmed with the duty cycle, ability to create board definition files and synthesis definition files ,Ports can be visually located on the PCB, the ace editor recycles the session, which stores history and cursor position, and dirty marking is based on SHA checksum [50].

```
module AMR {
  register uint<4> counter=counter+1;
  out bit<4> led;
  led=counter{3:0};
}
```

B. Structural Languages:

1. JHDL, Hardware Join Java (HJJ):

JHDL (Just-Another Hardware Description Language) is a low-level structural hardware description language, focused on building circuits via an Object Oriented approach that bundles collections of gates into Java objects. Implemented as a toolset and class library on top of the Java programming language, its primary use is for the design of digital circuits for implementation in FPGAs. Particular attention was paid to supporting the Xilinx series of chips [51].JHDL was developed at BYU in the Configurable Computing Laboratory, the project initiated in 1997.The latest update to the JHDL project was made in 2006 according to the official JDHL website[*http://www.jhdl.org/*]. An example for JHDL Counter cod is here [52]. JHDL doesn't support all forms of familiar digital systems design . In particular, asynchronous loops are unsupported and, it gives an error with simulator.Also behavioural synthesis is not yet fully supported. JHDL is found to be a very effective way to do FPGA design and will continue to use it to develop our applications. JHDL language features include structural hardware design, flexible module generators, table-generated finite state machines(FSM),a graphical "Workbench" toolkit,.

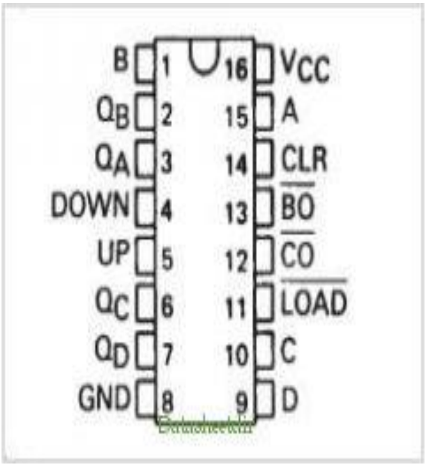
The Java Optimized Processor (JOP) has been created by Martin Schoeberl as part of his Ph.D. project (2005) and then being developed as an open source project. JOP is a soft-core Java-processor written in VHDL which makes it possible to execute Java code on FPGA. JOP is designed to be time predictable, and the number of Java byte codes (JVM instructions set) can be predicted in clock cycles. It is therefore possible to build real time Java applications. JOP is an open source, Additional information about JOP can be found at the official site [*http://www.jopdesign.com*] [53].

4.6.2 PCB-HDL based Languages:

1. PHDL:

PHDL is an open-source research-based Hardware Description Language (HDL) created at Brigham Young University that models text-based schematics for Printed Circuit Boards (PCBs). Its primary objectives for increasing designer productivity are to overcome limitations of scale, collaboration, and design reuse inherent with graphical schematic capture CAD tools. An eclipse plugin; developed with the Xtext frame work; provides many desirable integrated development environment (IDE) features to PHDL users Xtext. Standalone functionality is also provided for those who wish to work from the command-line and use their own editor. the plugin include many features such as highlighting, content assist, and automatic code completion[54]. An example for 4-bit counter

```
device Counter is
   attr refPrefix = "U";
   attr pkg_type = "16-TSSOP";
   attr supplier = "Jameco
Electronics";
   attr cost = "3.49";
   pin B = {1};
   pin Qb = {2};
   pin Qa = {3};
   pin Down = {4};
   pin Up = {5};
   pin Qc = {6};
   pin Qd = {7};
   pin gnd = {8};
   pin D = {9};
   pin C = {10};
   pin _Load = {11};
   pin _Co = {12};
   pin _Bo = {13};
   pin clr = {14};
   pin A = {15};
   pin vcc = {16};
end device Counter;
```



4.7. Tools:

Tools can be divided into three types web based tools, C to RTL tools, and graphical tools.

4.7.1 C to RTL Tools:

C to HDL or C to RTL tools also known as C-based hardware description languages (CHDLs) is a type of HLS tools. Generally HLS take functional specifications in high-level languages (in this c_ based language) and generate RTL or gate-level specifications that can be used in well-established RTL synthesis and place-and-route flows. The generated RTL is optimized to the specific target ASIC or FGPA technology.. The converted code can then be synthesized and translated into a hardware device such as a field-programmable gate array. The limitations of Hardware C and similar CHDLs are rooted in the modification of the C language semantics to support hardware constructs or requirements, thus making each CHDL a different dialect of C. Table.8 and 9 shows a brief abstraction of some of these HLS tools based on C.

Table 8 .Shows brief abstraction of C to Verilog tools, C to HDL tools and derive C languages

C to Verilog	C to HDL	Derived C languages
C to Verilog	Autopilot is developed by AutoESL .it compiles HDL code from various C_based languages such as C++, System C, ANSI. Xilinx revealed Vivado Electronic system level (ESL) as the new name for this tool [55].	SystemC Handel-C
bambu (free and open source ANSI C to Verilog tool based on GCC compiler)	ROCCC is free and open source C to HDL tool.it uses Eclipse as its use design environment	Impulse C SA-C
C-to-Verilog tool (NISC) from University of California, Irvine		Mitrion-C
Vsyn (C to Verilog, Russian project)		

Table 9. Shows brief abstraction of compiler tools , Synthesis tools and framework

Compiler Tools	Synthesis tools and framework
Altium Designer , Nios II C-to-Hardware Acceleration Compiler from Altera	Catapult C tool [56]
CebaTech's C2R Compiler , DIME-C	Cynthesizer C/C++ to RTL. Co-processor C to RTL synthesizer [57]. Agility compiler which is used to synthesize SystemC.
Symphony C compiler .the entry languages are ANSI C and C++. Trident compiler for floating point algorithms written in C. Streams C	HercuLeS is a high-level synthesis tool that automatically generates RTL VHDL for non-programmable hardware
FpgaC compiler works with C language and it is able to create a binary file to be downloaded into the FPGA card[58]. Compaan [with Eclipse SDK tool as its user environment] developed by compaan design for Xilinx FPGA [10].	SPARK is a C-to-VHDL high-level synthesis framework that employs a set of innovative compiler, parallelizing compiler, and synthesis transformations to improve the quality of high-level synthesis results

In this section ,a collection of comparative study between C to RTL tools will be represented here .A Comparative Evaluation study [59] is made between some of Automated HDL Generation tools or languages like SPARK, ROCCC, and DWARV based on a dataset of roughly 140 C functions. The number of functions in each application domain for DWARV and SPARK under consideration is depicted. These numbers represent the number of kernels who's generated VHDL was successfully synthesized. Total number of functions for all application is equal to 158. Table 10 summarize results between these compilers.

Another comparison ;Survey of C-based Application Mapping Tools for Reconfigurable Computing [60] ; based on implementing three classical algorithms Finite-Impulse Response (FIR), N-Queens and Radix Sort using DIME-C, Handel C, Impulse C, and VHDL. Experiments performed on Nallatech BenNUEY-PCI card with VirtexII-6000 FPGA. Table 11 summarize results of this comparison with a small approximation (research describe results in bar charts).

Table 10.Comparison results between DWARV, ROCCC ,and SPARK compilers

Compiler	**Resource Specification**	**Optimizations Guidance**	**Array dimension**	**High Level Constructs (%)**	**Supported Applications (%)**	**Reason For Limited Support**	**Number of Function per application domain (%)**	**summary**
DWARV	NO	N/A	1D	62%	50%	Structures	51/158=32.8 %	Least designer intervention , more relaxed restrictions ,does not offer any optimization
ROCCC	NO	Yes	2D	62%	6%	Perfectly Nested Loops	--	hides the hardware details from designer, highly application oriented ,requires detailed guidance to perform high level optimizations
SPARK V1.2	YES	Yes	1D	74%	50%	Constant size arrays	44/158=27.8 %	least automation, more relaxed restrictions required specification of the resources and optimization

Table 11. Comparison between Three based Languages with VHDL Related to Three Algorithms.

Algorithms	Resource utilization (%)	DIME-C	Handel C	Impulse C	VHDL
Finite-Impulse Response	Slices	10-20%	15-20%	30-40%	0-10%
	Multiplier	30-40%	30-40%	30-40%	30-40%
	Blocks Ram	Less than 10%	Less than 10%	Less than 10%	Less than 10%
	Clock frequency	High 100%	About 80%	50-60%	About 80%
N-Queens	Slices	Less 20-30%	More than 80%	30-40%	More 20-30%
	Clock frequency	80-90%	25%	65%	65%
Radix Sort	Slices	10%	Less than 10%	30%	Less than 10%
	Blocks Ram	10-20%	10-20%	10-20 % less	10-20%
	Clock frequency	100%	50%	55%	100%

Other comparison uses more languages and another features like abstraction level, test bench generation, implementation, and FPGA slices [61, 10].These two researches depends on same evaluation test application .the application chosen for these studies was Sobel edge detection algorithm. The complexity of the system design can be made easily with these tools .The design verification is made faster with the help of HLS.

It is clear that HLS do not provide a direct algorithmic to gate design path. In a similar way high performance architecture programmer do, designers need to modify and rewrite their source code so as to get the tool to derive the hardware architecture they actually want for their application. Even though such a rewrite process is much easier to carry with a HLL, it still leaves a lot of burden to the designer [62].

Table 12.Shows advantages and disadvantages of C to HDL tools

Advantages	Disadvantages
Wide acceptance	
Required HDL knowledge is significantly reduced or eliminated	Programs still require some tweaking for hardware compilation. C programs sometimes rely on features which are difficult or impossible to implement in hardware (as dynamic memory allocation). So a minimum knowledge of hardware design principle is actually required.
Time to preliminary results is much less than manual HDL	Reduced development time can come at cost of performance, resource utilization and power consumption.
Software-to-hardware porting is considerably easier	
Visualization of C hardware is far easier and fast for scientific community. It relate to leaving implementation details to the design algorithms and tools, including the ability to determine the precise timing of operations, data transfers, and storage [63].for more benefits of HLS tools in general [64].	Mapper instructions are many times more powerful than CPU instructions, but FPGA clocks are many times slower.
The productivity of designers increases with the abstraction level, as demonstrated by practices in both the software and hardware domains	
Allow designers to deliver an increasing larger number of integrated systems as compared to integrated circuits	
Facilitate the verification process of the generated hardware.	In general the code has to undergo various optimizations and transformations before the actual HDL generation, Optimized Software C is not equal to Optimized Hardware C
Mappers can parallelize and pipeline C code	however they generally cannot automatically instantiate multiple functional units. compiler has to identify parallelism in the sequential code before mapping it onto the target hardware because C is intrinsically sequential whereas hardware is truly concurrent[65,60].

4.7.2. Graphical Tools:

The LabView FPGA Module allows LabView to target FPGAs on NI Reconfigurable I/O hardware (RIO) so scientists and engineers can take advantage of the performance and flexibility of FPGAs without needing to learn the low-level design tools.

Using the LabView FPGA Module, a custom control and measurement hardware can be developed without any prior knowledge of HDL or board level hardware design. Using LabView is also an intuitive way to represent the timing, concurrency and parallelism of FPGA hardware. Measurement and control system components can be easily integrated with FPGA design using the LabView FPGA Interface. National Instruments (NI) offers a number of targets that include the R Series line of data acquisition products, the PXI Timing and Synchronization Module, the Compact Vision System and the rugged and modular Compact RIO platform [66].A Counter example designed by lab view [67, 68]

Data I/O's Azido, is a graphical hardware development environment. One of the features that distinguish it from schematic capture and other graphical tools is the automatic compile-time resolution of data types. This enables graphical polymorphism, and allows recursive instantiations of an object within itself. Azido provides a large library of logic, arithmetic, and control objects called the CoreLib, and defines a framework for providing support for arbitrary target platforms [69].

The objective of the MATCH (Matlab Compiler for Heterogeneous computing systems) compiler project is to make it easier for the users to develop efficient codes for configurable computing systems .It allows Matlab code to synthesised directly to an FPGA [70].Since it was developed, its technology was transferred to a start-up company, AccelChip. In 2006, Xilinx acquired AccelChip and the technology is now called as AccelDSP (Xilinx, 2009a). The design flow still follows the principles outlined above. The MATLAB code has to follow a particular coding style that assumes streamed input and output data.

Compaan/Laura is another MATLAB/C-based tool. This toolset converts a sequential MATLAB design into a network of independent processes, which is more dataflow orientated. The revised processing model is easier to map to hardware .This tool is used for systems that execute high-performance real-time signal processing and multimedia applications [71].

Xilinx System Generator is used in conjunction with Simulink and HDL Coder. Background a unified environment. System Generator provides a block-diagram front-end which takes advantage of the Xilinx CoreGen tool for generating high-performance computation cores. The abstractions shown by both Simulink and System Generator indicate a heavy focus on dynamic systems and signal processing applications [69].

```
-- Generated by MATLAB 7.14 and HDL Coder 3.0
-- -------------------------------------------------------------
-- Module: counter2
-- -------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std..ALL;

ENTITY counter2 IS
  PORT( clk    :  IN    std_logic;  reset    :  IN    std_logic;  clk_enable       :  IN    std_logic  );
END counter2;

ARCHITECTURE rtl OF counter2 IS

  SIGNAL enb                          : std_logic;
  SIGNAL HDL_Counter_count               : unsigned(3 DOWNTO 0);  -- ufix4
  SIGNAL HDL_Counter_out1               : unsigned(3 DOWNTO 0);  -- ufix4

BEGIN
  enb <= clk_enable;

  HDL_Counter_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      HDL_Counter_count <= to_unsigned(0, 4);
    ELSIF clk'EVENT AND clk = '1' THEN
      IF enb = '1' THEN
        HDL_Counter_count <= HDL_Counter_count + 1;
      END IF;
    END IF;
  END PROCESS HDL_Counter_process;

  HDL_Counter_out1 <= HDL_Counter_count;
```

4.7.3 Web based Tools (EDA Playground):

EDA Playground gives engineers immediate hands-on exposure to simulating SystemVerilog, Verilog, VHDL, C++/SystemC, and other HDLs. All you need is a web browser. The goal is to accelerate learning of design/testbench development with easier code sharing, and with simpler access to EDA tools and libraries. EDA Playground is specifically designed for small prototypes and examples.

With a simple click, run your code and see console output in real time. Pick another simulator version and run it again. View waves for your simulation using EPWave browser-based wave viewer.

Save your code snippets. Share your code and simulation results with a web link. Perfect for web forum discussions or emails. Great for asking questions or sharing your knowledge [72].

4.8. Compilers (ex: PERL):

Perl is a high-level programming language. Larry Wall invented Perl and thousands have contributed their time making it a very powerful tool. Perl borrows heavily from the C programming language and copies the really useful bits from sed, awk, Unix shell, and many other tools and languages.

Perl's process, file, and text manipulation facilities make it particularly well-suited for tasks involving automatic code generation, report filtering, Netlist patching, generating test vectors and controlling tools.

Perl has been used for translating between related languages: VHDL to Verilog, Xilinx Netlist Files to VHDL, VHDL to SystemC, etc. Source files with low originality are prime candidates for automatic generation. Perl is also useful for Coding Style conformance checking and enforcement [73].

Perl has been used for translating between related languages: VHDL to Verilog, Xilinx Netlist Files to VHDL, VHDL to SystemC, etc. Source files with low originality are prime candidates for automatic generation. VHDL engineers regularly write test benches. Perl is also useful for Coding Style conformance checking and enforcement.

5. CONCLUSION

From this study we conclude that there are many languages and systems that can be used for programming FPGA, for Example SA_C, Handle_C, Impulse_C are powerful language for image, video processing application, VHDL, VERILOG languages are difficult to learn and need a background knowledge about hardware devices, but they have a wide-ranging support, both in terms of software development tools and vendor support. AHDL has more control but less high-level support.

There are moves to raise the abstraction level of the design in order to reduce the complexity of programming in HDLs, creating a sub-field called high-level synthesis. Companies such as Cadence, Synopsys and Agility Design Solutions are promoting SystemC as a way to combine high level languages with concurrency models to allow faster design cycles for FPGAs than is possible using traditional HDLs. Approaches based on standard C or C++ (with libraries or other extensions allowing parallel programming) are found in the Catapult C tools from Mentor Graphics, the Impulse C tools from Impulse Accelerated Technologies, and the free and open-source ROCCC 2.0 tools from Jacquard Computing Inc. Annapolis Micro Systems, Inc.'s CoreFire Design Suite and National Instruments LabVIEW FPGA provide a graphical dataflow approach to high-level design entry and languages such as SystemVerilog, *SystemVHDL*, and Handel-C seek to accomplish the same goal, but are aimed at making existing hardware engineers more productive, rather than making FPGAs more accessible to existing software engineers. It is also possible to design hardware modules using MATLAB and Simulink using the Mathworks tool *HDL Coder* or *Xilinx System Generator* (XSG) from Xilinx (formerly *Accel DSP*).It is easier to create more readable and safer coding with Java which is also platform independent. The use of JOP will facilitate the acceptance of Java for embedded systems, and furthermore Java can broaden the use of FPGA.

Table 13 Shows proposed overview table of all PLD languages, tools, compilers.

Languages	Early HDL languages		--	PALASM, ABEL,AHDL,CUPL , LOLA(simple language used for teaching)
	No common use		--	ELLA, ISPS, KARL,M (Mentor Graphics),Confluence
	Rarely used	Digital ,	Non	Dart BASED :PSHDL(generate VHDL CODE,

	and Alternative languages	Software based languages	Structural approaches	Alternative syntax, web interface)
				Ruby based: RHDL.
				Python based: MYHDL(generates VHDL code, Alternative syntax)
				Java based: JOP, Hardware Join Java (HJJ)
				Haskell based: Bluespec, Lava, Hydra, HHDL
				Haskell and Verilog based: Blue spec System Verilog (BSV)
			Structural approaches	Java based: JHDL
				Block composition based :Quartz
				Prolog based system :HIDE
		PCB HDL	--	PHDL(Open source research based) ,EDA solver
	Widely used Traditional languages	analog	Non visual	,VHDL-AMS, Analog Hardware Descriptive Language, Spectre HDL , VERILOG-AMS,HDL-A
		digital	Non visual	VHDL, VERILOG-HDL, system Verilog, THDL++(VHDL extension)
			Visual languages	Behavioural :VVHDL
				Dataflow: Khoros based, Simulink based, others (like labview, pixel stream)
				Hybrid:VERTIPH
	Verification(HVL)		--	PSL,UVM,OVM, System Verilog, Open Vera, e, and System C
	Control language		--	TCL, System TCL
Tools	C to RTL(languages, tools) High Level Synthesis tools(HLS)		Software based languages Augmented languages	SYSTEM-C,HANDEL-C,SA-C ,STREAM-C, ESys.net, Co WareC (c-based), ParC (Parallel C++),ASC, Mitrion-C, A/RT, Handel-C derivatives (Bach-C).
			Software based languages Native compilation techniques	Transmogrifier C,SUIF-Based Compiler ,Impulse C, Dime-C, Dime-Talk, Catapault C, Matlab based
	Web based tool		--	www. Edaplayground.org
	Graphical tools(HLS)		--	National Instrument's LABVIEW, Matlab HDL coder, SAMSA tool(analog), Xilinx System Generator, Data I/O's Azido
Compilers	PERL, C++, PYTHON, Csh, CUPL,PSHDL (ONLINE compiler)			

REFERENCES

[1] Esam El-Araby1 , Mohamed Taher1 , Mohamed Abouellail1 , Tarek El-Ghazawi1 , and Gregory B. Newby2," a comparative analysis of high level programming for reconfigurable computers: methodology and empirical study", 3rd Southern Conference on Programmable Logic, 2007.

[2] Donald G. Bailey ,"Design for Embedded Image Processing on FPGAs", August 2011.

[3] Richard Wain, Ian Bush, Martyn Guest, Miles Deegan, Igor Kozin and Christine Kitchen ,"An Overview of FPGAs and FPGA Programming"; Initial experiences at Daresbury, Council for the Central Laboratory of the Research Councils ,November 2006 Version 2.0.

[4] http://www.design-reuse.com/

[5] http://www.design-reuse.com/articles/22773/analog-mixed-signal-modeling.html

[6] http://ufdcimages.uflib.ufl.edu/UF/00/09/47/49/00002/MAPLD05_Holland.pdf

[7] Peter Athanas, Nicolas Sklavos, Dionisios Pnevmatikatos," Embedded Systems Design with FPGAs", Springer Science Business Media, LLC 2013, pp 201-224.

[8] Yana Yankova, Koen Bertels, Stamatis Vassiliadis, Roel Meeuws, Arcilio Virginia," Automated HDL Generation: Comparative Evaluation" ,IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.

[9] Grant Martin, Gary Smith, "High-Level Synthesis: Past, Present, and Future, IEEE Design & Test of Computers",2009.

[10] Wim Meeus · Kristof Van Beeck · Toon Goedemé · Jan Meel · Dirk Stroobandt," An Overview of Today’s High-Level Synthesis Tools", Springer Science and Business Media, LLC 2012.

[11] Clive "Max" Maxfield ,"The Design Warrior's Guide to FPGAs" ,Mentor Graphics Corporation and Xilinux,2004.

[12] Lattice Semiconductor Corp ,"Lattice Generic Array Logic Handbook OCR", 1986.

[13] David Pellerin, Scott Thibault, "Practical FPGA Programming in C" , Prentice Hall PTR, April 22, 2005.

[14] http://en.wikipedia.org/wiki/PALASM

[15] "CUPL - The Universal Language For Programmable Logic" (Press release). San Jose, CA: Assisted Technology, Inc. 1983. An early 1983 pre-release datasheet for CUPL, CUPL borchur.

[16] Logical Devices, Inc.(LDI) , CUPL Programmer's Reference Guide,1997.

[17] Lee, Kyu Y.; Holley, Michael; Bailey, Mary; Bright, Walter. "A High-Level Design Language for Programmable Logic Devices". VLSI Design (Manhasset NY: CPM Publications), June 1985.

[18] http://en.wikipedia.org/wiki/Advanced_Boolean_Expression_Language

[19] Stefan Hans-Melchior Ludwig," Fast Hardware Synthesis Tools and a Reconfigurable Coprocessor", PHD thesis, Swiss Federal Institute of Technology Zurich,1997.

[20] Niklaus Wirth ," LOLA system notes", institute of computer system,1995

[21] http://www.cs.inf.ethz.ch/projects/lola/lola_lang/

[22] Hans Eberle," Tools for Digital Circuit Design using FPGA", institute of computer system, 1994.

[23] http://quartushelp.altera.com/14.1/mergedProjects/hdl/ahdl/ahdl_intro.htm

[24] http://en.wikipedia.org/wiki/Altera_Hardware_Description_Language

[25] http://www.design-reuse.com/articles/22773/analog-mixed-signal-modeling.html

[26] Christen E., Bakalar K.,"VHDL-AMS-a hardware description language for analog and mixed-signal applications",Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on] Volume 46, Issue 10, Oct. 1999.

[27] Accellera, "Verilog-AMS Language Reference Manual Analog & Mixed-Signal Extensions to Verilog-HDL", Version 2.1, January 2003.

[28] M. Zorzi, N. Special, G. Masetti DEIS, "A New VHDL-AMS Simulation framework in Matlab", IEEE International Behavioral Modeling and Simulation Workshop,2002.

[29] William Kafig, VHDL 101, Elsevier Inc , 2011.

[30] http://vhdl.renerta.com/mobile/source/vhd00080.htm

[31] https://startingelectronics.org/software/VHDL-CPLD-course/tut19-up-down-counter/

[32] http://www.verilog.com/

[33] http://en.wikipedia.org/wiki/Verilog

[34] http://iverilog.icarus.com/

[35] Bergeron, J., Cerny, E., Hunter, A., Nightingale, A., "Verification Methodology Manual for System Verilog ", 2006.

[36] http://www.asic-world.com/

[37] Stephen Bailey ,"Comparison of VHDL, Verilog and System Verilog", Digital Simulation White Paper ,2003.

[38] Accellera ," Property Specification Language Reference Manual", Version 1.1, June 9, 2004.

[39] http://download.springer.com/static/pdf/629/bbm%253A978-1-4020-8586-4%252F1.pdf?originUrl=http%3A%2F%2Flink.springer.com%2Fbook%2Fbbm%3A978-1-4020-8586-4%2F1&token2=exp=1456829997~acl=%2Fstatic%2Fpdf%2F629%2Fbbm%25253A978-1-4020-8586-4%25252F1.pdf%3ForiginUrl%3Dhttp%253A%252F%252Flink.springer.com%252Fbook%252Fbbm%253A978-1-4020-8586-4%252F1*~hmac=8bdf8e9a41127284c1c5e12ee1615f2a9c113dbb8c0bf9861d1736864154b96f

[40] http://core.tcl.tk/jenglish/gutter/browse.html

[41] https://www.doulos.com/knowhow/tcltk/xilinx/

[42] https://www.doulos.com/knowhow/fpga/Automating_Tool_Flows_with_Tcl/quartus.php

[43] https://www.doulos.com/content/training/tcl_essential.php

[44] http://www.tcl.tk/

[45] http://www.eetimes.com/document.asp?doc_id=1216487

[46] Bluespec ,"BSV 101: Designing a counter Using the Bluespec Development Workstation", Bluespec, October 27, 2009

[47] Paul Hudak, John Hughes, Philip Wadler, Simon Peyton Jones," A History of Haskell: Being Lazy With Class", April 16, 2007.

[48] Klaus D. Günther1, Irmtraut Günther1,"Lava – An Object-Oriented RAD Language Designed for Ease of Learning, Use, and Program Comprehension", Institute for Secure Tele-cooperation, Rheinstr.,Germany,2001.

[49] http://www.myhdl. Org/

[50] http://blog.pshdl.org/2014/03/web-updates.html

[51] http://www.encyclo.co.uk/meaning-of-JHDL

[52] http://www.jhdl.org/documentation/users_manual/behavioralModelling.html

[53] M.S. Sorensen, "Java on FPGA", Danish technological institute, 2008.

[54] http://phdl.sourceforge.net/2.1/index.php

[55] Zhiru Zhang, Yiping Fan, Wei Jiang, Guoling Han, Changqi Yang,and Jason Cong," AutoPilot: A Platform-Based ESL Synthesis System ,high level synthesis from Algorithm to digital circuits ",chapter 6, Springer Science + Business Media,2008.

[56] Fingeroff M High-level synthesis blue book". Xlibris, Philadelphia. ISBN 13 (HB): 978-1-4500-9724-6. ISBN 13 (eBook): 978-1-4500-9725-3,2010.

[57] Ben Hounsell, Richard Taylor, Co-processor Synthesis: A New Methodology for Embedded Software Acceleration, Automation and Test in Europe Conference and Exhibition, IEEE, 2004.

[58] Harish Bhutra , Usha Patel, Darshana Upadhyay, "FPGAC: - High Level Synthesis Tool", IJCSC, Volume 5 ,Number 1, March-Sep 2014, pp. 150-153

[59] Yana Yankova, Koen Bertels, Stamatis Vassiliadis, Roel Meeuws, Arcilio Virginia," Automated HDL Generation: Comparative Evaluation", IEEE International Symposium on Circuits and Systems, 2007.

[60] http://klabs.org/mapld05/presento/215_holland_p.ppt

[61] M. Chinnadurai and M. Joseph ,"High Level Synthesis Tools-an Overview from Model to Implementation", Middle-East Journal of Scientific Research,2014.

[62] Alexandre Cornu1, Steven Derrien, and Dominique Lavenier, "HLS Tools for FPGA: Faster Development with Better Performance, Reconfigurable Computing: Architectures, Tools and Applications", 7th International Symposium, Belfast, UK, March 23-25, 2011.

[63] Philippe Coussy , Adam Morawiec ,"High-Level Synthesis From Algorithm to Digital Circuit", Springer Science and Business Media,2008.

[64] Coussy P, Takach," introduction: raising the abstraction level of hardware design". IEEE Des Test Computer 26(4):4–6,2009.

[65] J. S´erot, F. Berry, S. Ahmed, "Implementing Stream-Processing Applications on FPGAs : A DSL-Based Approach" , Conference: International Conference on Field Programmable Logic and Applications, September 5-7, Chania, Crete, Greece, 2011.

[66] http://sine.ni.com/nips/cds/print/p/lang/en/nid/13743

[67] http://www.ni.com/white-paper/5967/en/

[68] http://zone.ni.com/reference/en-XX/help/371599G-01/lvfpgaconcepts/customizing_i_o/

[69] David C. Uliana , "FPGA-Based Accelerator Development for Non-Engineers", Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia,2014.

[70] N. Banerjee …. "MATCH: A MATLAB Compiler for Configurable Computing Systems," Technical Report, Center for Parallel and Distributed Computing, Northwestern University, August, 1999.

[71] Todor Stefanov Claudiu Zissulescu Alexandru Turjan Bart Kienhuis Ed Deprettere,' System Design using Kahn Process Networks: The Compaan/Laura Approach' Proceedings of the Design, Automation and Test in Europe Conference and Exhibition ,IEEE,Leiden Embedded Research Center Leiden Institute of Advanced Computer Science Leiden University, The Netherlands,2004

[72] http://edaplayground.readthedocs.org/en/latest/intro.html.

[73] https://www.doulos.com/knowhow/perl.