VERY LARGE SCALE INTEGRATION DESIGN ENVIRONMENT RESEARCH AND DEVELOPMENT: DESIGN TOOLS, DATA LIBRARY AND CROSS-SITE POPULATION MANAGEMENT SYSTEM

By

CALVIN BOEY MUN LEK

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

In partial fulfilment of the requirements

For the degree of

BACHELOR OF INFORMATION TECHNLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Perak Campus)

JAN 2014

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Very Large Scale Integration Design Environment Research and Development**: Design Tools, Data Library and Cross-site Population Management System" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : Calvin Boey Mun Lek

Date : 7th April 2014

ACKNOWLEDGEMENTS

I would like to extend my appreciation towards my project supervisor, Mr Kuek Chian Shiun who gave me the opportunity to work on this project. Mr Kuek have not only helped me academically, but have been an excellent mentor and friend. Working along him have lead me out of my comfort zone and allows me to meet countless of useful exposure and experiences which I can never ask for.

I would also like to give thanks to my buddies at Intel, Penang where I served for my industrial training. These includes my work supervisor, Mr Li Kok Wei who is the expert in design automation and my senior and colleague, Mr Kelvin Leong who have gave me many useful advice.

Last but not least, I owe much gratitude towards my parents, family and friend who have been supportive and understanding. This includes the love of my life who have been my pillar of strength.

ABSTRACTS

Digital design environment is a vital infrastructure in Very Large Scale Integration (VLSI) design. However, such infrastructure is lacking in the local education field as developers work in an unorganized manner and without intra-universities collaboration. In this project, a VLSI design environment capable of revision control, library and archive management is being developed. On top of that, design library sharing is also made possible by leveraging on cross-site data synchronization techniques. With all these in place, there will be an increase in productivity of VLSI designers as there is less need to focus on how to manage the tools and designs. Besides, human error is minimized with the introduction of Graphic User interface (GUI) design library management. The learning and research process on VLSI can be accelerated as duplication of work between universities will be greatly minimized via design library sharing.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CAD	Computer-aided Design
DC	Design Compiler
DRC	Design Rule Checker
DRCS	Distributed Revision Control Software
GUI	Graphical User Interface
HDL	Hardware Description Language
IDE	Integrated Development Environment
IP	Internet Protocol
JDK	Java Development Kit
JRE	Java Runtime Environment
LAN	Local Area Network
NCC	Network Consistency Checker
OS	Operating System
RCS	Revision Control Software
RTL	Register Transfer Level
SSH	Secure Shell
URL	Universal Resource Locator
VCS	Verilog Compiler Simulator
VLSI	Very Large Scale Integration
VNC	Virtual Network Computing

CHAPTER 1: INTRODUCTION

1.1 Introduction

While the Very-Large Scale Integration (VLSI) has grown to the scale of billions of transistor per chip, Malaysia is still lacking overall in the competency to handle large-scale and complicated design. The art of complex VLSI design is only practiced in a small fraction of the industry such as Intel(Intel Corporation 1968), Advanced Micro Devices(Advanced Micro Devices 1969) and Altera(Altera Corporation 1995) while the bulk of electronics/semiconductor companies only focus on back-end technologies (assembly and testing). The research in the area of VLSI within the country's universities is also minimal and could not keep up with industry's fast pace.

Part of the biggest reason for the lack of focus in this area is due to the poor infrastructure and environment to do complicated design. This project, therefore, aims to contribute in terms of developing a VLSI design environment system that unifies the resources among local universities and industries. The long-term target is to establish an ecosystem for VLSI design within the country. VLSI design environment is responsible for aiding designers at all level of design workflow; simplifying tedious but necessary steps, tracking and logging simulation result and reports, allowing multiple users to work on single or multiple design concurrently and ensuring proper structure and hierarchy of design.

A VLSI design environment system in our context involve four main areas; tools management, revision control of libraries, archive reporting system and cross-site population control. This project aims to create an all-in-one design environment solution similar to the in-house tools used in the industry but with certain modification to better suit university-level design work. Revision Control Software (RCS) which

allows management of changes to documents or files has been popular among software and web development but absent in university level work. A certain form of extension will be written to work with current RCS to enable better compatibility and smoother workflow in VLSI design. Furthermore, Graphical User Interface (GUI) will be developed to improve user friendliness by removing the need of remembering commands and switches. This solution is able to provide a platform for multiple developers to work on a single or multiple design concurrently.

VLSI design involved different abstract of design description such as behavioral, structural and physical, there are different tools for each design stages. Most often, a design will require translation from one abstraction level to another such as from behavioral description to schematic and then to physical layout. Furthermore, different project may require different tool version, process library or cell variation (speed/area/power) which adds to the complication of design flow. This project includes extensive scripting in automating the design process in hope of minimizing human error such as missing command-line switches, linking the wrong libraries or completely forgetting crucial design steps.

VLSI design tools may requires huge computational power beyond the ability of a standard home workstation. This project's solution enables the utilization of computational power of server and partially solving the costly licensing issue by making use of virtual network connection and remote access by allowing developer to login from a remote terminal. Session managers will also be utilized to enable work to be resume when changing terminal or after short disconnection. Furthermore, a network of servers between remote sites can leverage on cross-site data replication which allows data to be backup on different clusters of computing networks not dependent on the geographical location. This allows design library sharing and collaboration between different parties with ease.

1.2 Problem Statement

Up-to-date, a systematic and holistic VLSI design environment at institution of higher learning in Malaysia is in none existence; far from what is being practiced at industry level. Due to the scale of project which is often small, designers tend to work-solo based on the workflow each individual deem as right. Problem arises when several designers come to work on a single project as there is no commonly accepted design workflow.

VLSI design is a chain of process often requiring user to invoke each tool and files (i.e. constraint, library) manually which leads to **unnecessary/extra tools or wrong tool's version being called up**. Designer needs to have the knowledge to manually specify search paths and setting the operating system's environment variable correctly before the tools will run. Then, certain commands operates tools which requires designer to link the design file with targeted foundry process technology and spice models. All these requires user to work on command line with multiple switches and flags that are easily forgotten. Furthermore, different design files may requires different version of the same tool to be utilized. Sometimes, designer mistyped the library or forgotten some necessary flags which then produce unintended results unknowingly.

As design progresses, designer make changes to design files. If no updated backup was made and the design turned out to be worse, it will be very **tedious to rollback changes and impossible in certain situation**. It is ideal if a complete history of the project files together with simulation or output reports is always available. There is a need for a design environment where multiple users can work on a project concurrently and all changes being logged. However, files should not be revision controlled by simply taking snapshots as disk space consumption will quickly escalate. On the other hand, if a derivation of the current development objective is needed, designer should be able to quickly replicate the existing design to startup a new branch. A complete history

including branches and main design trunk should be clearly laid out for effective project management and objective direction.

A complete suite of **software and tools for all level of work in the field of VLSI design is extremely costly**. Besides, it is wasteful for engineering student to purchase these licenses when they will only be working on it for less than a couple of years. Even if license is provided, setting and linking the software will be tedious and computational power will be an issue when the design grows. There is a need for these tools to be hosted somewhere where designer can tap into the processing power of the available servers from a simple terminal. Furthermore, it would be better if designers can exit the current session and continue working at a later time without needing to start all over again. Infrastructure sharing should be practiced for the benefit of the industry.

VLSI design in the industry can expand to an incredibly large scale involving thousands of engineer for a single product. However, there are **too little collaboration and sharing of design library at the education level**. Each institution of higher learning tend to work on their own without collaboration with other universities which leads to a high duplication of work. Even if there were some form of minor sharing, passing of files around emails and flash drives is not professional nor synchronized. Besides, there is currently no proper platform for VLSI design library sharing between multiple server-sites which ensures modification made to the project are being propagated to all the other sites. Generic file synchronization available on the market often to not include revision control which makes it none suitable for VLSI design library.

1.3 Project Objective

The general objective of this project is to based it on open-source software, write customized script or code to better suit VLSI design workflow and enable certain automation in design work while not forgetting design collaboration between different sites.

The first objective is to **develop project management system to manage tool-sets, configuration and synchronization process.** Necessary directories and files will be generated when a new project is created. If a sub-project is to be created, the respective constrain files will be reimported. Since only a single foundry process technology should be used within a single project, scripts will be written to ensure that designer has no need to manually specify the files more than once. Besides, meddling with environment variables will no longer be needed as it will be automatically taken care of based on the project type and tools needed.

The second objective is to **setup and customize a Revision Control Software (RCS) based on VLSI design workflow** which is capable of logging changes made to projects. Besides, generated output and reports will be logged as well so that comparison can be easily made later on. On top of these, branching and merging of design will be well taken care of. Custom locking mechanism will be developed to work with the RCS which allows locking of design file so that other users cannot make modification while the file is being modified by the current developer (who holds the lock). This lock can be forcefully unlocked by system administrator.

The third objective is to **configure an environment capable of handling user session from remote location** in order to harness the computing power of server. Designer should have the ability to access project files stored on the server and performing operation on-the-fly without needing to download the actual design file. Remote access

also allows time-sharing of licenses although not part of this project's scope. Authentication and permission hierarchy will be established to ensure that design are not read or modified by unintended personnel. In the education field, sharing design on a central server helps instructor's job to check if students' work are satisfactory.

The forth objective is to **setup a distributed data replication scheme to enhance the reliability of design libraries**. Data reliability refers to file corruption and network down-time (being unable to access design library in other servers). Studies on synchronization issues especially on data conflict and resolution will be done to enable a better performing system with high data reliability. Modification of any files to the project will be propagated to other site servers. Data replication should avoid conflict such as when two parties intend to update the same file at a time. On top of these, data replication on multiple sites can be fully utilized as an emergency backup solution. When data such as revision control information goes corrupted on a server, it is possible to resynchronize based on other site's repository.

The last objective is to **develop a Graphical User Interface (GUI) for project management and design environment navigation** in order to enhance end-user experience. Instead of typing a long command to create a new project, user will have the ability to do it with few clicks of a button. Besides, GUI ease the tasks of system administrator as project status and information will be laid out in a systematic manner. GUI also acts as a wrapper which protects the system from being tempered by user. Simple invoking commands on the terminal will not be possible as most of the task have to be performed via the GUI.

1.4 Project Scope

This project involve heavily on development and minimal on research. On the part of development, a design environment which supports the following:

1. Centralized (local) revision control for project, files and generated output reports including branching support and file locking mechanism.
2. Project management system; scripting and automation in linking design files and libraries, foundry process technology and tools configuration.
3. Virtual network computing remote access to harness computing power of servers.
4. Graphical User Interface (GUI) for project management and navigation of design environment
5. Archive and reporting system as part of validation process in order to maintain integrity of library.

1.5 Contributions

This main contribution of this project is towards building an infrastructure whist establishing an ecosystem for the VLSI design industry and education. It will be the first comprehensive VLSI design environment in Malaysia which enables integration between several universities and the industry. Design library sharing and collaboration between different parties will be made easier. This infrastructure helps in bringing VLSI design in the education field to a whole new level.

This design environment will enforce a structured design workflow on developers and students instead of allowing each personnel to work based on however they like which is deemed unorganized. Implemented revision control system ensures that reverting to previous changes when necessary is always possible. Besides, evaluation of students' work will be made easier with the ability to trace every progress made. The archiving and reporting system is able to give a summary on the statuses of each library which in turn ease library integrity monitoring.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This project involve setting up a complete VLSI design environment which consist of design file version control & user management, automation of design tools setup & configuration and cross-site population and design library sharing between multiple sites. Thus, the next session will review on four different related systems; revision control software, VLSI design tools, data synchronization software and remote access.

In each category, few examples of existing software whether open source or commercial which is available on the market will be reviewed. The general workflow or methodology of the software will be explained very briefly. Advantage and disadvantages pertaining to this project will be listed and explained.

Besides, several papers on VLSI Design Automation and Design Environment will be reviewed. These also includes a publication on a system that is being deploy in the VLSI design industry. Other papers reviewed simply list the problems faced by engineers and the corresponding proposed solution.

2.2 Review of Related System

2.2.1 Revision Control Software

Revision Control Software (RCS) which carries other names such as version control and source control is the management of change to directories and files specifically. These software are not only aimed at providing a backup solution but to also establish a platform where contributors can work on projects concurrently with minimal synchronization overhead. RCS can generally be categorized into two main groups; centralize or distributed, although hybrid of the two has recently emerge. Most well established RCS are cross-platform compatible where some even supporting mobile devices such as application for phone and tablets.

Subversion(Apache 2000) and Concurrent Versions System(CVS 1990) are two common examples of a centralized version control software while Git(Git 2005) and Mercurial(Mercurial 2005) being distributed. Centralized system are typically client-server model on which users synchronize to a single and central repository hosted on the server. On the opposite end, the distributed system leverage on peer-to-peer model where patches of change are being shared among developers and end users. However, normally, there is a repository which is accepted as the main and official copy. The differences between the two architecture together with the main working mechanism will be reviewed in the following few paragraphs.

Revision control started off with the centralized model as companies need a reliable way to manage multiple developers working on a multiple projects concurrently with the ability to review and roll back changes. Thus, a developer will need to checkout necessary files before being able to make changes, proceed to make the modification on the local working copy and then check-in the changes back to the server. After the

file have been checked out, a lock is place on the file so that others cannot checkout the same file. This is the de facto way in which centralize version control software avoid conflicts. This architecture also means that developers must have connection to the server in order to do their work. Sometimes, the system administrator need to forcefully remove the lock such as when an employee checked out and went for a long vacation.

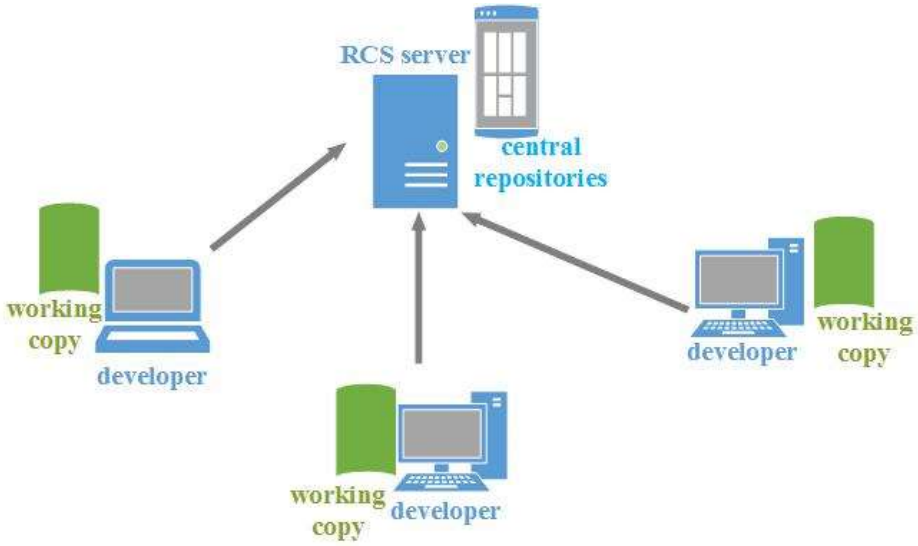


Figure 1: Centralized Revision Control System

Distributed RCS (DRCS) started off big when the open source community needed a system which can work around the hassle of the file locking mechanism. Besides, open source developers do not normally work together in the same geographical location neither are they synchronized between each other. Thus, the locking mechanism defeats the purpose of many developers working concurrently as workflow had become serialized. DRCS allows each developer to have the entire repository on their machine, commit changes to it and then only merging when is deemed necessary. These change sets can be imported/exported easily and shared via email or flash drive transfer. Typically, the repository will be temporary held online by some form of RCS manager

and a push/pull can be done to propagate the changes. When a developer newly join a project team, the repository can be duplicated to the local machine by pulling from a universal resource locator (URL) for open source projects. For close source projects, secure connection with some authentication will be required for pulling changes.

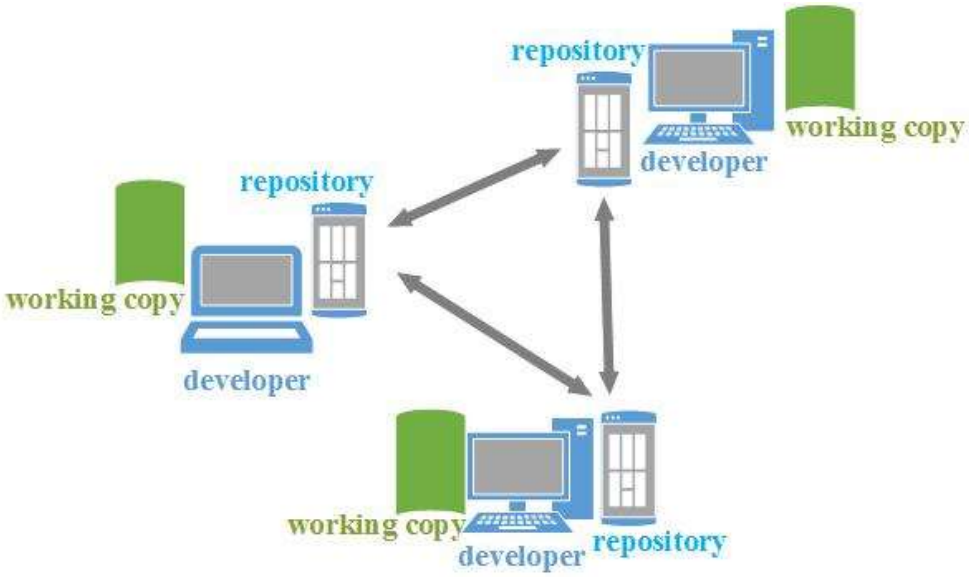


Figure 2: Distributed Revision Control System

Every changes being committed or checked in will leave a new entry in the RCS log file. This log files keep track of the history of changes of all files within the repository, such as when the changes were committed, who made the changes, why was it done (committer to comment) and what modification were made. Each changeset will be identified by a unique identification normally generated via hash function which has very low possibility of being the same. Most RCS make it difficult to remove changes from the history log as all changes should be accountable in case there is a need to review changes later in time. Project manager can choose to tag any particular version with a name, normally to identify project milestone or software release version.

Branching is another useful function of a RCS which allows different version of a project to be developed concurrently. Merging on the other hand, is to recombine two branches to a single trunk. Example, branching can be used when the current NAND gate would like to be improved in two separate areas; speed and low power. Then, merging is useful when a hybrid such as medium speed and medium power consumption NAND gate is to be developed. However, merging of VLSI design files such as text based schematic or layout description most often have to be done manually as current tools do not semantically comprehend the files. Alternatively, two new repository can be created if it is foreseen that merging will not be necessary.

2.2.2 VLSI Design Tools

A VLSI design process involve many different tools; Hardware Description Language (HDL) editor, digital simulator, logic generator, logic optimizers, schematic capture tools and physical layout editor. Not all tools will be used in a typical design workflow but a large number of them will. These software typically runs on variants of UNIX operating system. Two common vendors providing commercial Electronic Design Automation (EDA) tools highly used in the industry which are also available on the market is Synopsys(Synopsys 1986) and Cadence(Cadence Design Systems 1988). Electric VLSI(Static Free Software 1982) and LT Spice IV(Linear Technology 2003) are examples of open-source and freeware EDA tools respectively.

Synopsys provides a wide range of commercial products in the domain of EDA that manages circuit design from behavioral level to final layout file. License for each module can be purchased separately depending on what is needed. Verilog Compiler Simulator (VCS) is a tool which compiles Verilog HDL files and executes functional and timing simulation of a design. Verilog testbench file and external netlist have to be

specified. This tool also provides GUI but special flags have to be added to the command during compilation.

```
create_clock -p 10 [get_clocks clk]
set_max_area 200000
set_clock_uncertainty -setup 0.8 [get_clocks clk]
set_clock_latency -source -max 3 [get_clocks clk]
set_clock_latency -max 1 [get_clocks clk]
set_clock_transition 0.08 [get_clocks clk]
set_output_delay -max 0.2 -clock clk [all_outputs]
```

Figure 3: Specification of timing constraint in Synopsys DC shell

Design Compiler (DC) is among the popular tool provided by Synopsys which allows Register Transfer Level (RTL) code to be synthesize to produce HDL netlist. Design constrains such as timing constraint (clock source latency, skew, jitter and margin) must first be specified via the DC shell. Another useful tool provided by Synopsys is the Prime Time for timing analysis. Prime Time is able to generate timing report based on the compilation from Design Compiler and constraints specified earlier on. From this report, it is known if the design's timing specification is met. If not met, may need to try recompiling with more optimization or return to the previous design level and redesign or change the design timing requirement (often not possible).

Electric VLSI (EVLSI) is an open-source Computer-aided Design (CAD) system for VLSI circuit design. It includes a large set of tools such as design-rule checkers, simulators, routers, layout generators and supports schematic (logic and transistor level) drawing, physical layout and also HDL (Verilog and VHDL). Best of all, this software is written in Java and can run (without needing installation) in any OS with Java Runtime Environment (JRE) running. Besides, there is a simple built-in project management tool which enables multiple user to work on files based on simple locking

mechanism. However, the tool is not capable of handling large user base. The downside is that this software does not allows user to specify their own foundry process technology file.

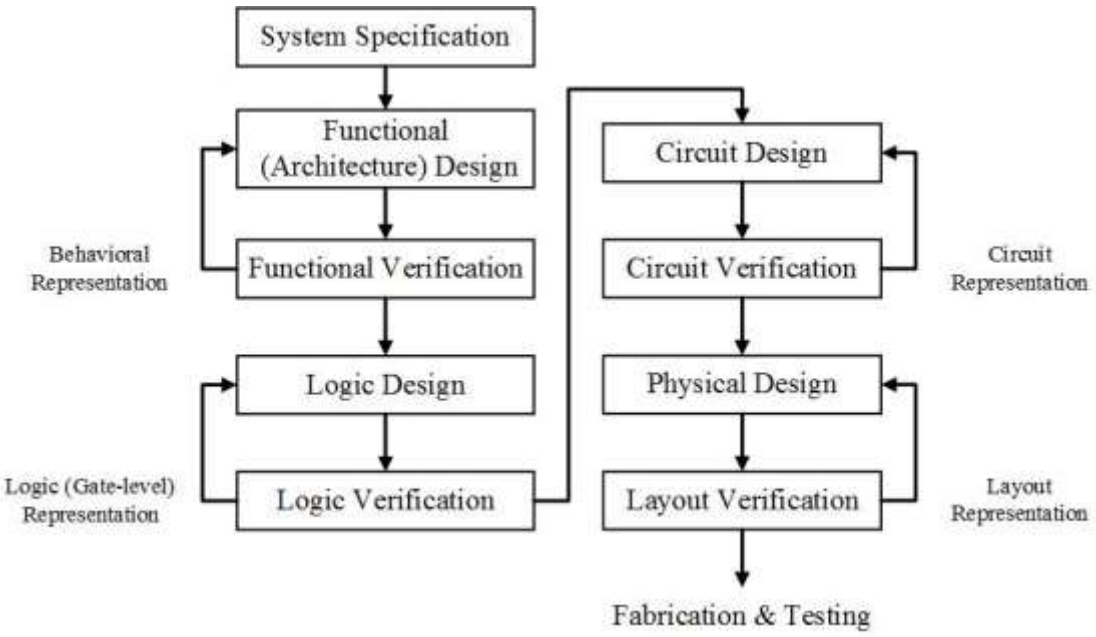


Figure 4: Simplified VLSI Design Flow

Shown in figure 4 is the simplified VLSI design flow diagram.(Debashish Mohapatra 2002) This is a generalized flow chart depicting a typical VLSI design workflow from system specification to fabrication and actual chip testing. Basically, there is no single EDA tool that can do everything but there are tools such as Synopsys (although they are of different modules) that can handle a large number of design stages. Certain stages of the design can be handle by several different software, depending on the requirement and needs of the developer. However, compatibility issue occurs when the output of a tool cannot match the input of the tool in the next stage. Translation between files or abstract layer will be needed.

Whist software vendors constantly release updates of tools with improvements and additional features, it is difficult to keep track of files compatibility as older design files may not work well in new version of the same software. Besides, there are a number of EDA tools available on the market which overlaps each other's job but may have compatibility issue when not executing on the original software. Furthermore, a user opening an existing design file written by someone else may not know which tool or version it was intended for. Thus, there is a need for certain form of automation in associating project to the correct tool and version to increase the productivity of designers.

2.2.3 Data Synchronization Solution

Data synchronization software is often confused with data replication software. These two terms are often argued as the same. In the case of data synchronization, files on multiple host are updated regardless of where the changes were made. This means that there may not be a central server where all the backup sites have to replicate. On the other hand, data replication functions more like a redundancy backup solution where the target machine for replication cannot modify the files. A tape drive back up solution is a good old example of data replication. The job of the target machine is solely to backup and only to be used when data recovery is needed. These software are typically segmented into two different groups; supports enterprise server level backup and only supporting common end-user personal computer with the former typically being commercial.

Synkron(Synkron 2011), Create Synchronicity(Create Synchronicity 2012), Dropbox(Dropbox, Inc 2008) and Ubuntu One(Ubuntu 2005) are examples of file synchronization software which do not support running as server backup solution.

These software performs basic operation such as allowing syncing of files between multiple devices. Synkron and Create Synchronicity allows file to be synced directly within the same Local Area Network (LAN) while Dropbox and Ubuntu One needs internet connectivity to function as files will be saved in the respective server and transferred to end devices. Both the latter two software are provided as a service by their respective vendors. In this case, data privacy will be an issue as the files are hosted on a remote server which users has no access nor control.

ViceVersa(ViceVersa 2001), GoodSync(GoodSync 2010) and PeerSync(Peer 2012) are examples of file synchronization software that allows operation between servers and workstations. These systems ensure that directories and files are being mirrored in real-time. Effectively, these software can run on typical household workstations although it is an overkill and licensing exceptionally expensive. This category of solutions are capable of being installed on the cloud and scales reasonably well when the number of device increases. However, these features come together with a hefty price tag which is beyond the budget of a typical university.

Typically, file synchronization software has two different methods of comparing and synchronizing file. The first method involved copying the entire file from the source to destination directory while the second method deploys differential method known as deltas. This differential technique compares the newly modified file to the original file, only the difference between the file together with the overhead will be transmitted via the network. There are two different method of comparison; binary or line/text based. The former helps in conserving huge among of space in handling binary files such as picture while the latter is suitable for text based files such as Verilog file or netlist.

2.2.4 Remote Access

"In computer networking, remote access technology allows logging into a system as authorized user without being physically present at its keyboard. Remote access is commonly used on corporate computer networks but can also be utilized on home network".(Bradley Mitchell 1996) Two popular examples will be the telnet and secure shell (SSH) network protocols. Telnet provides access to a common-line interface on a remote host. Commands can be executed on the targeted machine (such as server or router) at a remote location. SSH does the same job, except that it employs additional security measure by leveraging on encryption technology. These two methods of remote access are not suitable for this project as GUI over network is not supported.

Among the simplest yet popular remote access is the built-in Remote Desktop Connection(Microsoft 1975b) in the Windows(Microsoft 1975a) operating system (OS). This tool allows Windows OS user to access and control the desktop of another computer with Windows installed from a remote location via the local area network or internet. Unlike telnet and SSH mentioned previously, Remote Desktop Connection supports actual visual of the remote desktop. Since this is a proprietary tools bundled with Windows OS, UNIX machine will not be able to make use of it.

X Window System (X.Org Foundation 2004) is a relatively old network protocol that provides a basis for GUI and input device capability for networked computers. The main disadvantage of X Window is that a session on one X server cannot be migrated to another server easily. This also means that session cannot be continued without needing the application to be restarted. Besides, network traffic between the X server and remote clients are not encrypted by default which makes it easy to be intercepted with packet sniffer.

"Virtual Network Computing (VNC) is a technology for remote desktop sharing. VNC enables the desktop display of one computer to be remotely viewed and controlled over a network connection."(Yazdanipour et al. 2012, p. 1–5) Unlike X Window System, VNC supports virtual session which allows user to reconnect to the server and continue previous session without needing to restart the application. The advantage of VNC protocol is that pixel based information are being transmitted. This leads to great flexibility where any type of desktop can be displayed without much issue. However, bottleneck in networking will affect the performance of VNC application as transmitting raw pixel data requires heavy bandwidth.

2.3 Review of Related Publication

Steve Hodges and Peter Rounce wrote about certain common problems in VLSI Design especially in reference to the design methods and tools used in the Computer Science department at University College London(Hodges and Rounce 1991). The paper presents a list of problems such as tools compatibility issue with different hardware as well as between each tools (tedious conversion process needed), manual invoking of tool increases prone to error and high tendency of tools crashing when left to run for a long time on a large design.

The paper suggested some requirement and specification of a digital design environment such as introduction of a system capable of automatically backing design to be used at later date. Besides, the importance of a user interface is also mentioned as to not allow user to manipulate tools and files directly. Design environment should be flexible and totally configurable so that designers can incorporate new tools easily. The design environment should also be capable of enforcing the designer to follow the design stages accordingly (in correct order). Revision Control System (RCS) is a must

to ensure that every version of the design is available as reverting back from a bad change may become necessary. Furthermore, the paper mentioned that RCS may also prevent unauthorized user from making changes to design as all changes need to be commented into the system.

Steves and Peter gave a very good overview of common design workflow issue as well as some proposed solution. The key point is to remove unnecessary and repetitive work from the designer so that they can concentrate on the design itself. However, this paper describes the requirement very briefly and does not have enough details on the implementation technique. Since the paper was published decades ago, it also did not mention about the system capable of library and design sharing between different sites or education institution.

Chander S. Sarna et al. gave a detailed but narrow description on the EDeN Data Management System which includes concurrency control, checkin/checkout mechanism, associations control, cell-to-subcell relations control, audit trail, cell catalog and version control(Sarna, Reddy and Hsieh 1986, p. 36–40). The paper describes the workstation configuration of such system together with the project structure. User on workstation operates by obtaining the cells from the host by a mechanism called checkin/checkout (Bancilhon, Kim and Korth 1985, p. 25–33).

The papers also explains about how cell cataloging of various cell attributes are performed. A status field is also used to record the current status of each particular cell in order to recover from a broken link between the host and workstation. This paper also describes the consistency of cell hierarchy. However, this paper is so specific to the point that it has no benefit to design environment at educational level. Furthermore, this system is overly complex and only the industry (with plenty of resources) is capable of employing such sophisticated techniques in design environment.

Kirk Sherhart et al. provides a more general description of the EDeN Data Management System inclusive of the model, hardware system, user interface and engineer's view(Sherhart, Vershel and Owen 1984, p. 466–472). The paper mentioned of engineering productivity lost due to conflicting interactions between multiple designers on a project. Thus, the EDeN uses separate directories for different projects, common data are shared via maintaining private project-specific areas.

The system described in the paper seems to be near to flawless, except that it is not readily available to the general public not educational institution. In fact, it is an in-house tool developed in the early days for design work at Intel Corporation(Sarna, Reddy and Hsieh 1986, p. 36–40). Reproducing the exact same system for academic purpose is an uphill and almost difficult task due to the lack of manpower. Thus, this project introduces a similar but simpler system for use at educational institution.

CHAPTER 3: METHODOLOGY

3.1 Technology and Tools

This design environment project is developed in Linux OS running on VMware virtualization platform using VMware player (VMware n.d.). Running the operating system (OS) virtually allows multiple machines to be running logically at the same time without requiring many physical machines which is costly. Furthermore, network simulation can be performed by running the virtualized OS via bridged network configuration. These are the software technology and tools deployed in the development of our design environment.

Host OS: Linux, CentOS

The entire development is performed on Linux operating system. Unlike the common Windows for desktop PC, Linux is a networking OS which allows multiusers at the same time. The CentOS (CentOS n.d.) distro have been chosen due to its less frequent stable releases which makes it easier to maintain from the design environment administrator point of view. Besides, many industrial EDA tools is known to only work on UNIX like OS and requires long list of environment variables. Furthermore, Linux has built in job scheduler which is also known as crontab (Anon n.d.).

Revision Control: Mercurial

Design files such as schematic, netlist and layout files need to be revision controlled. Mercurial have been chosen due to its distributed nature which fits our application well. Besides, open-source custom locking mechanism have already been developed,

utilizing the plugin will save time on redeveloping the complicated locks which deals with system calls to ensure atomicity.

Repository Hosting: SCM-Manager

In order to allow cross-site population of repositories, hosting of the repositories will be necessary. Although mercurial has built-in web server, it is not use as each webserver can only handle a single repository. This means that sharing multiple libraries will require multiple web servers running simultaneously which is not resource efficient. SCM-Manager (Sebastian n.d.) allows a single server to serve multiple repository regardless of the logical location of the files.

Remote Access: Virtual Network Computing (VNC)

The only way end user (designer) will be able to access the computation machine/server with GUI available is via VNC. Fortunately, VNC server is fully compatible and work excellently on the Linux OS. From the user point of view, it doesn't matter what OS they are running as VNC clients runs on Windows as fine as on Linux. In order to connect to a remote machine, user will need just the Internet Protocol (IP) address (or domain name) and login (username and password) information.

GUI: Java with IntelliJ IDEA IDE

Java is an open source high-level programming language which has many documentation available. Besides, java programs are able to run on almost any platform as long as the Java Runtime Environment (JRE) is enabled. This allows the program to be deployed easily without needing to worry about the host operating system.

Furthermore, the EDA tool that is integrated in this environment is open source and written in java. Java Development Kit (JDK) is used for development.

Web service: Apache with PHP Hypertext Preprocessor

HTTP server is required to publish libraries information which can be easily accessible through a web browser. This web interface allows user to monitor the status and health of each library. Besides, user may click on a button to rerun test for selected libraries. Thus, PHP server-side programming is used to pick up information from the database and display to user. PHP also allows alteration of values on tables (for daemon script to pick up).

Database: MySQL

Database is used to store project and library specific information such as list of cells. Besides, status of verification tests are also stored in database to enable quick retrieval of information which is then summarized and displayed to user. MySQL have been selected as the database management system (DMS) as it is open source and free.

EDA: Electric VLSI

Electric VLSI have been chosen to be the main EDA tool supported and fine-tuned for this project as it provides many features for an open-source and free tool. In this project, EDA tool with command line interface is needed which sadly cannot be provided by EVLSI. However, the source code written in Java is easily available which allows certain modifications for this particular project.

Scripts: Perl and Tcsh

Perl have been chosen as the project main scripting language due to its flexibility and power in text file manipulation whist not compromising on system administration. Perl is known as “the Swiss Army chainsaw of scripting languages”. (Doug Sheppard n.d.) Tcsh was chosen as the command language interpreter which works hand-in-hand with Perl.

3.2 Methodology

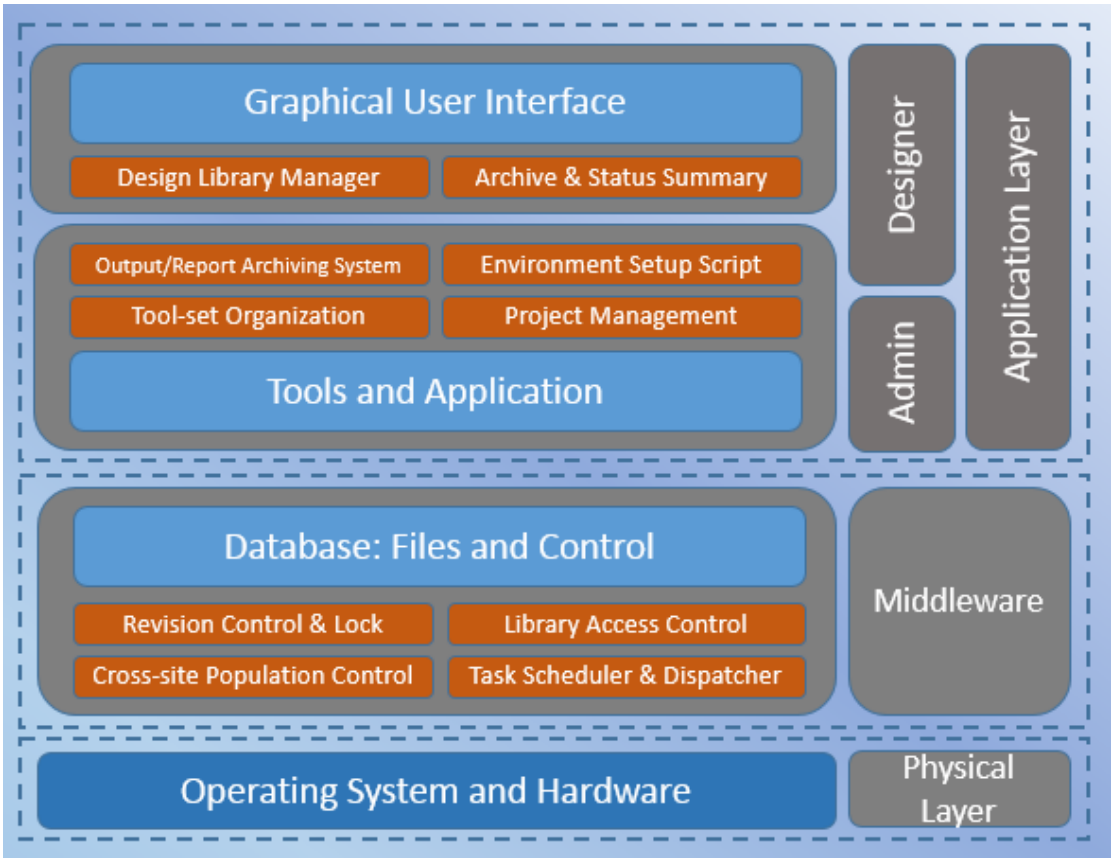


Figure 5: Design Environment System Architecture

Figure 5 depicts the system architecture of the design environment. Typical designer will only need access to the GUI interface and, output/report archiving system and environment setup script. Advanced user such as design environment administrator may access project management and tool-set organization module. The middleware layer is totally transparent to user. Library access control system seats here. This layers takes care of populating the changes to other server, ensuring files are revision controlled and provides a locking mechanism which operates based on user check-in/check-out via the design browser. Cross-site population control and revision controls relies on the task scheduler and dispatcher.

Figure 6 depicts a simple network setup comprising of three sites based on logical (such as university) segmentation. Each developer will access their respective server for the design environment. Hosted on each server is the revision controller, remote-access manager, EDA tools and cross-site population manager. Besides, basic authentication will be needed in order to enter the design environment.

Main repositories belonging to developers assigned to that particular server will also be stored in the server. On the local level between the developers and servers, it is deemed as a centralized model as developer must be connected to the server in order to work. User will be able to checkout files for modification (files will be locked) or download a copy for read-only purposes.

Looking at the bigger picture, each servers will be connected to each other via the internet. There will not be any central server as they are setup in a distributed manner and any changes made on any server will be propagated to the rest. If two more changes from different servers were attempted to be propagated at the same time, all the servers will have to communicate to resolve the conflict automatically. If automatic resolution is not possible, the respective developers on each of the server will be notified.

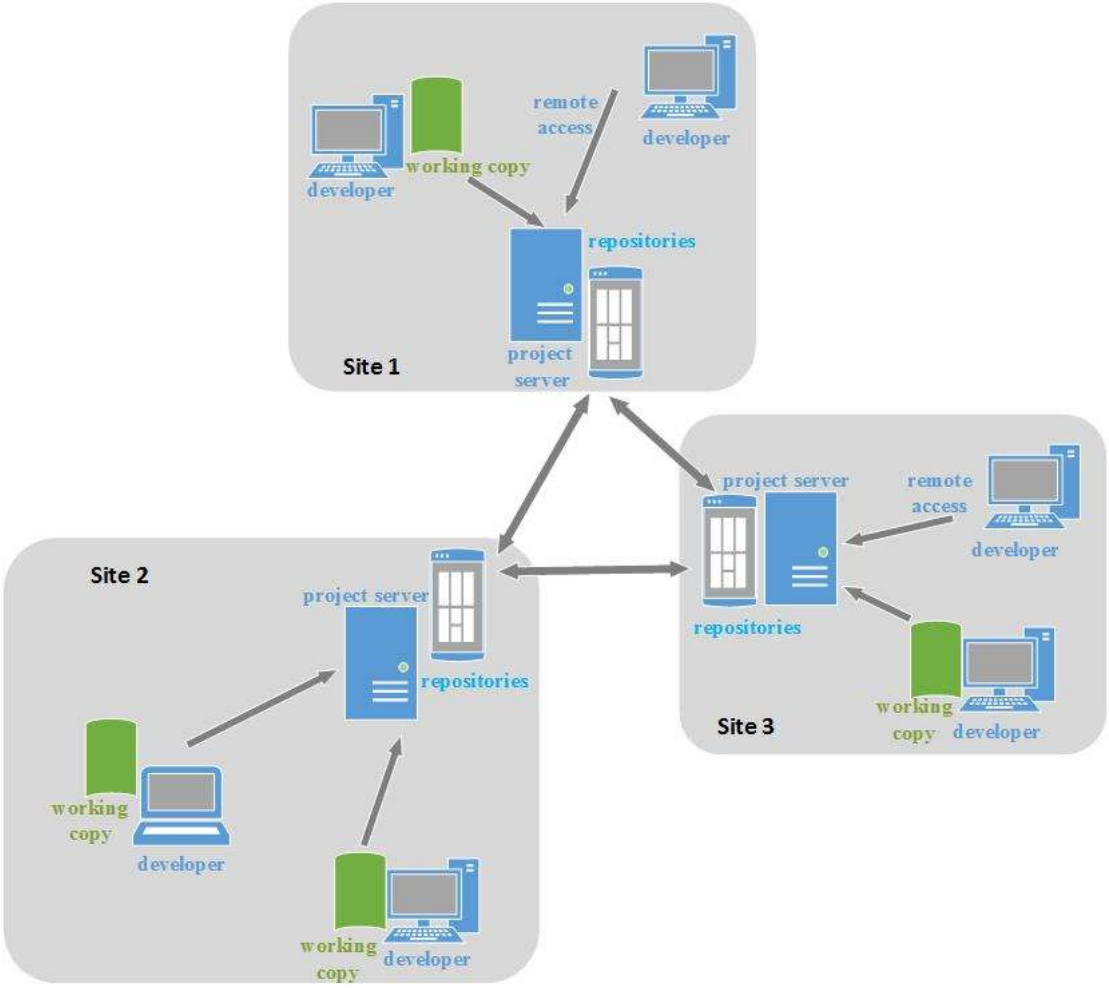


Figure 6: Design Environment Network Architecture

3.3 Requirement Specifications

3.3.1 Use Case Diagram

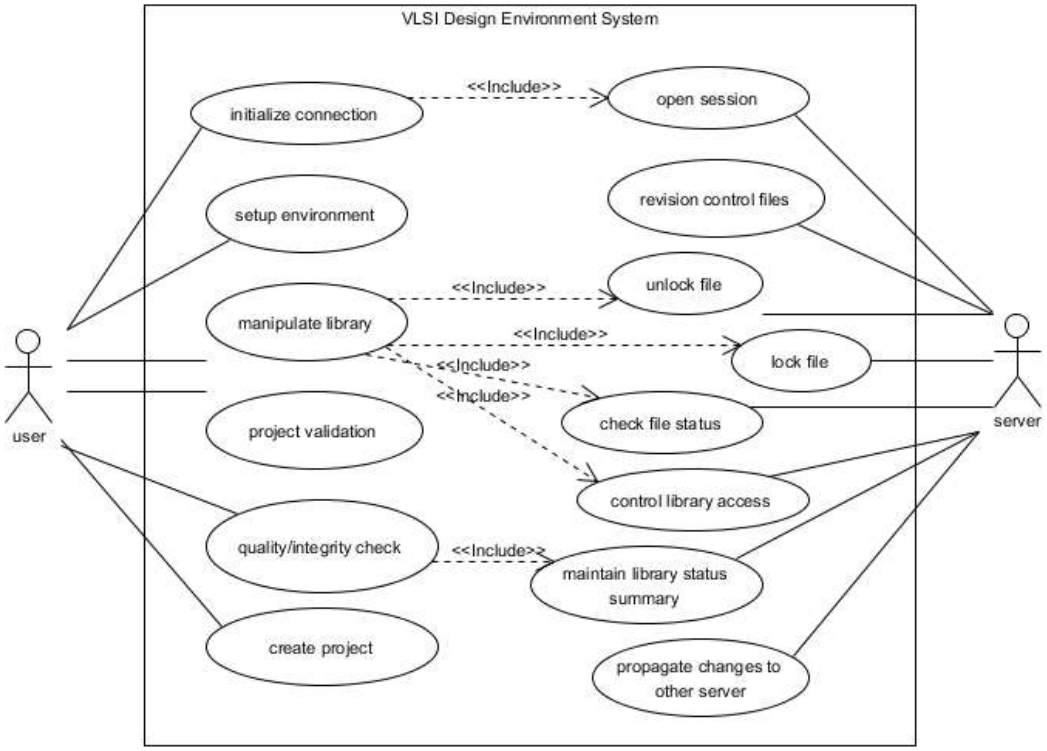


Figure 7: Use case diagram of VLSI Design Environment

The use case diagram of the VLSI design environment is depicted in Figure 7. User are allowed to initialize connection (connect to the design environment system), setup environment, manipulate library, perform project validation and cross-checking with quality/integrity checker. Project creation is only applicable for user with elevated permission. On the other hand, the project server will open session when user initiated session and setup environment. Locking and unlocking of file are taken care of by the project server. Running in the server is also the revision control daemon which is in charge of propagating the changes to other project server. File status checking mechanism is maintained to ensure speedy library manipulation operation.

3.3.2 Sequence Diagram

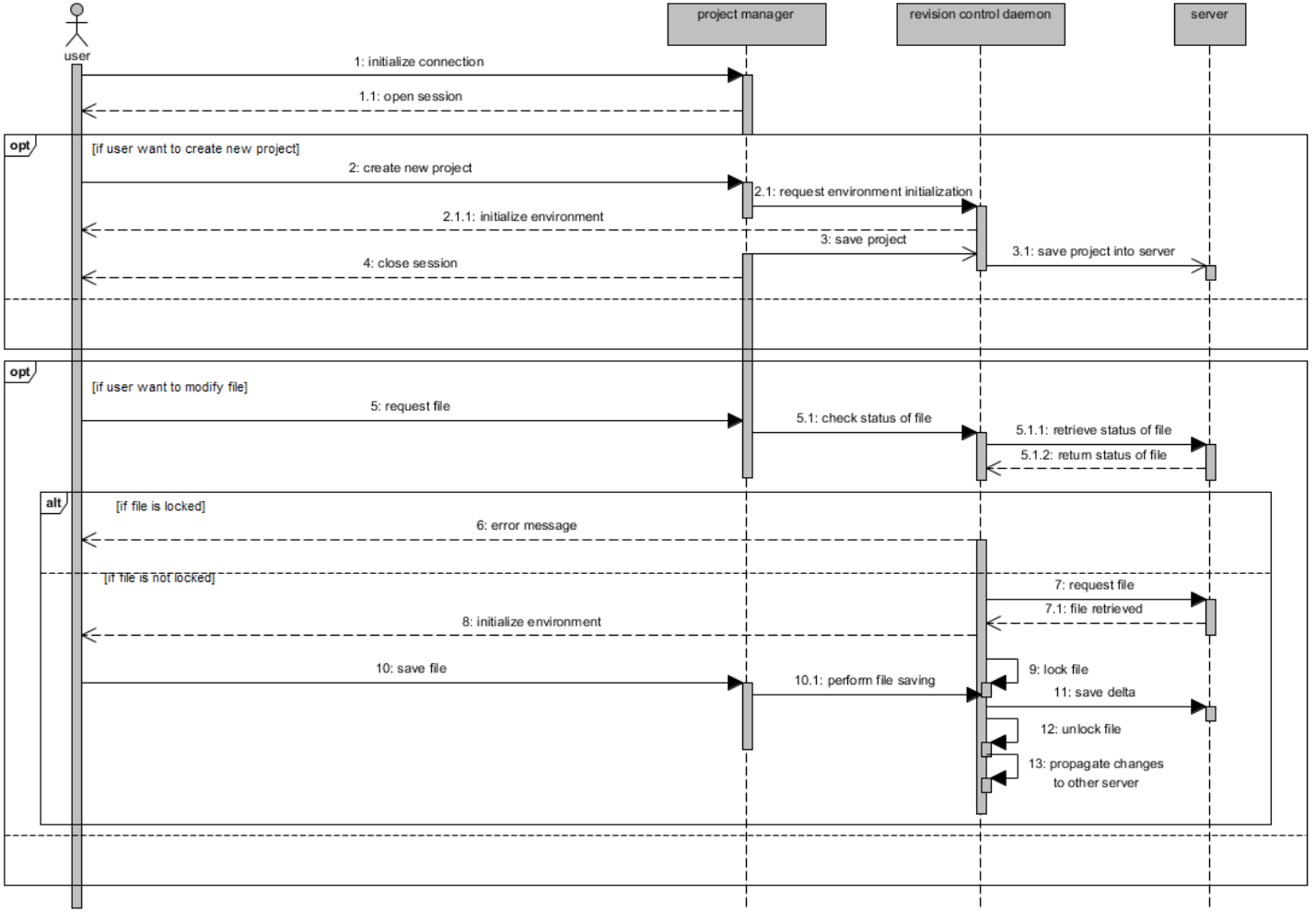


Figure 8: Sequence diagram of proposed VLSI Design Environment

As shown in the sequence diagram in Figure 8, user will need to initialize connection to the project manager (project server). Once connection is successful, session will be opened. User can then choose to create a new project or work on an existing project. If a new project is to be created, the project manager will request for an environment to be setup based on project type and placed under revision control.

Project creation is normally done by the project manager (human intervention), not user. Normal user typically will work on existing project. Thus, after joining a session, user will navigate the library and request to modify certain file. The project manager will check with if the file is being locked by other users. If yes, then an error message will be returned. Else, user will be able to work on the file and the file being locked.

After completing the modification, user invoke the save operation. For every file save, an entry will be placed in the revision control software to keep track of changes. The file will also be unlocked when user quits working on the project so that other user will be able to work on it. Daemon tool running in the background will attempt to synchronize the changes with other servers sharing the same library.

3.4 Timeline

This project have been completed phase by phase. This ensures a systematic workflow in ensuring the system is completed successfully in a tight time frame.

Phase 1

Design environment infrastructure tools and dependency such as Python, RCS, SCM-manager, MySQL and VNC server up and running. Project management capable of creating project (manually), initializing environment and tools setup.

Phase 2

Working from phase 1, mercurial customization is given the locking mechanism. Wrapper scripts have been written to check library access control before invoking mercurial commands. Design library manager's GUI drafted and not all function implemented as yet. EVLSI source code modification to enable command line option completed in this phase. Completed scripts to merge and split cells from library file.

Phase 3

Continuing from phase 2, Apache webserver up and running. Completed the tools test flow script and quality checker script. This phase were focused on working with web interface.

Phase 4

While previous three phases were focused on getting local environment. Rough simulation of network topology being carried out in phase 4. The final phase was used for testing and minor bug fixing.

ID	Task Name	Start	Finish	Duration
1	Discussion of Project Proposal and Title	5/27/2013	5/31/2013	5d
2	Problem and Objective Formulation	6/1/2013	6/7/2013	7d
3	Research and Literature Review	6/1/2013	6/22/2013	22d
4	System Requirement Analysis	6/22/2013	7/5/2013	14d
5	Preliminary Report	7/2/2013	7/19/2013	18d
6	Submission of Preliminary Report	7/19/2013	7/19/2013	1d
7	Enhancement of Project 1 Report	7/20/2013	8/19/2013	31d
8	Finalization of project 1 report	8/19/2013	8/19/2013	1d
9	Poster preparation	8/20/2013	8/25/2013	6d
10	Oral Presentation of project 1	8/26/2013	8/26/2013	1d
11	Poster Submission	8/26/2013	8/26/2013	1d

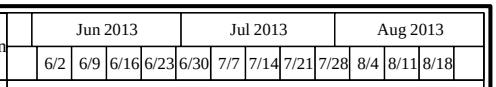


Figure 9: Timeline of Project 1

ID	Task Name	Start	Finish	Duration
1	Project 1 Report Review	10/1/2013	10/2/2013	2d
2	Setting up development environment	10/3/2013	10/9/2013	7d
3	Design and implementation of Phase 1	10/10/2013	11/1/2013	23d
4	Phase 1 review	11/2/2013	11/8/2013	7d
5	Design and implementation of Phase 2	11/9/2013	12/1/2013	23d
6	Phase 2 finalization	12/2/2013	12/9/2013	8d
7	Design of Phase 3	12/10/2013	12/20/2013	11d

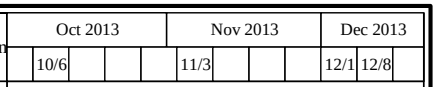
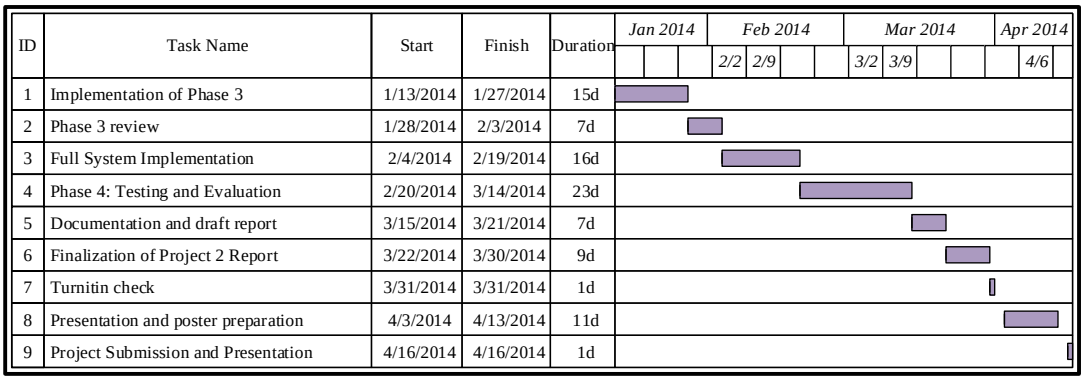


Figure 10: Timeline of Development



ID	Task Name	Start	Finish	Duration
1	Implementation of Phase 3	1/13/2014	1/27/2014	15d
2	Phase 3 review	1/28/2014	2/3/2014	7d
3	Full System Implementation	2/4/2014	2/19/2014	16d
4	Phase 4: Testing and Evaluation	2/20/2014	3/14/2014	23d
5	Documentation and draft report	3/15/2014	3/21/2014	7d
6	Finalization of Project 2 Report	3/22/2014	3/30/2014	9d
7	Turnitin check	3/31/2014	3/31/2014	1d
8	Presentation and poster preparation	4/3/2014	4/13/2014	11d
9	Project Submission and Presentation	4/16/2014	4/16/2014	1d

Figure 11: Timeline of Project 2

3.5 Testing

There are several methods of software testing, namely black box, white box and grey box testing. Black box testing refers to testing without having access to source code nor knowledge of internal workings of the application. On the other hand, white box testing is the detailed investigation where tester needs to possess the knowledge of the internal working of the code. Grey box testing is the mixture of both where the tester has slight knowledge on the internal workings.

In our project, due to limited timeframe, mainly white box testing was deployed especially from end-user point of view. However, certain test cases falls within the boundary of grey box testing. Testing is performed in the same development environment. That being said, virtual machines are used as there is no access to a dedicated server environment.

CHAPTER 4: DESIGN AND IMPLEMENTATION

4.1 Overall System Architecture

There are two main areas to the design environment's repositories; central and db_root. For each design sites, these two areas are mandatory and need to be defined by symbolic linking the path. The purpose of each area is as below:

central — This area marks the central area of a project which host the repository of each logical site.

db_root — Acronym of "database root", which holds a collection of central repository, which host repositories gathered from multiple sites.

As explained, central area only contains project repository that is owned by the particular site. On the other hand, database root host all repositories regardless of which site owns the project. The database root area of a site will pick up changes from central area of other sites. Thus, all user within the site will only communicate and alias with its local central and database root area.

In order to maintain both of these areas, a total of three scripts need to be ran. The first two scripts are run as cron job. Cron is a time based job schedule found in the Linux OS. It allows the script to be run periodically at fixed time or intervals. However, for testing purposes, the script can be executed manually to observe the inner workings. Central area is maintained by the script know as central cron. The central cron observe the database for changes, such as when a project or library is added. The cron then picks up the changes and created the directory and mercurial repositories in the central area.

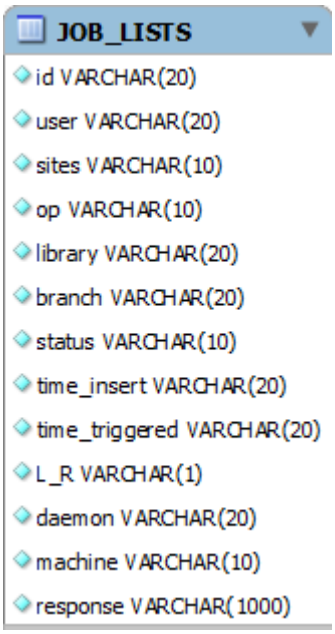


Figure 12: job_lists Entity Diagram

After the central cron have acknowledge the changes, extra commands will be placed in the job_lists table as new jobs that will be picked up by database root cron. As shown in figure 12, the job_lists tables contains entries use to track the job. Jobs are tracked based on insertion time, unique ID, user, library name and command (placed in response column). Once the job have been performed, the status field will be updated. The name of daemon who successfully completed the task will also be updated. The jobs are mainly mercurial commands that will clone and pull the repository from other site's central area.

Changes made to the library (such as check in or check out) will also be performed as jobs. The job dispatcher script running as a daemon will monitor changes on the database. If there's a new job, the dispatcher will assign the task to a child process to be executed. Thus, each site will have its daemon running and cross checking the same table for new jobs. This means that the daemon should have minimal downtime as possible.

Each project consist of multiple directories of different section. There are four main section specific to a project:

setup_area — Store all scripts related to maintaining and setting up the particular project. The setup script is located in this directory.

tools_area — Contains all environment tools as well as project specific tools. Consist of two sub-section, cad_root which stores design environment infrastructure related tools while proj_tools contains project specific tools.

work_area. Mounted disk or directory for all user workspace area based on group assignation.

archive Contains successful test log files. Log files generated when user perform validation step (section 4.4).

Each of this section is represented by a respective environment variable after setup have been performed (refer to section 4.2). Due to the nature of how disk are managed in the industry, segmentation of design environment into these areas are necessary. This means that the design environment may be distributed over many physical disk but appears to be consolidated logically.

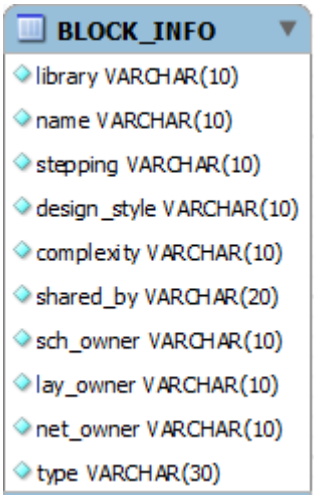


Figure 13: block_info Entity Diagram

Figure 13 depicts the block_info entity diagram. This table is used to maintain a list of library. Thus, addition of library or introduction of new stepping have to be performed by appending the new info into this table. Mandatory information is library and stepping. The rest of the columns are only for non-critical tracking purpose which can be omitted. In order to state dependency library, the three "owner" columns can be utilized. As stated previously, the central cron will depend on this table to monitor if there is new library or stepping added.

Figure 14 depicts an example of library access control file. This file is located in the setup area and defines the user permission of each library. Users are identified by its Unix username. Unix group can also be used to enable group permission policy. Any library not listed in this file is accessible to all. Library lockdown (read-access only for

all user) can be triggered by toggling the last value of each line to yes or no. The first line of the file defines superuser permission which gives write-access to all libraries.

```
*=admin
@grp_aa=boey g2 group3
http://192.168.1.106:8080/scm/hg/proj/bwd/dB/a/afub3_bwd_lay,b_
bwd%A0,cboey @grp_aa &adm,No
http://192.168.1.106:8080/scm/hg/proj/bwd/dB/a/afub3_bwd_net,b_
bwd%A0,cboey @grp_aa &adm,Yes
```

Figure 14: Example of library access control file

In this project, only EVLSI EDA tool is supported upfront. Thus, in the next few section, the design can be seen revolving around the particular tool. However, some form of customization can be performed to support other tools although not entirely straightforward.

4.2 Unix Development Environment

4.2.1 Functionality and Overview

In order to prepare the environment, user will need to open a terminal, navigate to the setup area and invoke the setup command together with the relevant switches. Example usage of this command:

./setup -p <project> -cfg <version> -t <tool> -b <group> -w <workblock> -lib <library>

Below are the explanation of each switch and argument:

./setup — Setup script written in Perl which will invoke (by sourcing method) other scripts to setup the environment. It will first check

	if setup have been performed before, if no, then only proceed to perform setup. The setup script will also check if all the arguments are valid, such as if library or project specified by user is in existence. Exit and display error message on output stream if otherwise.
-p <project>	Specifies the project that user intends to work on. Only valid projects that have been added by design environment (DE) admin can be specified. Ordinary end user may not create project by simply specifying the desired project name as an argument to the setup script.
-cfg <version>	Tool configuration switch request the setup script to check for dependency tool package that is required by the specific project. The dependency file can be found in the setup area specifying which version of each tools should be used.
-t <tool>	The tool switch provides additional flexibility for user to invoke extra tools not included in the configuration file. The tools that is specified here must already be installed in the tools area for the setup script to work.
-b <group>	The group switch specifies the specific directory that have to be used by the particular user based on pre-assigned group.
-w <workblock>	The work block specifies the directory and session that user wants to enter. If user had prior unfinished work, user can specify the same work block and the previous session will be continued as the work area will be same.

-lib <library>	Library switch specifies the libraries that is required by the user. If multiple libraries is needed, user can use the colon to separate the library names (i.e. lib1:lib2:basic).

After the script have been ran, if successful, the user is considered to have entered the design environment. The current work directory will also be known as the work area root directory (WARD) which follows this structure:

<pre-assigned_disk_path>/<group>/<username>/<workblock>

```
[boey@localhost test1]$ pwd
/DE/tmp/utar_adm/disk1/grp1/boey/test1
[boey@localhost test1]$ echo $WARD
/DE/tmp/utar_adm/disk1/grp1/boey/test1
```

Figure 15: Current directory after successful setup

User will now be able to invoke the project specific tools by entering the command in terminal. All user preferences and tools temporary area will be the current session's WARD.

4.2.2 Implementation

When user enters the "setup command", the setup script written in Perl will firstly be called. The script will check if setup have already been performed previously by checking the value of environment variable (EV) STP_DONE. If the variable is not equal to 1 or doesn't exist, setup will proceed. The setup area EV will be set. Next, project configuration script will be invoked. Certain arguments that have been passed on to the setup command such as project and library name will be check for validity. If there is no problem with the arguments, then the environment variables will be created before other scripts are being called. The project configuration script will also check for present of certain directory path such as setup area, tools area and work area.

```
#!/usr/bin/perl -w
use strict;
our %project_hash=(
 'R0P0V0'=>{
    'TOOLS_STP_REV'  =>'1.0',
    'TOOLS_PARENT'   =>'NONE',
    'TOOLS_CFG'      =>'R0V0P0',
    'STEPPING'       =>'A0',
    'MODEL'          =>'BWDA0latest',
  },
};
```

Figure 16: Example of project map file

After project configuration completes, the tool setup script will be invoked. The project map file will be read to determine the tools configuration version. Project map files stores the configuration revision number and the corresponding stepping and model that is easily read into Perl's hash function. Then, based on the map file, the exact version of tool configuration file will be opened to determine all the tools and corresponding version to be loaded into the environment. The tool configuration file stores the tool name and the tool version to be used.

```
tool1       1.4
tool2       1.3
toolEVLSI   1.1
tool7       1.1
```

Figure 17: Example of tool configuration file

Setup will be done by sourcing several files in the tools folder in the particular order of pre-Setup, Setup and post-Setup. The work area root directory (WARD) will be created. All the necessary tools directory will also be created in the WARD.

```
afub1_bwd_lay BWDA0latest   LAY:afub1:afub2LAY
afub1_bwd_sch BWDA0latest   SCH:afub1:afub2SCH
afub2_bwd_lay stableRelease LAY:afub2
afub2_bwd_sch customTag     SCH:afub2
basic_bwd_lay  BWDA0latest  LAY:basic
basic_bwd_sch  testTag      SCH:basic
```

Figure 18: Example of model file

Next, another script will be called to generate the list of libraries and its corresponding path. Note that the list of libraries here refer not to the complete list, but only the list requested by user via the switches specified during setup command. Library path is dependent of the model file which states the library dependency and valid default tag to be used. A file which follows the cds.lib.<session id> prefix will be created. This file defines the loaded libraries specified by user and the logical directory path. Session id is borrowed from the running script process id (PID). This file will be relied on by the design library manager in the next phase.

4.3 Design Library Manager

4.3.1 Functionality and Overview

The Design Library Manager (DLM) is a GUI that allows user to manage design cells without needing to know or understand the underlying command. The DLM key area is the navigation panels that consist of 6 column/panels. The columns display information in this particular manner:

| library_project_type | tag | category | cell | view | version |

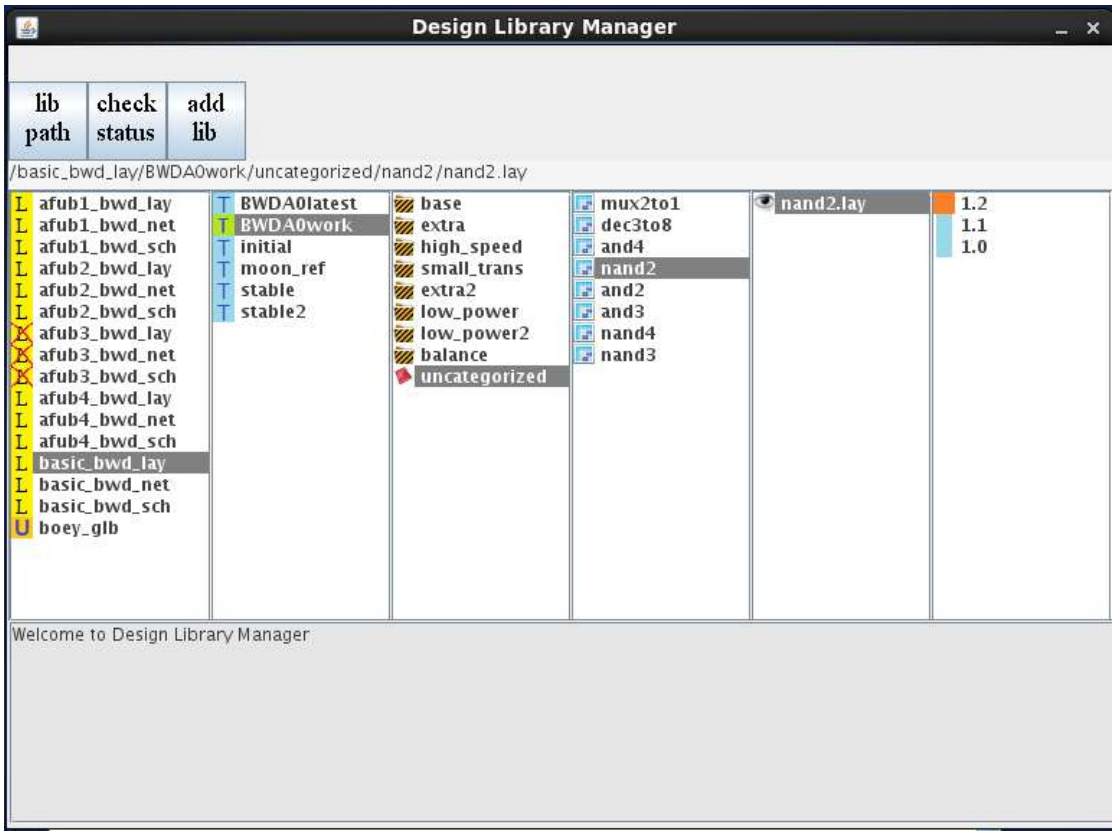


Figure 19: Screenshot of Design Library Manager

The single row text area at the top of the DLM display user current navigation path while the text message area at the bottom displays the output stream of the underlying running script. It allows the DLM to notify user on the status of each action.

Each cell in the column can be right clicked for more options, explained as below:

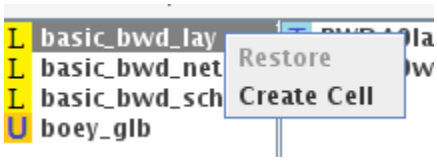


Figure 20: Library -Right click menu

Library_project_type (first column)

Restore	Restore previously removed cell from the selected library. User will be prompted to enter the removed cell name.
Create Cell	Create new cell in the selected library.

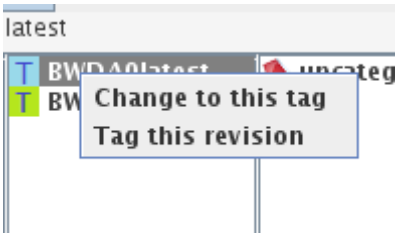


Figure 21: Tag -Right click menu

Tag (second column)

Change to this tag	Change the working tag to the selected tag. Read only, no write allowed on non-work tag.
Tag this revision	Tag (snapshot) the currently selected library. Only latest tag can be tagged.

Category (third column)

There is no right click option available in this column.

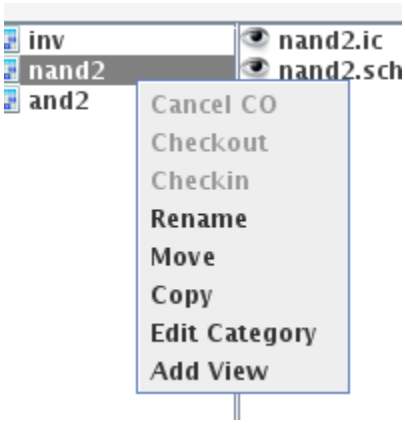


Figure 22: Cell -Right click menu

Cell (fourth column)

Cancel CO	Cancel checkout to unlock the cell. Can only be performed by user who previously checkout the particular cell.
Checkout	Checkout all views in the selected cell. All the views will be locked and no other user allowed to make modification.
Checkin	Checkin/save changes of all views. This is performed after making modification to cell/view.
Rename	Allows user to rename cell. Renamed cell remains in same library.
Move	Allows user to move cell to other library. User must have write access to destination library.
Copy	User may duplicate cell to another library. However, user will be prompted for a new cell name as a single project may not consist of cells with same name.
Edit Category	Allow user to edit the category of selected cell.

Add View Add new view into the selected cell.

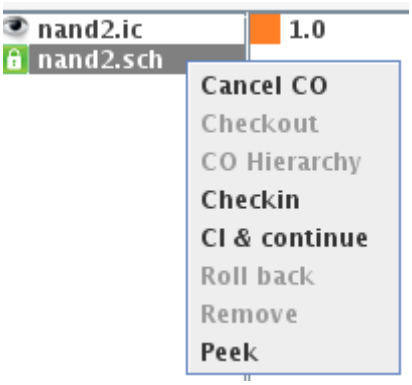


Figure 23: View -Right click menu

View (fifth column)

Cancel CO	Cancel checkout selected view. Can only be performed by user who previously checkout the particular view.
Checkout	Checkout selected view. View will be locked and no other user is allowed to make modification.
CO Hierarchy	Selected view and all dependency views will be checked out.
Checkin	Checkin selected view after making changes.
CI & continue	Checkin selected view but keep copy in WARD.
Roll back	Restore selected view to previous version. User will be prompted to key in the desired revision number.
Remove	Remove selected view from library. View will not be physically deleted and it can be restored.

Peek — Read latest version of selected view without checking out. Read-only view will be copied to WARD.

Figure 24: Version -Right click menu

Version (sixth column)

Read — Display selected view (read only) in text editor.

There are three buttons on the top panel: lib path, check status and add lib.

lib path — Display a list of library and its directory path in a new window. (Figure 25)

check status — Opens another dialog which prompts user to select from the list of available libraries. (Figure 26) A status summary of all views in the chosen library will be displayed. Besides, user is able to perform bulk action such as check in, cancel checkout or ignore views. (Figure 27)

add lib — User will be prompted for additional library to be included. This feature is useful in cases where user forgotten to specify the library during the setup phase.

Library	Path
basic_bwd_lay	/DE/test/p/proj/bwd/db_root/bwd/basic_bwd/lay/BWDA0work/basic_bwd_lay
basic_bwd_net	/DE/test/p/proj/bwd/db_root/bwd/basic_bwd/net/BWDA0work/basic_bwd_net
basic_bwd_sch	/DE/test/p/proj/bwd/db_root/bwd/basic_bwd/sch/BWDA0work/basic_bwd_sch
boey_glb	/DE/test/p/proj/bwd/db_root/glb/boey_glb/std/work/boey_glb
afub1_bwd_lay	/DE/test/p/proj/bwd/db_root/bwd/afub1_bwd/lay/BWDA0work/afub1_bwd_lay
afub1_bwd_net	/DE/test/p/proj/bwd/db_root/bwd/afub1_bwd/net/BWDA0work/afub1_bwd_net
afub1_bwd_sch	/DE/test/p/proj/bwd/db_root/bwd/afub1_bwd/sch/BWDA0work/afub1_bwd_sch
afub2_bwd_lay	/DE/test/p/proj/bwd/db_root/bwd/afub2_bwd/lay/BWDA0work/afub2_bwd_lay
afub2_bwd_net	/DE/test/p/proj/bwd/db_root/bwd/afub2_bwd/net/BWDA0work/afub2_bwd_net
afub2_bwd_sch	/DE/test/p/proj/bwd/db_root/bwd/afub2_bwd/sch/BWDA0work/afub2_bwd_sch
afub3_bwd_lay	/DE/test/p/proj/bwd/db_root/bwd/afub3_bwd/lay/BWDA0latest/afub3_bwd_lay
afub3_bwd_net	/DE/test/p/proj/bwd/db_root/bwd/afub3_bwd/net/BWDA0latest/afub3_bwd_net
afub3_bwd_sch	/DE/test/p/proj/bwd/db_root/bwd/afub3_bwd/sch/BWDA0latest/afub3_bwd_sch
afub4_bwd_lay	/DE/test/p/proj/bwd/db_root/bwd/afub4_bwd/lay/BWDA0work/afub4_bwd_lay
afub4_bwd_net	/DE/test/p/proj/bwd/db_root/bwd/afub4_bwd/net/BWDA0work/afub4_bwd_net
afub4_bwd_sch	/DE/test/p/proj/bwd/db_root/bwd/afub4_bwd/sch/BWDA0work/afub4_bwd_sch

Figure 25: Screenshot of library path dialog

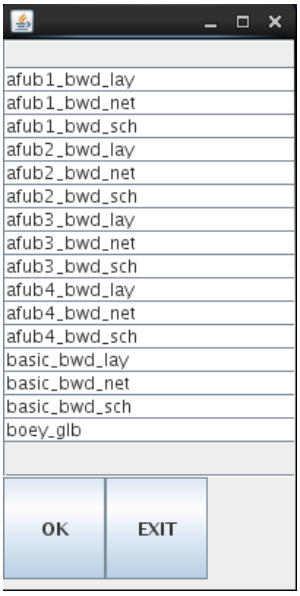


Figure 26: Screenshot of check status –library list

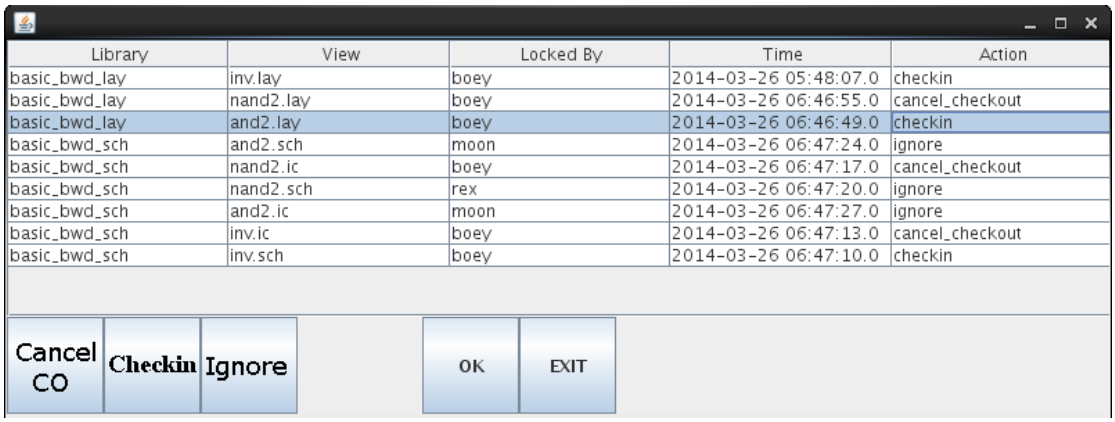


Figure 27: Screenshot of check status -bulk action

4.3.2 Implementation

The Design Library Manager (DLM) relies on reading the library path file located in the current session WARD. From the previous section, when user enters the environment, a file will be generated which stores the library and respective logical path.

```
[boey@localhost test1]$ cat cds.lib.8358
DEFINE basic_bwd_lay
/DE/test/p/proj/bwd/db_root/bwd/basic_bwd/lay/BWDA0work/basic_bwd_lay
DEFINE basic_bwd_net
/DE/test/p/proj/bwd/db_root/bwd/basic_bwd/net/BWDA0work/basic_bwd_net
DEFINE basic_bwd_sch
/DE/test/p/proj/bwd/db_root/bwd/basic_bwd/sch/BWDA0work/basic_bwd_sch
DEFINE boey_glb  /DE/test/p/proj/bwd/db_root/glb/boey_glb/std/work/boey_glb
```

Figure 28: Example of library path file in WARD

When user checkout a view, the WARD area will be scanned of existing views. If the view already exist in WARD, user will be prompted if they want to overwrite or replace

the copy in WARD. Access control file will also be check to ensure user has the write access to the particular library. User will be prompted if it does not have access and checkout will fail. After successful checkout, the view will be locked so that other user will not be able to checkout and make modification to that particular view. The view will be combined with other view of the same library in the WARD area as EVLSI requires cells in the same library to be in a single library file (.jelib).

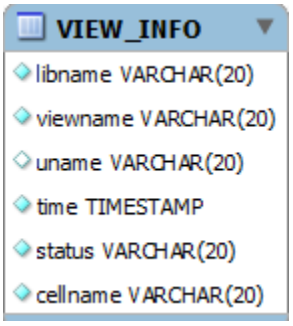


Figure 29: view_info Entity Diagram

Figure 29 depicts the view_info entity diagram. It can be seen that this table stores basic view information such as modified timestamp, status (locked or free), cell that it belongs to, username (if being locked, blank otherwise) and library name. The GUI relies on this table in order to retrieve cell status quickly instead of checking the physical lock status. This is particular effective when the number of view grows.

There are two different kind of check in, namely newly created view or previously checked out view. For the latter case, checkin can only be performed by the same user who checked out the view earlier. When user click on the checkin menu, the view will be extracted from the library file and copied to the db_root area. Copies in the WARD will only be preserved if "Checkin & Continue" menu was selected. Then, access control file will be checked to ensure user has write permission before the mercurial commit command will be placed into the job queue. If the intended view to be checked

in is not tracked yet, the mercurial add command will simply be added to the job queue. Checking-in views will also update the database's view table.

Removing cell or view is to delete the object from the library, except that it is tracked and revisable. Removing object relies on the mercurial remove command which tells the RCS to stop tracking changes to the particular file. Both restore and roll back function relies on the mercurial update command. Since mercurial track changes as a changeset and not individual files, cell version log file residing in the library directories is used to record the view and respective MD5 hash. Thus, the hg update command will be recursively executed until the desired version is found (the MD5 checksum matches) and then recommitted (check-in again).

```
and2.lay 1.0 Active 6d76035f7e1c5bcc6312920a00528d9d
inv.lay 1.0 Removed 6cfbe069154676d2ec23d9e70a2a451c
nand2.lay 1.0 Active 24f8e1132a1fa9a0470c95bab04ed5b7
```

Figure 30: Example of cell version log file

Tagging allows user to snapshot and quickly associate the current library condition to an arbitrary name. Tagging is only possible on current library development and not past history. There are two different kind of tags, user or system specified. System specified follows the format <project><stepping><latest/work>, in example BWDA2latest. User tags are tag names given by user and may not contain the "work" or "latest" keyword. In a library, there is only a single tag denoted by latest and newest stepping which is writable while other tags are only readable. When user tags a library, the library directory will be duplicated and marked read-only so that it can be referred to at a later time by using the "Change to this tag" menu option.

Cell category information is being maintained and stored in multiple files sitting in the library directory. There are two types of file identified by its extension .topcat or .cat.

There is only a single topcat file in a library listing all the category name. The .cat file is named after the category and list all the cells residing in it. Categories are only realize in the GUI Library manager, it is abstract and this means that it is only visible to user. That being said, category does not impact the implementation of design environment.

```
$ cat basic_bwd_lay.topcat
base.cat
extra.cat
high_speed.cat
small_trans.cat
extra2.cat
low_power.cat
balance.cat
```

Figure 31: Example of .topcat file

```
$ cat extra.cat
and2
and3
nand4
mux2to1
$ cat base.cat
inv
```

Figure 32: Example of .cat file

4.4 Validation and Archive

4.4.1 Functionality and Overview

At any point of time, as long as the WARD contains the library file (all dependency cells must be included), user may perform flow check by invoking the validation script. This process is invoked either when user completes a project (prior to library lockdown) or when user intend to simply ensure they are on track. If all the tests pass, the result will be exported to archive area by default. Below shows the argument that the script accepts:

val.pl -flow <tool>.<test1>:<test2> -lib <library> -cell <topcell> -checkonly

val.pl	This script is located at the setup area. It cross checks the user arguments against the tool flow configuration file before invoking the relevant tools.
-flow <tool>.<test>	The flow switch allows user to specify the tool and respective tests to be tested against the library. Multiple tests can be applied by separating with colon. The test will be performed as according to the sequence specified in the argument.
-lib <library>	Specifies the library that is intended to be checked. Library file must already be in WARD area.
-cell <topcell>	There is possibility that cell of a higher hierarchy exist in the same library file. Thus, user need to specify the top level cell that is intended to be checked.
-checkonly	If this switch is specified, the result will not be exported into the archive area.

The script will invoke the tools specified by user and runs through all the tests against the specified library in the WARD area. The result of ran tests will be displayed on the terminal. (figure 33) The log file generated from the tool will be stored in the $WARD/<tool>/validation/ directory with filename <tool>_<flow>.log. If all the test passes, log files and result summary together with timestamp will be exported to the archive area.

As far as this design environment is concerned, only EVLSI tool is supported. The tests that the said tool support are network consistency check (ncc) and design rule check for layout and schematic (drcL and drcS) respectively. However, provided the tool supports command line access, user may configure extra tools in the tool flow configuration file.

```
$ val.pl -flow evlsi.ncc:drcS:drcL -lib basic -cell and1
============LOG FILES============
evlsi_ncc:        $WARD/evlsi/validation/evlsi_ncc.log
evlsi_drcS:       $WARD/evlsi/validation/evlsi_drcS.log
evlsi_drcL:       $WARD/evlsi/validation/evlsi_drcL.log
result:   $WARD/evlsi/validation/result
=============SUMMARY=============
evlsi_drcL        pass
evlsi_drcS        pass
evlsi_ncc         pass
overall  pass
============ARC PATH=============
$ARC/lib/basic/latest
=================================
```

Figure 33: Example screen dump of validation tool script

Next, the quality check script allows user to do certain checks against the result in archive and db_root area. Below is the explanation for each check available:

Requirement	Checks the log file in the achieve area and rerun the same tests based on reassembling the views from db_root. If all tests pass, requirement test past. If any test fails, requirement check is considered to fail.
Time	Check timestamp of log file in archive area and compare with the log file in local WARD directory. Time check fails if the archive copy is found to be earlier than the latter.
Duplicate Cell	Check the library file exported to archive area and compare each cell against the database. If same cell name is found to be from different library, the duplicate cell check fails.
Content	Content check reassemble the library file from the db_root and checks the MD5 checksum against the reference in archive area.

If it doesn't match, the design isn't the latest copy or have been tempered with.

qc.pl –lib <library> -cell <topcell>

-lib <library> Library in archive area to be checked. The flow test script must must be ran prior to quality check.

-cell <topcell> User to specify the top level cell that is intended to be checked.

After the script finish running, the summary will be displayed on the terminal.

```
$ qc.pl -lib basic -cell and2
reqCheck        pass
timeCheck       pass
dupCellCheck    pass
contentCheck    pass
```

Figure 34: Screen dump of quality check script

To ease management, users are able to check the summary of all libraries through a web interface (figure 35). For failed test, clicking on the cell will pop-up the corresponding error log (figure 37). Besides the ability to view library status, user may also rerun test for selected libraries through the browser. After the rerun button is clicked, the selected library will have its status changed to queuing. Then the daemon will pick up the job from the database and rerun the test (status changed to running). On top of these, user can waive certain failed test if the particular library does not need it. User will need to supply a reason (viewable by all) when waiving test for a library (figure 36).

Figure 35: Screenshot of web-based library status summary

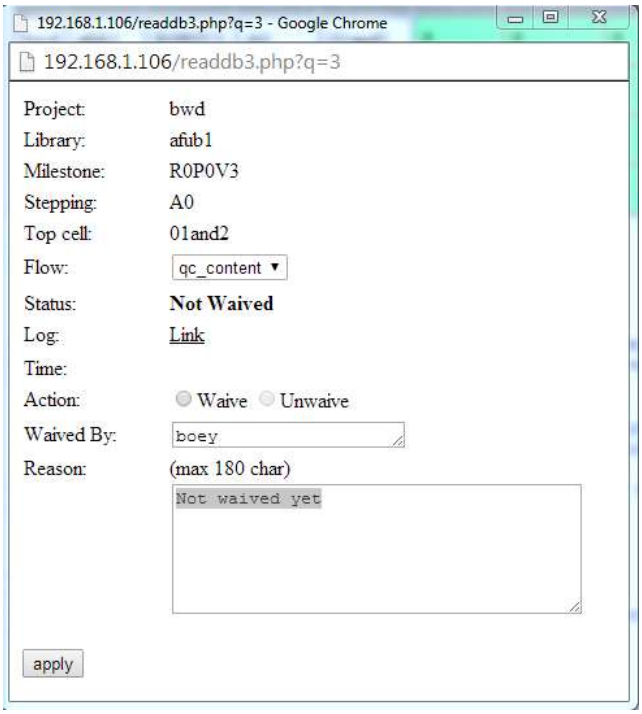


Figure 36: Screenshot of web-based waiver function

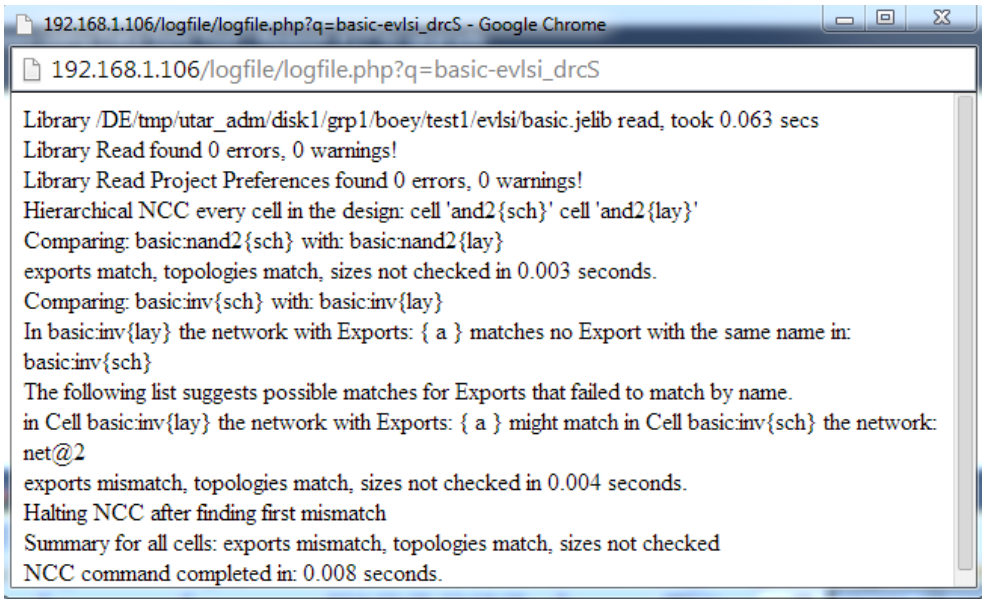


Figure 37: Screenshot of web-based error log display

4.4.2 Implementation

The essential element that allows the checks and web-based summary is that the EDA tool needs to support command line interface (CLI). Since EVLSI do not have built-in CLI ready for user, the program have to be modified based on the source code found in the internet. EVLSI is written in Java based on object-oriented coding style which relies on classes and encapsulation heavily. After the enhancement is made, EVLSI can perform network consistency check (NCC) and design rule check (DRC) via the –ncc and drc switches respectively. However, it is not necessary for user to use this commands directly as it can be performed easily via the EVLSI's GUI. Figure 38 shows a small code snippet made to the open source EVLSI tool in order to enable command line interface.

```
DRC.DRCPreferences dp = new DRC.DRCPreferences(false);
Cell cell1 = null;
String drcOp2 = (drcOp.trim().split("[.]"))[0] + "{" + (drcOp.trim().split("[.]"))[1] + "}";
if(((drcOp.trim().split("[.]"))[1]).equals("lay"))
        cell1 = Library.findCellInLibraries(drcOp2, View.LAYOUT, libOp);
else if(((drcOp.trim().split("[.]"))[1]).equals("sch"))
        cell1 = Library.findCellInLibraries(drcOp2, View.SCHEMATIC, libOp);

if (cell1.isLayout())
        new MTDRCLayoutTool(dp, cell1, true, null).startJob();
else
        DRC.checkDRCHierarchically(dp, cell1, null, null,
GeometryHandler.GHMode.ALGO_SWEEP, false);
```

Figure 38: Code snippet of EVLSI modification

Tool flow configuration file defines the available tools and test in this system. Design environment administrator may edit the file located in the setup area. Figure 39 shows the tool flow configuration file content format that must be strictly observed.

```
[tool1]
test1    "command and argument"
in       path to input (library) file
out      path to output log file
test2 ……(follows the same format)
```

Figure 39: Tool flow configuration file format

Figure 40 depicts an example of tool flow configuration file. It can be seen "in" parameter defines the input library file while "out" defines the output log file. Each test requires a single entry.

```
[evlsi]
ncc      "evlsi -threads 1 -batch -lib [in] -ncc [cell]"
in       $WARD/evlsi/[lib]
out      $WARD/evlsi/validation/evlsi_ncc.log
drcL     "evlsi -threads 1 -batch -lib [in] -drc [cell].lay"
in       $WARD/evlsi/[lib]
out      $WARD/evlsi/validation/evlsi_drcL.log
drcS     "evlsi -threads 1 -batch -lib [in] -drc [cell].sch"
in       $WARD/evlsi/[lib]
out      $WARD/evlsi/validation/evlsi_drcS.log
```

Figure 40: Example of tool flow configuration file

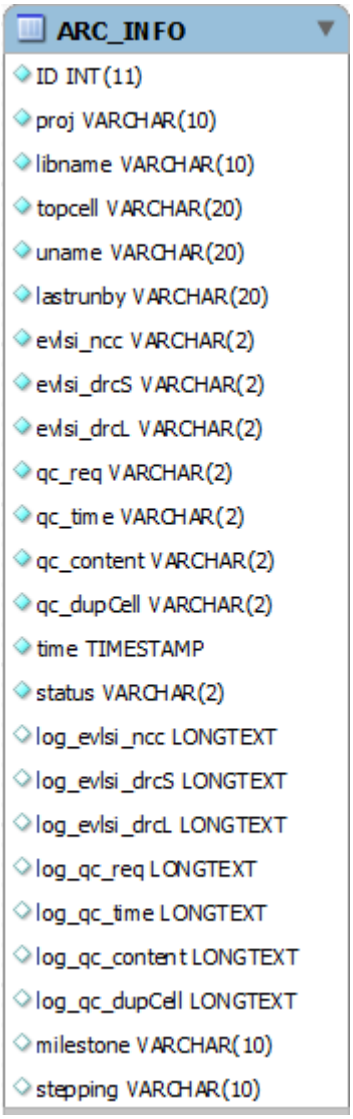


Figure 41: arc_info Entity Diagram

Whenever the quality check script is being ran, the database will be updated. The figure 41 shows the entity diagram of table arc_info which stores the library summary. Basic library information such as project, top cell name, milestone, stepping, timestamp, overall status and user information, it can be seen that there must be two columns defined for each flow; varchar field for P/F (Pass/Fail) status while the longtext field is for storing error logs.

The arc_proj.pl Perl script which is ran under the crontab monitors the arc_info table. When the status column changes to “Q” which sense for queuing, the arc_info will check the archive area for the relevant log file and rerun the test. The status column will then be changed to “R” which is short form of running. Each test will have the column changed to “-“ and updated accordingly in testing sequence.

CHAPTER 5: TESTING

5.1 Cross-site population and Environment Setup

Task	Expected	Result
Invoke setup script with valid arguments. ./setup -p bwd -cfg R0P0V3 -t toolEVLSI -b grp1 -w test1 -lib basic	Setup successful. Tools loaded. "basic" library loaded. User work directory changed to <disk>/grp1/<user>/test1 cds.lib.<pid> file created.	Pass
Setup with non-existence project name. -p noProj	Setup fail. User prompted "invalid project name".	Pass
Setup with none valid configuration. -cfg R0P0V20	Setup fail. User prompted "invalid configuration"	Pass
Setup with non-existence library. -lib noLib	Setup fail. User prompted "invalid library"	Pass
Setup with multi-hieratical library -lib afub1	Setup successful. Tools loaded. Libraries afub1, afub2 and afub3 loaded since they are dependency.	Pass
Setup with multiple libraries (all valid). -lib afub1:afub2:basic	Setup successful. Tools loaded. Libraries loaded. User work directory changed. cds.lib.<pid> file created.	Pass

Setup with library non-default library tag -lib basicLAY.BWDA0latest	Setup successful. Tools loaded. Only user and basic_bwd_lay library loaded with BWDA0latest tag.	Pass
Setup with valid library but invalid tag. -lib basicLAY.noTag	Setup pass. Library basic_bwd_lay loaded with default tag. User prompted that (tag) path not found.	Pass
Add new library Append "mfub1" into block_info table with stepping "A0".	New repository created in local central area. New jobs created for each site in job queue. After job dispatched, all sites able to see the new library.	Pass
Introduce new stepping Append "mfub1" into block_info table with stepping "A1".	New jobs created for each site in job queue. Once job dispatched, all sites and local central reflects the new tag A1work and A1latest.	Pass

Table 1: Cross-site Population and Environment Setup Test Cases

5.2 Design Library Manager

Task	Expected	Result
Invoke Design Library Manager GUI Command "dmt" on terminal.	Java GUI loaded with list of libraries in first column. "Cross" icon for library with no write access, "U" icon for user library and "L" symbol for normal library.	Pass

Left clicking on library.	Load all the tags of the particular library in second column. Current tag blue icon, other valid tag green icon and non-valid tag/path not found in "cross" icon.	Pass
Right clicking on library.	Same as previous, load tags in second column. Right click menu with restore and create cell option.	Pass
Restore previously removed view. Right click library→Restore → select view and click "OK"	View restored. User able to see the view by navigating the library.	Pass
Create new cell. Right click library → Create cell → "newCell"	Cell created. Directory created in db_root area. Default cell permission write access to all. Cell visible by navigating library.	Pass
Create new cell with duplicate name.	Cell creation fail. User prompted that cell already exist.	Pass
Left clicking tag in tag (second) column.	List of categories displayed on third column. Additionally, "Uncategorized" displayed.	Pass
Right clicking on tag.	"Change to this tag" and "tag this revision" displayed.	Pass
Change current tag. Right click on desired tag to change → "Change to this tag"	Change tag successful. Desired tag icon changes to green (indicate that it is current tag) while original tag changes to blue icon.	Pass

Custom tag current library. Right click "latest" tag → "Tag this revision" → enter desired tag name.	Tagging successful. New folder with tag name created in library directory. Tag (read-only) displayed in 2nd column.	Pass
Tag current library with duplicated name.	Tagging fail. User prompted that tag already exist.	Pass
Tag library with keyword "latest" in name.	Tagging fail. User prompted that keyword "latest" or "work" not allowed.	Pass
Tag library with keyword "work" in name.	Tagging fail. User prompted that keyword "latest" or "work" not allowed.	Pass
Left clicking category.	Display list of cells in forth column based on .cat file in library path.	Pass
Right clicking category.	Same expected outcome as previous. No right click menu option.	Pass
Clicking (left or right) "uncategorized".	Display list of cells present in library path but not belonging to any category.	Pass
Left clicking cell.	List of available views displayed in 5th column. "Eye" icon for non-locked view, green icon for view locked by current user, red icon for view locked by other user and "cross" icon for non-tracked view.	Pass

Right clicking cell.	Drop-down menu with "Cancel CO", "Checkout", "Checkin", "Rename", "Move", "Copy", "Edit Category" and "Add View".	Pass
Checkout cell. (library: basic_bwd_lay) Right click cell "nand2" → "Checkout"	All views in nand2 cell checkout successful. basic.jelib EVLSI library file created in WARD. View icon changes to green for current user, red to any other user.	Pass
Check in previously checked-out cell. Right click cell "nand2" → "Checkin"	All views in nand2 cell checkin successful. Views in basic.jelib removed. View icon changes to "eye" for current and other (all) user.	Pass
Cancel checkout on previously checked-out cell. Right click cell "nand2" → "Cancel CO"	All views in nand2 unlocked. Icon for all view changes to "eye" for current and other (all) user.	Pass
Rename cell Right click cell "nand2" → Rename → "nand2fast"	Cell nand2 renamed to nand2fast. Cell remains in current library.	Pass
Move cell (library: basic_bwd_lay) Right click cell "nand2" → Move → "afub1_bwd_lay"	Cell nand2 from library basic_bwd_lay moved to afub1_bwd_lay. Cell name remains the same.	Pass

Copy cell (library: basic_bwd_lay) Right click cell "nand2" → Copy → "afub1_bwd_lay" → "nand2fast"	Cell nand2 duplicated to library basic_bwd_lay moved to afub1_bwd_lay. Cell name remains the same.	Pass
Change cell category Right click cell "nand2" → Edit Category → "testCat"	Cell nand2 placed under "testCat" category.	Pass
Add new view Right click cell "nand2" → Add View→ "nand2.test"	New view created in db_root area. View is blank.	Pass
Add invalid view Right click cell "nand2" → Add View→ "notSameName.test"	View creation fail. User prompted that cell and view name must match.	Pass
Left clicking view.	List of version is displayed where newest is at the top in orange icon. Blue icon are for none latest version.	Pass
Right clicking view.	Same as above, with additional right click menu with "Cancel CO", "Checkout", "CO Hierarchy", "Checkin", "CI & continue", "Roll back", "remove" and "peek" options.	Pass

Checkout view Right click view "inv.lay" → Checkout	inv view checkout successful. basic.jelib EVLSI library file created in WARD. View icon changes to green for current user, red to any other user.	Pass
Checkout hierarchy (dependency) Right click view "and2.lay" → CO Hierarchy	inv, nand2 and and2 view checked out. basic.jelib EVLSI library file created in WARD. View icon changes to green for current user, red to any other user.	Pass
Cancel checkout on previously checked-out view. Right click view "nand2.lay" → Cancel CO	View in nand2.lay unlocked. Icon for view changes to "eye" for current and other (all) user. View removed from basic.jelib library file.	Pass
Check in cell Right click view "nand2.lay" → Checkin	Check in successful, db_root copy updated (revision controlled). View in basic.jelib removed. View icon changes to "eye" for current and other (all) user.	Pass
Check in cell but preserve copy in library. Right click view "nand2.lay" → CI & Continue	Same as above except that view in basic.jelib not removed.	Pass
Promote old version to be latest (current version 1.7). Right click view "nand2.lay" → Roll back → "1.2"	View nand2.lay updated to 1.8 where it is same as version 1.2.	Pass

Remove view from library. Right click view "nand2.lay" → Remove	View nand2.lay removed from library. Right clicking on library basic_bwd_lay shows "nand2.lay" available for restoration.	Pass
Open latest copy of view. Right click view "nand2.lay" → Peek	Latest version of nand2.lay copied to basic.jelib library file in WARD.	Pass
Left clicking version.	Version selected, but of not much use.	Pass
Right clicking version.	Right click menu with only "read" option.	Pass
Read old version (view: nand2.lay) Right click "1.3" → Read	Version 1.3 of nand2.lay opened in text file editor in read-only mode.	Pass
Check status. Right click on "check status" and select all library.	All checked out view displayed in the list. For view checkout but local user can be changed to cancel checkout and checkin. Ignore option only if view checked out by other user.	Pass
Performing bulk action. Click on "check status" button and randomly selecting checkin, ignore and checkout.	Bulk action performed. View with "checkin" will be committed to repository, "ignore" views will be ignored while "cancel checkout" views is unlocked.	Pass

Table 2: Design Library Manager Test Cases

5.3 Validation and Archive

Task	Expected	Result
Validate “basic” library (library file in WARD). val.pl –flow evlsi.ncc:drcS:drcL –lib basic –cell and2	EVLSI ncc, drcS and drcL performed on library basic. Terminal display summary of result. Log files generated in $WARD/evlsi/validation exported to $ARC/lib/basic/latest.	Pass
Validate library using invalid tool -flow noTool.test1	Test fail. User prompted. Partial log file generated in WARD but not Archive area.	Pass
Validate with none valid top cell -cell noCell	Test fail. User prompted. Partial log file generated in WARD but not Archive area.	Pass
Validate with library not found in WARD. -lib libNotInWard	Test fail. User prompted. Partial log file generated in WARD but not Archive area.	Pass
Validate with none valid library (not in existence) -lib noLib	Test fail. User prompted. Partial log file generated in WARD but not Archive area.	Pass
Validate “basic” library without exporting to archive area. -checkonly	Terminal display summary of result. Log files generated in $WARD/evlsi/validation but not exported to archive.	Pass
Quality checking after successful validation. qc.pl –lib basic –cell and2	Quality test pass. All 4 checks, requirement, time, duplicate cell and content pass.	Pass

Checkin new changes to library then quality checking.	Quality check fail. Content not passed as validation wasn't perform after checking in new changes.	Pass
Checkin problematic new changes to library then quality checking.	Quality check fail. Requirement test fail as re-running tools test flow gives error.	Pass
Quality checking none valid library (have not been exported to archive).	Quality test fails. User prompted of library not found.	Pass
Providing none valid top cell for quality checking.	Quality test fails due to requirement not met (validation result not same as re-test).	Pass

Table 3: Validation and Archive Test Cases

CHAPTER 6: PROJECT REVIEW

6.1 Conclusion

In a nutshell, the current VLSI design infrastructure in the educational field of Malaysia is not capable of delivering a holistic environment where developers can collaborate and work systematically. Although it is convenient to continue working in the way things have been done for the past decade, developers need a paradigm shift as the current way of working is not systematic and has no accountability in check.

This project attempted to create a superior design environment which is able to increase the productivity of developer and supervisor as well. Management of VLSI design in universities' curricular and project work is made more complete with the introduction of revision control and GUI library management. Furthermore, quality checking steps have been introduced to ensure the integrity of projects. Web-based library health status which comes in handy is the icing on the cake.

Ultimately, library sharing between universities and industry is much needed in order to decrease the time spent doing duplicated work. Thus, this project helps in achieving the goal by replicating and synchronizing library on each server without the need of human intervention. Design files are versioned control and cross-site populated at the same time.

This project showcase the fundamental concept of a unified design environment. However, there are still a number of stability and compatibility issues that have to be ironed out before it can be fully deployed into production environment. Regrettably, the learning curve for system administrator is steep. Future work involve vigorously testing the system on production server.

6.2 Limitation of System

6.2.1 Manual library creation and initialization

There is currently less automation and GUI for administrator. Creation of project and addition of libraries has to be performed by adding entries into the MySQL database system. Besides, the library's mercurial repository will need to be manually imported into the scm-manager in order for it to be hosted on the network. Due to the nature of scm-manager, this step cannot be automated as the tool only operates based on visual interaction with user.

6.2.2 Limited tool and workflow coverage

Addition of tools although to some extend can be done by tweaking the tool configuration file, however, custom support must still be manually coded. In example, the checkout and checkin function have been customized (hard coded) to support EVLSI directly. This means that design environment admin will need to do some programming in order to include other vendor tools. This also includes determining success status from the output log. Besides, introducing new tool also involve appending the database which process is not automated.

6.2.3 Single point of failure on database

Although each sites host its own physical copy of the design repository. However, due to the synchronization architecture, all sites rely on reading a single database for the job queue. This is among the major weakness of this system that can be a single point of failure. If the database system is down, all synchronization stops. Besides, the GUI and reporting system also relies on the database system.

6.2.4 Relatively slow response time of GUI

Due to the fact that checkin and checkout is not merely committing changes and locking the file respectively but pre-processing the design file so that it is compatible to the EDA tool. Pre-processing involves multiple steps such as concatenating the views into the library file in the WARD which makes it slow. Furthermore, the cell status and access control is checked. If checkout hierarchy is selected, cell dependency is also checked. When the file grows larger, response time may be in the scale of few seconds.

6.3 Future Work

With all the limitations mentioned in section 6.2, this design environment project can be further enhanced and refined. Some of the features here seems to be a "must have" but have been omitted in this project due to time limitation. These are some of the possible enhancement:

6.3.1 Installation package

Add installation script in order to ease deployment process. System administrator will not need to manually install all the dependency software and packages. The installation package will be able to source and install dependency packages from local directory or online repositories. Besides, database creation can also be automated with installation scripts.

6.3.2 Increase GUI coverage

Introducing GUI menu for setup step so user will not need to remember the project, library name and other parameter. Instead, available/valid arguments can be displayed in some sort of dropdown menu. This concept can also be applied to validation steps. GUI menu can be introduced to ease system administrator job instead of manually dealing directly with configuration files and database entry.

6.3.3 Widen Scope of Workflow

Workflow of front-end VLSI design such as behavioral and functional design style should be covered. This includes work dealing with HDL, state diagrams and test plans in both assembly and HDL. Future work should include cross project referencing especially in a highly partitioned design style.

6.3.4 Increase reliability and stability

As mentioned under limitation, the database is a single point of failure. Suggested improvement is to enable each site to have its own database while the central job queue can still remain at a single point. However, a mirror job queue table can be implemented in order to increase redundancy which is critical when failure occurs. Physical repository monitoring daemon can be introduced to maintain integrity and synchronization of design libraries.

BIBLIOGRAPHY

Advanced Micro Devices 1969, Global Provider of Innovative Graphics, Processors and Media Solutions | AMD, viewed 14 July, 2013, <http://www.amd.com/us/Pages/AMDHomePage.aspx>.

Altera Corporation 1995, FPGA CPLD and ASIC from Altera, viewed 14 July, 2013, <http://www.altera.com/>.

Anon n.d., UNIX man pages : crontab (5), viewed 5 April, 2014, <http://unixhelp.ed.ac.uk/CGI/man-cgi?crontab+5>.

Apache 2000, Apache Subversion, viewed 14 July, 2013, <http://subversion.apache.org/>.

Bancilhon, F, Kim, W and Korth, HF 1985, A model of CAD transactions, *Proceedings of the 11th international conference on Very Large Data Bases - Volume 11*, VLDB Endowment, Stockholm, Sweden, pp. 25–33.

Bradley Mitchell 1996, What Is Remote Access to Computer Networks?, viewed 15 July, 2013, <http://compnetworking.about.com/od/internetaccessbestuses/f/what-is-network-remote-access.htm>.

Cadence Design Systems 1988, EDA Software and Verification Tools - Cadence Design Systems - a Top EDA Vendor, viewed 14 July, 2013, <http://www.cadence.com/us/pages/default.aspx>.

CentOS n.d., CentOS Project, viewed 5 April, 2014, <http://www.centos.org/>.

Create Synchronicity 2012, Create Synchronicity: easy, fast, and lightweight backup and synchronization application, viewed 14 July, 2013, <http://synchronicity.sourceforge.net/>.

CVS 1990, CVS - Open Source Version Control, viewed 14 July, 2013, <http://cvs.nongnu.org/#introduction>.

Debashish Mohapatra 2002, VLSI Tutorial by Debashish, viewed 16 July, 2013, <http://debashish.info/e-learning/vlsi/mod-I/design_flow.html>.

Bibliography

Doug Sheppard n.d., Beginner's Introduction to Perl - Perl.com, viewed 2 April, 2014, <http://www.perl.com/pub/2000/10/begperl1.html>.

Dropbox, Inc 2008, Dropbox, viewed 13 July, 2013, <https://www.dropbox.com/>.

Git 2005, Git, viewed 14 July, 2013, <http://git-scm.com/>.

GoodSync 2010, GoodSync: File Synchronization Software, File Backup, File Sync, viewed 5 July, 2013, <http://www.goodsync.com/>.

Hodges, S and Rounce, P 1991, A VLSI Design Management Environment.

Intel Corporation 1968, Intel Company Overview, viewed 14 July, 2013, <http://www.intel.my/content/www/my/en/company-overview/company-overview.html>.

Linear Technology 2003, Linear Technology - Design Simulation and Device Models, viewed 14 July, 2013, <http://www.linear.com/designtools/software/#LTspice>.

Mercurial 2005, Mercurial SCM, viewed 14 July, 2013, <http://mercurial.selenic.com/>.

Microsoft 1975a, Microsoft Windows, viewed 14 July, 2013, <http://windows.microsoft.com/en-gb/windows/home>.

Microsoft 1975b, Remote Desktop Connection - Microsoft Windows, viewed 14 July, 2013, <http://windows.microsoft.com/en-us/windows7/products/features/remote-desktop-connection>.

Peer 2012, File Synchronization, viewed 13 July, 2013, <http://www.peersoftware.com/solutions/file-synchronization.html>.

Sarna, CS, Reddy, GR and Hsieh, D 1986, Managing VLSI data in a workstation configuration, *Circuits and Devices Magazine, IEEE*, 2, (4), pp. 36–40.

Sebastian, S n.d., SCM-Manager | The easiest way to share your Git, Mercurial and Subversion repositories over http., viewed 5 April, 2014, <https://www.scm-manager.org/>.

Sherhart, K, Vershel, M and Owen, J 1984, The engineering design environment, *Proceedings of the 21st Design Automation Conference*, pp. 466–472.

Static Free Software 1982, Static Free Software Home Page, viewed 14 July, 2013, <http://www.staticfreesoft.com/>.

Synkron 2011, Synkron – Folder synchronisation, viewed 13 July, 2013, <http://synkron.sourceforge.net/>.

Synopsys 1986, Synopsys.com, viewed 14 July, 2013, <http://www.synopsys.com/home.aspx>.

Ubuntu 2005, Ubuntu One : Home, viewed 13 July, 2013, <https://one.ubuntu.com/>.

ViceVersa 2001, ViceVersa File Synchronization Software: Synchronize Files and Folders, Laptop, Desktop, Server, viewed 13 July, 2013, <http://www.tgrmn.com/web/file_synchronization.htm>.

VMware n.d., VMware Player Plus: Easiest Way to Run a Virtual Machine | United States, viewed 5 April, 2014, <http://www.vmware.com/products/player>.

X.Org Foundation 2004, xorg, viewed 16 July, 2013, <http://www.x.org/wiki/>.

Yazdanipour, M, Mahmoudi, D, Yazdanipour, A, Yazdanipour, M and Mehdipour, A 2012, Comprehensive review and selection criteria for virtual network computing technology, *Wireless and Optical Communications Networks (WOCN), 2012 Ninth International Conference on*, pp. 1–5.