JOINT INSTITUTE
交大密西根学院

ECE4810J SoC Design

Fall 2022

Lab #4 Getting Started with Design Compiler, VCS, and OpenLane

Due: 11:59pm Oct. 30th, 2022

Logistics:

- This lab is a team exercise.
- Please use the discussion board on Piazza for Q&A.
- All reports and code (if available) MUST be submitted to the assignment of Canvas.
- Internet usage is allowed and encouraged.
- No late submission is allowed for this lab.

Contents

1 Overview

In this lab, you will learn about ASIC design flow. The goals of this lab are to:

- Learn the commands and how to compile the design with Design Compiler
- Learn the commands and how to compile the design with DFT Compiler.
- Learn the commands and how to compile the design with VCS.

2 Environment Setup

For those who are off campus, please connect to the SJTU VPN first every time.

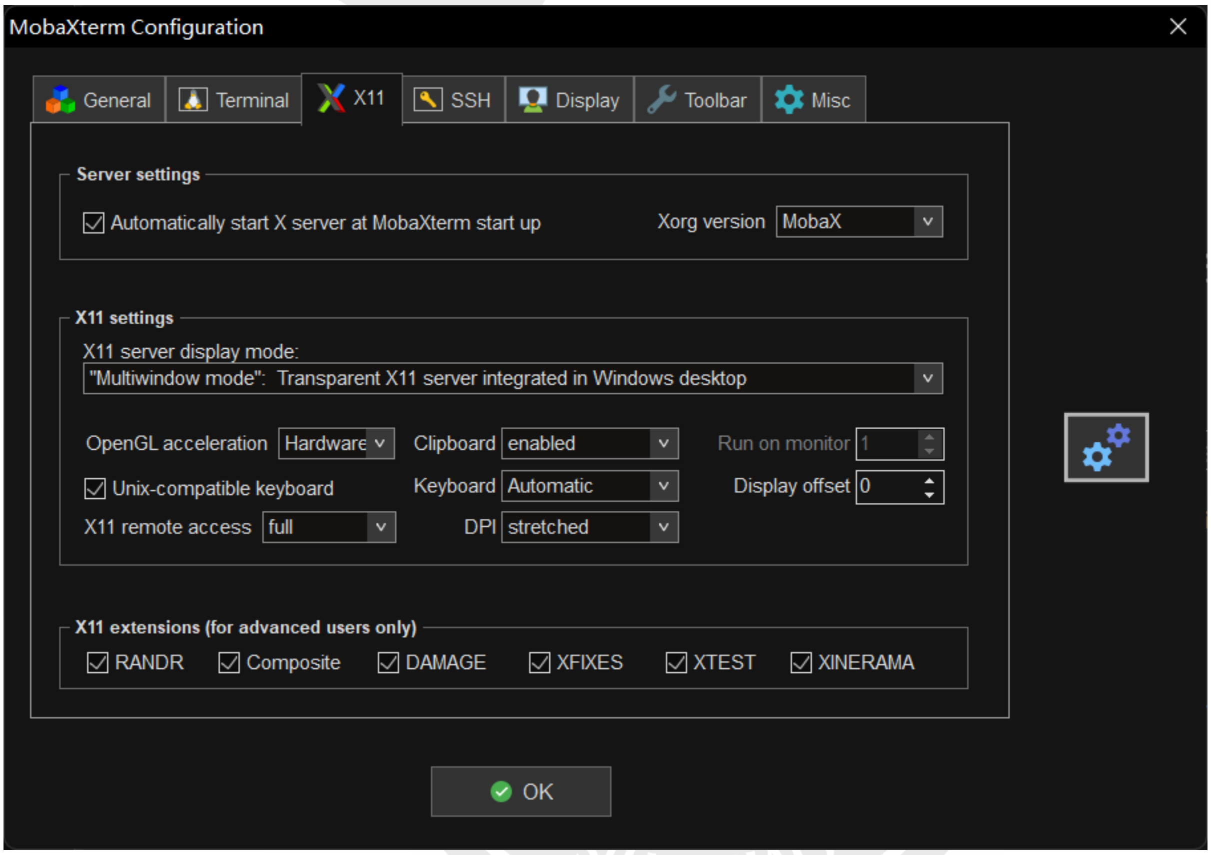
2.1 SSH connection

2.1.1 Connect by MobaXterm

Although MobaXterm has been one of the most popular and powerful enhanced terminals on Windows, it is neither free nor open-source. It is commercial software, and worse still, it does not have a community. If you want to ask for any help, you need to buy a commercial license and log in to its enclosed and exclusive customer area, i.e., you need to pay for support.

1. Download and install MobaXterm from `https://mobaxterm.mobatek.net/`.
2. Create a new session. For Basic SSH settings, specify the remote host address 202.121.180.10 and port, and specify the username as suggested. In "Advanced SSH settings", check "X11-Forwarding" and "Compression". Generally, in "Network settings", do not configure the "Proxy settings (experimental)". If you choose to use proxy settings, you should not use 127.0.0.1 as the Host because it will make X11-forwarding unavailable. The X11 process may not be responding, the connection to port 6000 or 6010 may be refused, the MobaXterm X server may shut down without notification, or the error "Can't open display" may occur. Besides, if you configure the proxy settings, you should connect to your proxy server every time you log in to the session; otherwise, you will get "network error: connection refused".
3. Double-click the session or right-click the session and click "execute" to start the session. MobaXterm allows saving passwords. The default shell is `csh`. Do not change your shell unless you are very sure about what you are doing and you have correctly migrated the complex environment settings from `csh` to your shell.
4. You can inspect the status of the X server on the top right. You can manage user sessions through the Sessions tab and browse files and directories through the SSH browser (SFTP) tab on the left panel. To transmit files quickly and conveniently, you can use the powerful free (GPL) and open-source software WinSCP. You can also set the "SSH-browser type" to SCP (enhanced speed) or SCP (normal speed) in the Advanced SSH Settings of the Session settings.

5. Go to the Settings menu->configuration, open the "X11" tab, and in the Server settings frame, or right-click the X Server icon on the top left and click "Configure", check "Automatically start X server at MobaXterm startup", so that MobaXterm X Server will automatically start at MobaXterm startup, and the icon of X Server will light. In the X11 settings frame, select X11 server display mode as "Multiwindow mode": Transparent X1 server integrated in Windows desktop. The configuration can be modified directly in C:/Users/<username>/AppData/Roaming/MobaXterm/MobaXterm.ini. The Display offset is just the display number. MobaX is selected for the Xorg version. The five options MobaX, MobaX_1.20.4, MobaXterm_1.16.3, Cygwin_1.16.3, and Cygwin_1.14.5 are mapped to XWin_MobaX.exe, XWin_MobaX_1.20.4.exe, XWin_MobaX_1.16.3, XWin_Cygwin_1.16.3.exe, and XWin_Cygwin_1.14.5.exe under C:/Users/<username>/AppData/Roaming/MobaXterm/slash/bin respectively. Just use the first one.



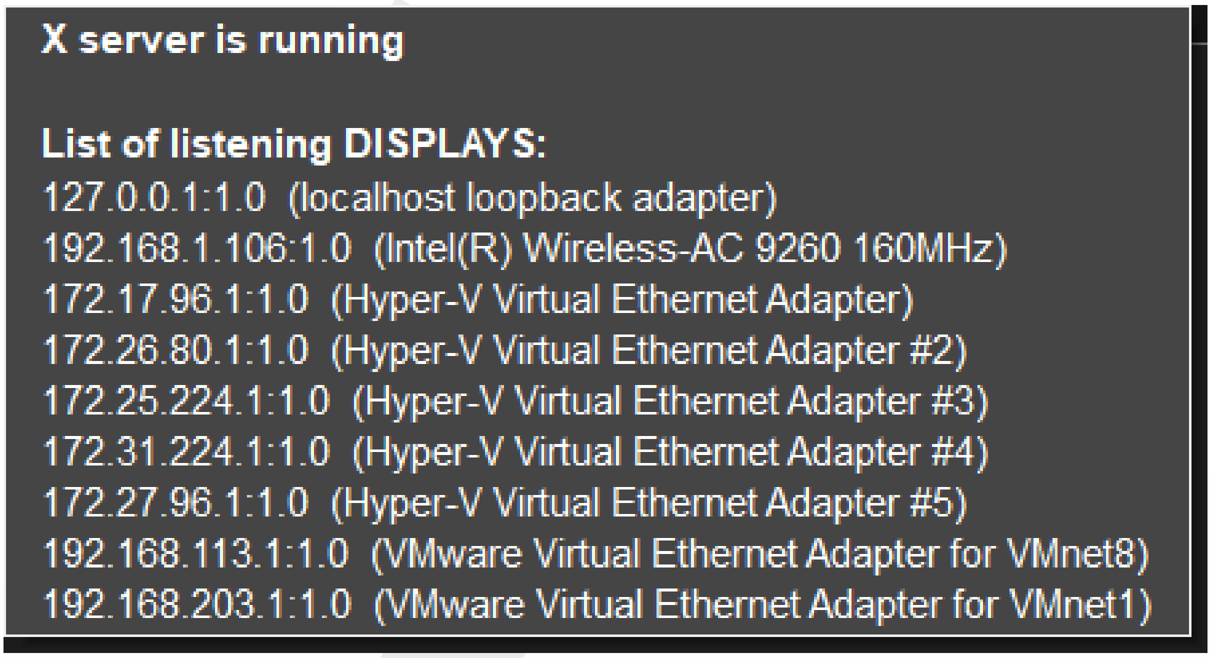
Server settings
☑ Automatically start X server at MobaXterm start up
Xorg version: MobaX

X11 settings
X11 server display mode: "Multiwindow mode": Transparent X11 server integrated in Windows desktop
OpenGL acceleration: Hardware
Clipboard: enabled
Run on monitor: 1
☑ Unix-compatible keyboard
Keyboard: Automatic
Display offset: 0
X11 remote access: full
DPI: stretched

X11 extensions (for advanced users only)
☑ RANDR ☑ Composite ☑ DAMAGE ☑ XFIXES ☑ XTEST ☑ XINERAMA

Figure 1: MobaXterm X11 Configuration.

Move the cursor on the icon of "X server", and you will see that it listens to all displays broadcasting [7]:

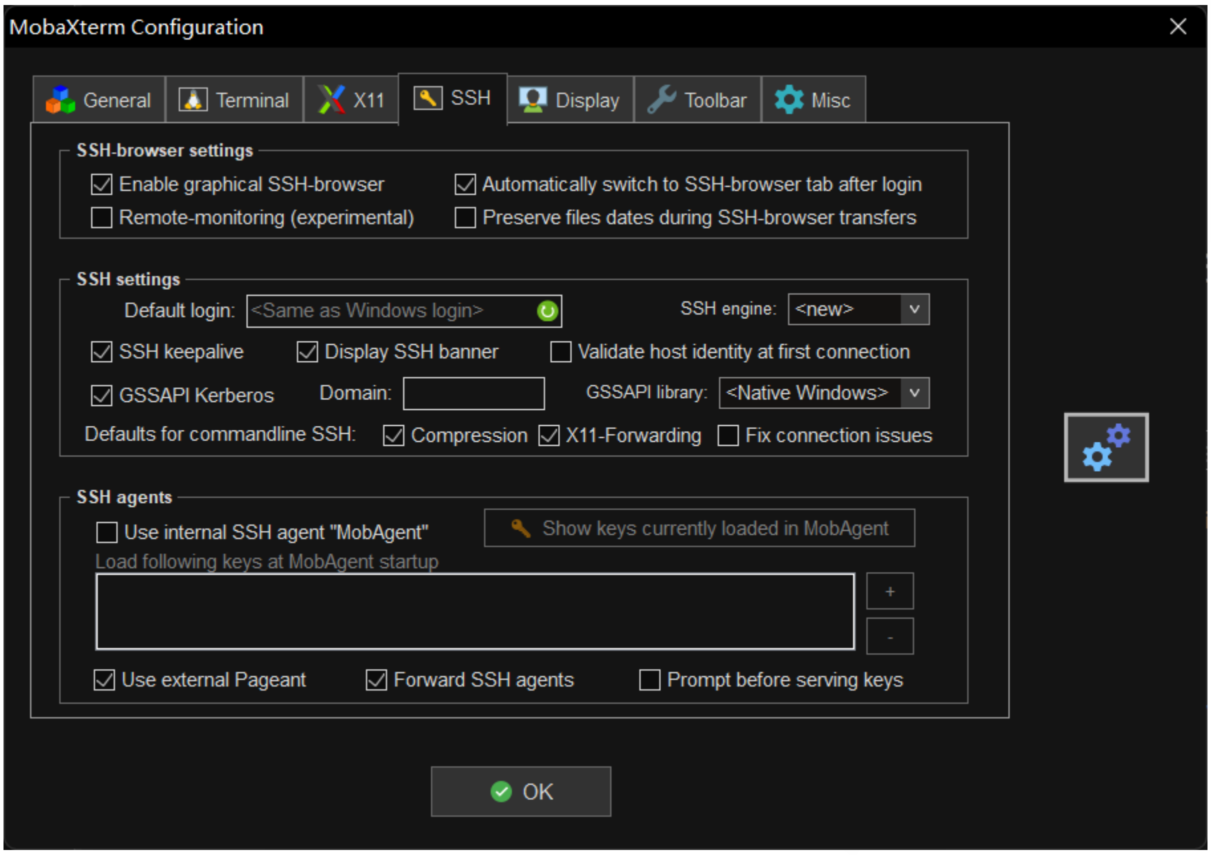


```
X server is running

List of listening DISPLAYS:
127.0.0.1:1.0  (localhost loopback adapter)
192.168.1.106:1.0  (Intel(R) Wireless-AC 9260 160MHz)
172.17.96.1:1.0  (Hyper-V Virtual Ethernet Adapter)
172.26.80.1:1.0  (Hyper-V Virtual Ethernet Adapter #2)
172.25.224.1:1.0  (Hyper-V Virtual Ethernet Adapter #3)
172.31.224.1:1.0  (Hyper-V Virtual Ethernet Adapter #4)
172.27.96.1:1.0  (Hyper-V Virtual Ethernet Adapter #5)
192.168.113.1:1.0  (VMware Virtual Ethernet Adapter for VMnet8)
192.168.203.1:1.0  (VMware Virtual Ethernet Adapter for VMnet1)
```

Figure 2: X server is running.

Also, check you have set up X11-forwarding and compression for command-line SSH:



MobaXterm Configuration

SSH-browser settings
- [✓] Enable graphical SSH-browser
- [✓] Automatically switch to SSH-browser tab after login
- [ ] Remote-monitoring (experimental)
- [ ] Preserve files dates during SSH-browser transfers

SSH settings
- Default login: <Same as Windows login>
- SSH engine: <new>
- [✓] SSH keepalive
- [✓] Display SSH banner
- [ ] Validate host identity at first connection
- [✓] GSSAPI Kerberos
- Domain:
- GSSAPI library: <Native Windows>
- Defaults for commandline SSH: [✓] Compression [✓] X11-Forwarding [ ] Fix connection issues

SSH agents
- [ ] Use internal SSH agent "MobAgent"
- Load following keys at MobAgent startup
- [✓] Use external Pageant
- [✓] Forward SSH agents
- [ ] Prompt before serving keys

Figure 3: MobaXterm SSH Configuration.

6. You can run `echo $DISPLAY` to check if your DISPLAY variable is automatically set to localhost:*.0 where * is a number. If not set, run `setenv DISPLAY localhost:*.0` to set. Note that the remote server is CentOS 6.10 Final and tcsh 6.17.00 Astron, so commands are different from Ubuntu Bash. Run graphical UI applications like `xclock` or `gedit` to test whether your X server works correctly. If you see the output

   ```
   MoTTY X11 proxy: unable to connect to forwarded X server: Network
    ↪  error: Connection refused
   Error: Can't open display: localhost:11.0
   ```

   go to Appendix: B.3. If you see the output

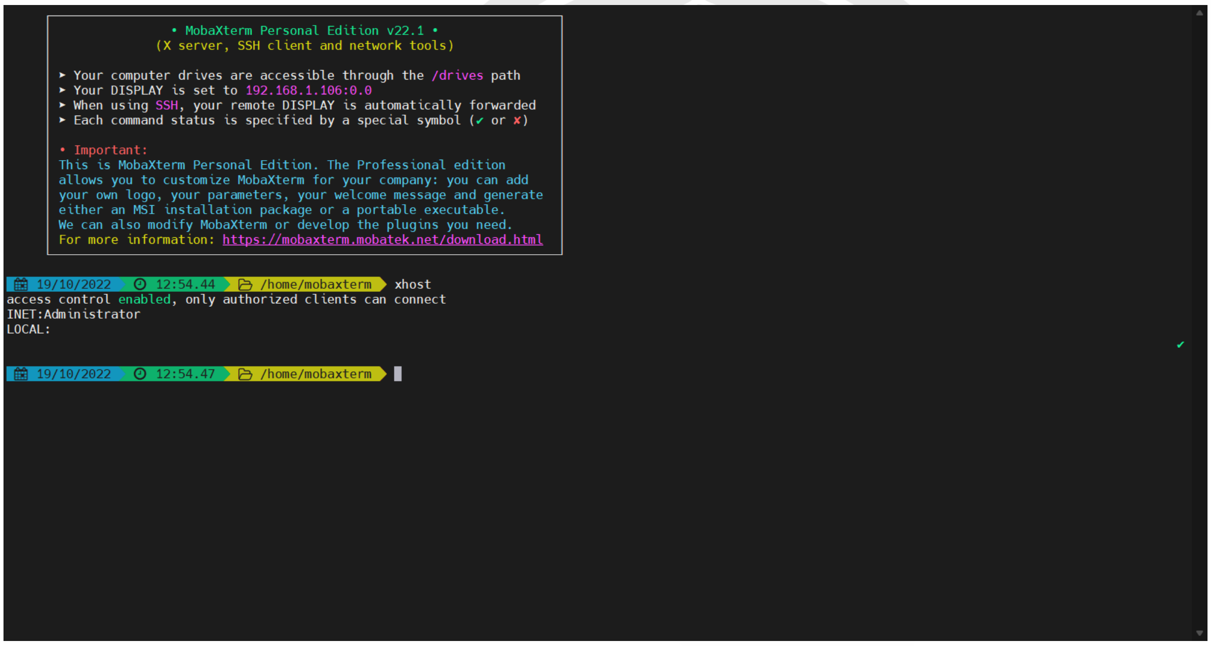
   ```
   Warning: Missing charsets in String to FontSet conversion
   ```

   Run [10]

   ```
   setenv LC_TYPE C
   ```

MobaXterm is based on Cygwin technology using the MoTTY terminal. You can open the MobaXterm terminal from the Terminal tab->Open new tab or click the "+" button on the tab control bar. The bundled Cygwin system is CYGWIN_NT-10.0-WOW i686 GNU/Linux. You can check the status of X server host by running `xhost` in the MobaXterm terminal. If it outputs like

```
access control enabled, only authorized clients can connect
INET:Administrator
LOCAL:
```



```
            • MobaXterm Personal Edition v22.1 •
           (X server, SSH client and network tools)

 ➤ Your computer drives are accessible through the /drives path
 ➤ Your DISPLAY is set to 192.168.1.106:0.0
 ➤ When using SSH, your remote DISPLAY is automatically forwarded
 ➤ Each command status is specified by a special symbol (✓ or ✗)

 • Important:
 This is MobaXterm Personal Edition. The Professional edition
 allows you to customize MobaXterm for your company: you can add
 your own logo, your parameters, your welcome message and generate
 either an MSI installation package or a portable executable.
 We can also modify MobaXterm or develop the plugins you need.
 For more information: https://mobaxterm.mobatek.net/download.html

 19/10/2022  12:54.44  /home/mobaxterm  xhost
access control enabled, only authorized clients can connect
INET:Administrator
LOCAL:
                                                                 ✓

 19/10/2022  12:54.47  /home/mobaxterm 
```

Figure 4: MobaXterm terminal and `xhost`.

Then everything is OK [8].

The log file of MabaXterm can be viewed from the Help menu->View MobaXterm log file in C:/Users/<username>/AppData/Roaming/MobaXterm/slash/tmp/MobaXterm.log. Open Windows Settings->Update & Security for Windows 10 or Privacy & Security for Windows 11->Windows Security->Open Windows Security. In Windows Security, go to Firewall & network protection, click "Allow an app through firewall", or go to Control Panel->System and Security->Windows Defender Firewall->Allowed apps, where you can allow apps to communicate through Windows Defender Firewall. If you want, you can click "Change settings" so that you can click the "Allow another app...(R)" button. Scroll down the list of allowed apps and features, find the item "xwin_mobax.exe" and it is checked on both the Private column and the Public column. You can check the log file of MobaXterm X Server from C:/Users/<username>/AppData/Roaming/MobaXterm/slash/var/log/xwin named as XWin.*.log and XWin.*.log.old.

Also, check your C:/Users/<username>/AppData/Roaming/MobaXterm/slash/etc/hosts file that it is symlinked to the Windows hosts by

```
!<symlink>/drives/C/Windows/system32/drivers/etc/hosts
```

and files like protocol, services, and networks are also symlinked to Windows' counterparts.

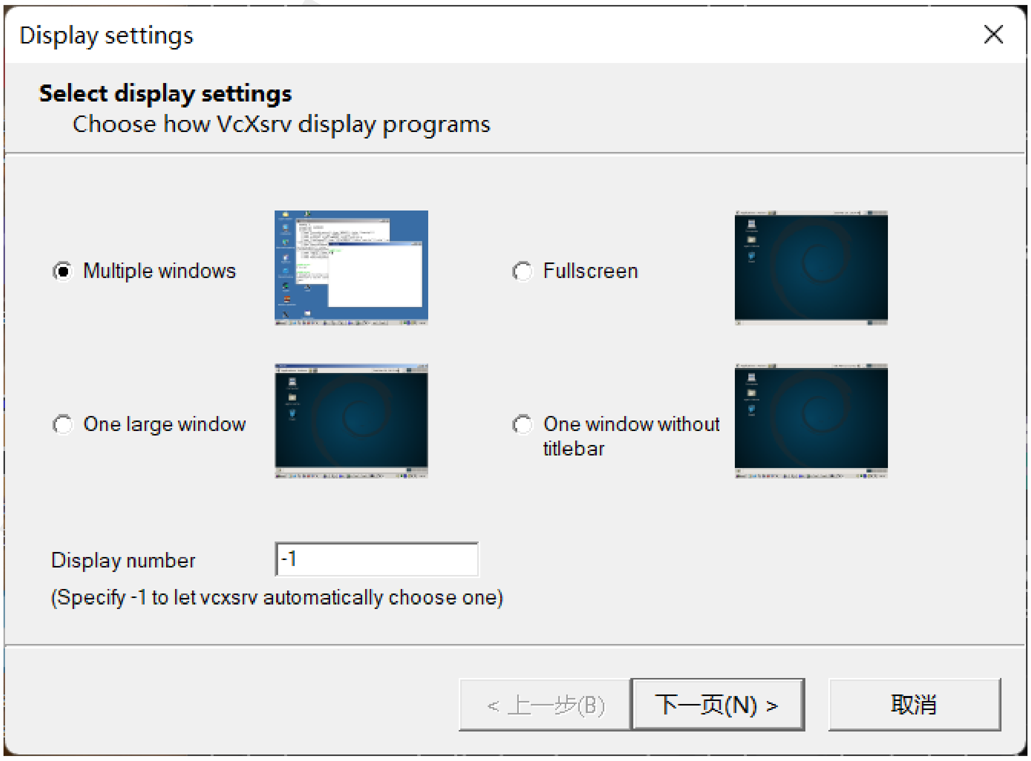
The XWin_MobaX.exe is updated from CygWin's XWin.exe, so it also accepts arguments of XWin. The documentation of XWin is [1].

2.1.2 Connect by PuTTY + VcXsrv

There are two popular options for standalone X servers on Windows: Xming and VcXsrv. However, Xming is no longer purely free and open-source: its latest free release is Xming 6.9.0.31, released on May 4, 2007, on SourceForge Xming X Server for Windows. The newer releases require £10 donations to obtain from Xming X Server. Thus, Xming is not recommended. Instead, VcXsrv is actively under development on SourceForge, freely and open source, so using VcXsrv is recommended.

For SSH client terminals, there are too many alternatives. For simplicity, you can just use PuTTY. You can also use other products like SecureCRT.

1. Download and install PuTTY.
2. Download and install VcXsrv Windows X Server.
3. After installation, you should find XLauch shortcuts; if not, go to the installation directory of VcXsrv and run `xlaunch.exe`.
4. In the Display settings, select Multiple windows and leave the Display number as -1 (0 is also OK).


Display settings

Select display settings

Choose how VcXsrv display programs

(● Multiple windows) ○ Fullscreen

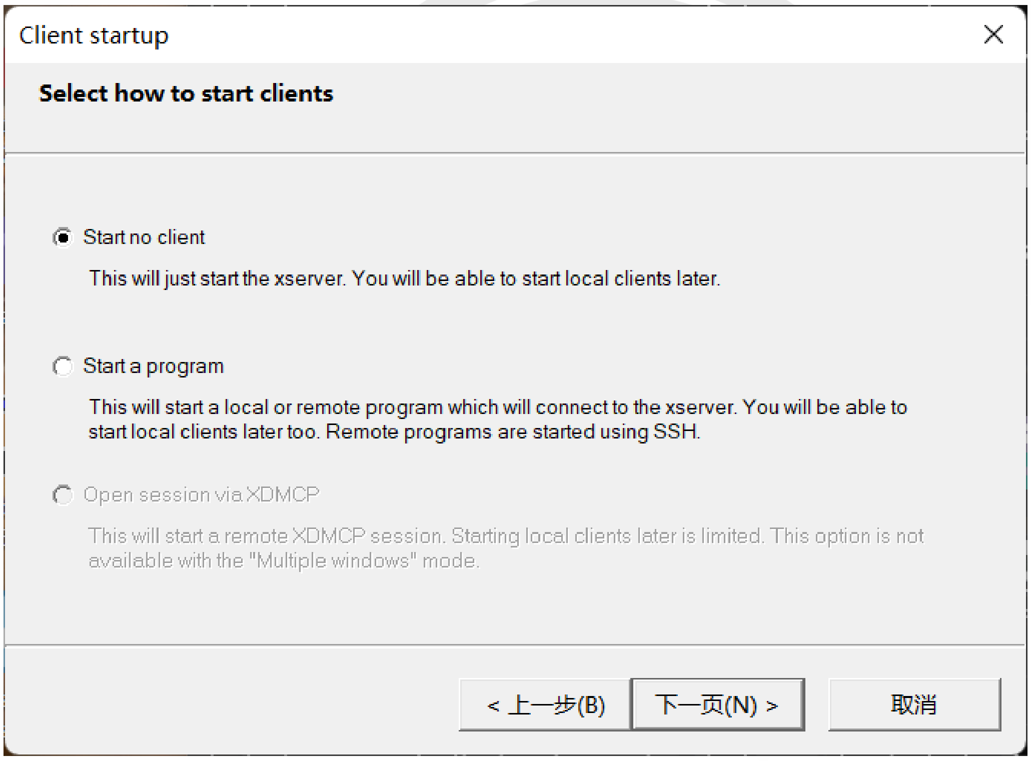
○ One large window ○ One window without titlebar

Display number: -1

(Specify -1 to let vcxsrv automatically choose one)

Figure 5: VcXsrv XLaunch Display settings.

5. In the Client startup, select "Start no client".


Client startup

Select how to start clients

● Start no client

This will just start the xserver. You will be able to start local clients later.

○ Start a program

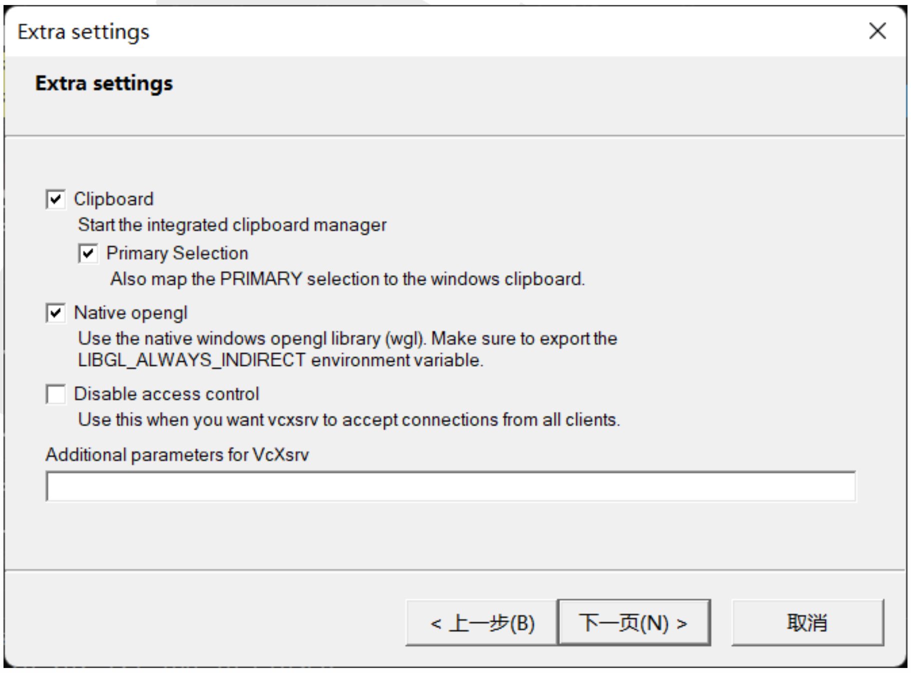
This will start a local or remote program which will connect to the xserver. You will be able to start local clients later too. Remote programs are started using SSH.

○ Open session via XDMCP

This will start a remote XDMCP session. Starting local clients later is limited. This option is not available with the "Multiple windows" mode.

Figure 6: VcXsrv XLaunch Client startup.

6. In the Extra settings, check "Clipboard", "Primary Selection", and "Native opengl". Generally, you do not need to check "Disable access control". Leave the Additional parameters for VcXsrv empty.

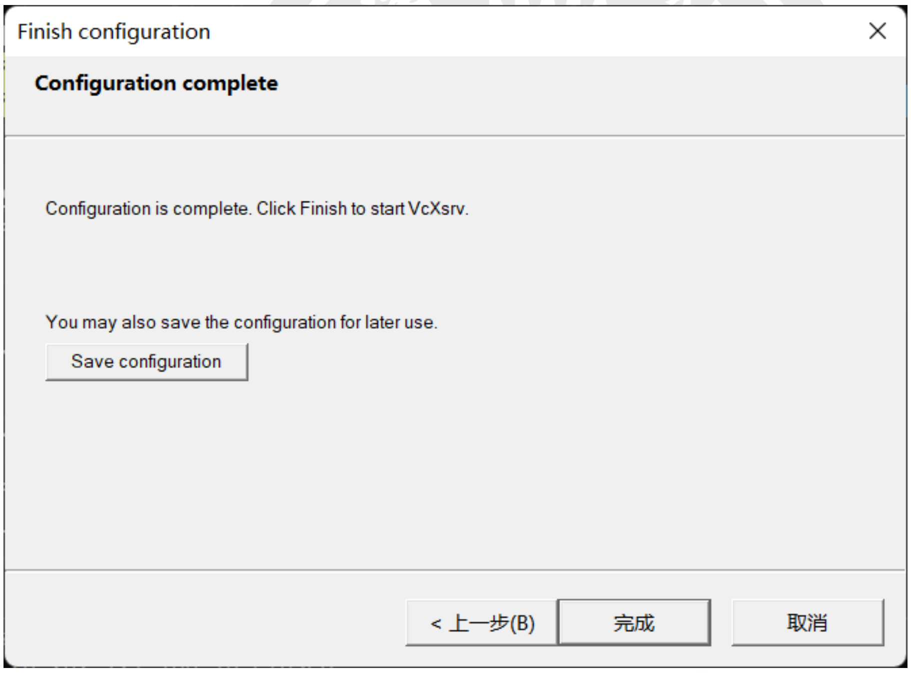

Extra settings

Extra settings

[✓] Clipboard
Start the integrated clipboard manager
[✓] Primary Selection
Also map the PRIMARY selection to the windows clipboard.
[✓] Native opengl
Use the native windows opengl library (wgl). Make sure to export the LIBGL_ALWAYS_INDIRECT environment variable.
[ ] Disable access control
Use this when you want vcxsrv to accept connections from all clients.
Additional parameters for VcXsrv

Figure 7: VcXsrv XLaunch Extra settings.

7. In the Finish configuration, you can save the configuration for later use, but it is not necessary.


Finish configuration

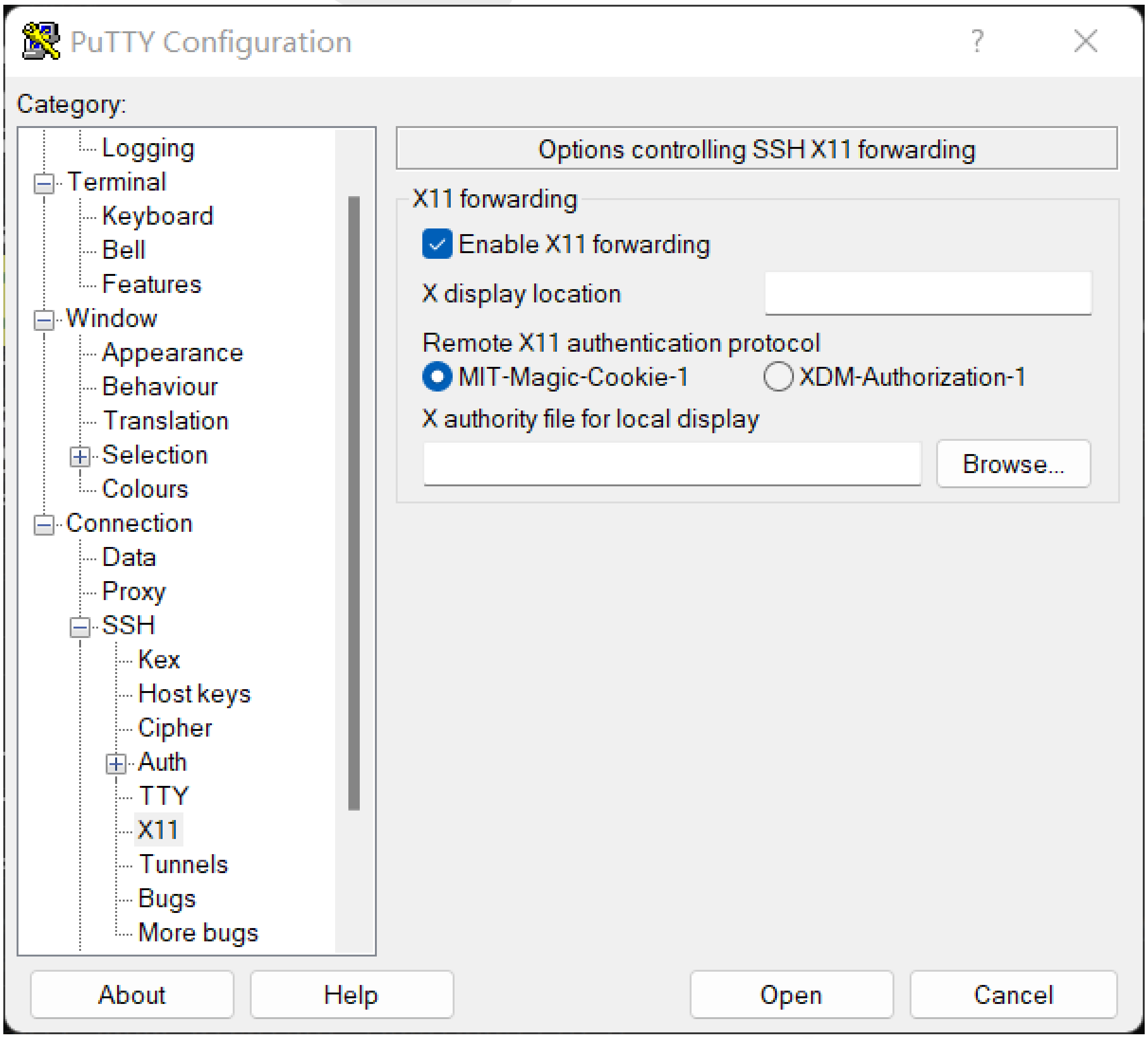
Configuration complete

Configuration is complete. Click Finish to start VcXsrv.

You may also save the configuration for later use.

Figure 8: VcXsrv XLaunch Finish configuration.

8. Make sure the icon of XLaunch keeps showing in the tray. If XLaunch exits silently, go to Appendix: B.1.
9. Open PuTTY, then the PuTTY Configuration dialog will be opened. In the Category, select "Session". In the "Specify the destination you want to connect to" frame, fill in the Host Name (or IP address) and the Port. For the Connection type, select SSH. You can save the session in the "Load, save or delete a stored session" frame, and load the session from the Saved Sessions later.
10. Go to Category->Connection->SSH->X11, in the X11 forwarding frame, check "Enable X11 forwarding". Select MIT-Magic-Cookie-1 as the Remote X11 authentication protocol. Generally, you can just leave X display location and X authority file for local display. But if needed, specify the X display location as ":0.0" or "127.0.0.1:0.0" or "localhost:0.0". Here, the first 0 is the display number, and the second 0 is the screen number. The X authority file for local display is `xauth.exe` under the installation directory of VcXsrv.


PuTTY Configuration

Category:

Options controlling SSH X11 forwarding

X11 forwarding

✓ Enable X11 forwarding

X display location

Remote X11 authentication protocol

[● MIT-Magic-Cookie-1] ○ XDM-Authorization-1

X authority file for local display

Figure 9: PuTTY X11 configuration.

11. Click "Open". In the PuTTY terminal, input your username and password as prompted. After you have done this, you can run graphical UI applications like `xclock` or `gedit` to test whether your X11 forwarding works well.

The process for Xming and MobaXerm X Server is the same as above. Open MobaXterm X Server; then establish an SSH connection with X11 forwarding through PuTTY, you can also open X11-forwarded windows.

2.1.3 Connect by Command Line + VcXsrv

Still, download and install VcXsrv and run it first.

Following [5], to install OpenSSH using PowerShell, run PowerShell as an Administrator. To make sure that OpenSSH is available, run the following cmdlet:

```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
# Install the OpenSSH Client
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
# Install the OpenSSH Server
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
# Start the sshd service
Start-Service sshd
# OPTIONAL but recommended:
Set-Service -Name sshd -StartupType 'Automatic'
# Confirm the Firewall rule is configured. It should be created
 ↪  automatically by setup. Run the following to verify
if (!(Get-NetFirewallRule -Name "OpenSSH-Server-In-TCP" -ErrorAction
    SilentlyContinue | Select-Object Name, Enabled)) {
    Write-Output "Firewall Rule 'OpenSSH-Server-In-TCP' does not exist,
     ↪  creating it..."
    New-NetFirewallRule -Name 'OpenSSH-Server-In-TCP' -DisplayName
     ↪  'OpenSSH Server (sshd)' -Enabled True -Direction Inbound
     ↪  -Protocol TCP -Action Allow -LocalPort 22
} else {
    Write-Output "Firewall rule 'OpenSSH-Server-In-TCP' has been created
     ↪  and exists."
}
```

Alternatively, to install OpenSSH by GUI,

1. Open Settings, select Apps, then select Optional Features.
2. Scan the list to see if the OpenSSH is already installed. If not, at the top of the page, select Add a feature, then:
   - Find OpenSSH Client, then select Install
   - Find OpenSSH Server, then select Install

   Once setup completes, return to Apps and Optional Features, and you should see OpenSSH listed.

Installing OpenSSH Server will create and enable a firewall rule named OpenSSH-Server-In-TCP. This allows inbound SSH traffic on port 22. If this rule is not enabled and this port is not open, connections will be refused or reset.

- Use Command Prompt (cmd.exe): Run

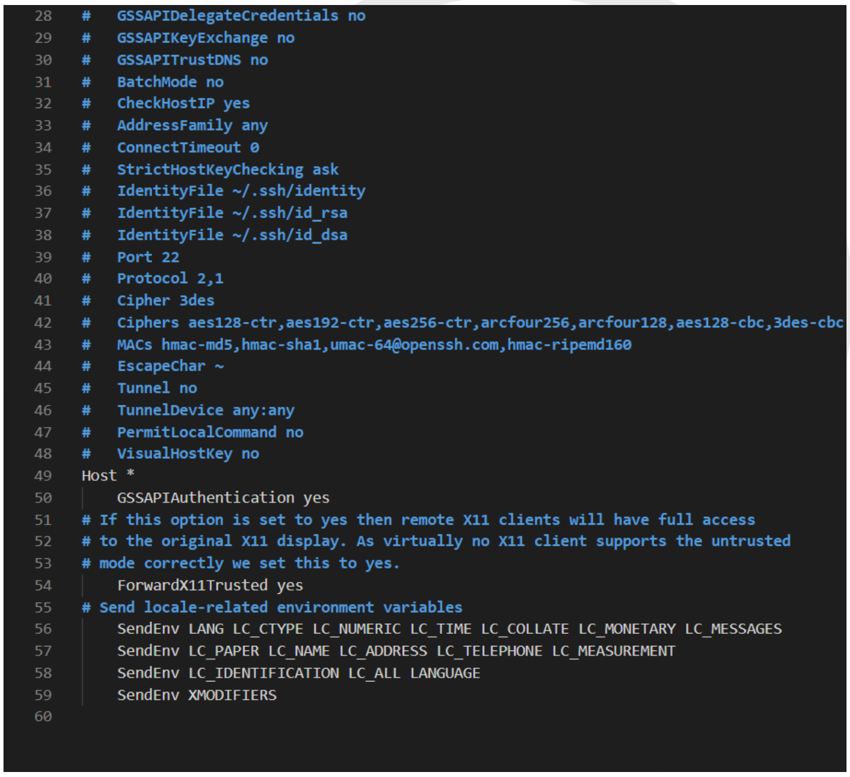
  ```
  set DISPLAY=localhost:0.0
  ssh -Y <username>@<hostname> -p <port>
  ```

- Use Windows PowerShell (powershell.exe) or PowerShell (Core) (pwsh.exe): Run [6]

  ```
  $env:DISPLAY='localhost:0.0'
  ssh -Y <username>@<hostname> -p <port>
  ```

You can run CMD or PowerShell in (Windows) Terminal.

Please use the `-Y` argument. Usually, the command `ssh` is aliased by `ssh -X`. The `-X` argument is to enable X11 forwarding, but users with the ability to bypass file permissions on the remote host (for the user's X authorization database) can access the local X11 display through the forwarded connection, and an attacker may then be able to perform activities such as keystroke monitoring. Instead, `-Y` argument enables trusted X11 forwarding. Trusted X11 forwardings are not subjected to the X11 SECURITY extension controls [11]. In `/etc/ssh/ssh_config` on your server, the `ForwardX11Trusted` option has been set to "yes", so that remote X11 clients will have full access to the original X11 display [12].

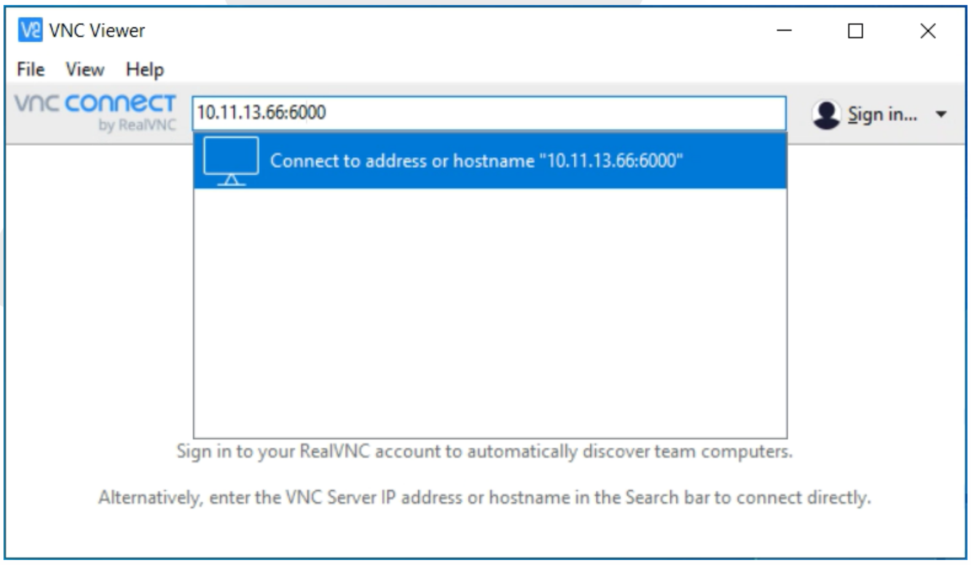


```
#   GSSAPIDelegateCredentials no
#   GSSAPIKeyExchange no
#   GSSAPITrustDNS no
#   BatchMode no
#   CheckHostIP yes
#   AddressFamily any
#   ConnectTimeout 0
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc
#   MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
#   EscapeChar ~
#   Tunnel no
#   TunnelDevice any:any
#   PermitLocalCommand no
#   VisualHostKey no
Host *
    GSSAPIAuthentication yes
# If this option is set to yes then remote X11 clients will have full access
# to the original X11 display. As virtually no X11 client supports the untrusted
# mode correctly we set this to yes.
    ForwardX11Trusted yes
# Send locale-related environment variables
    SendEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
    SendEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
    SendEnv LC_IDENTIFICATION LC_ALL LANGUAGE
    SendEnv XMODIFIERS
```

Figure 10: `ssh_config`.

2.2 VNC connection

1. Open VMware Horizon Client and login `vdi.ji.sjtu.edu.cn`.
2. Then, start the Student virtual machine through VMware Blast.
3. Open the VNC Viewer. Enter the VNC Server address 10.11.13.66:6000.

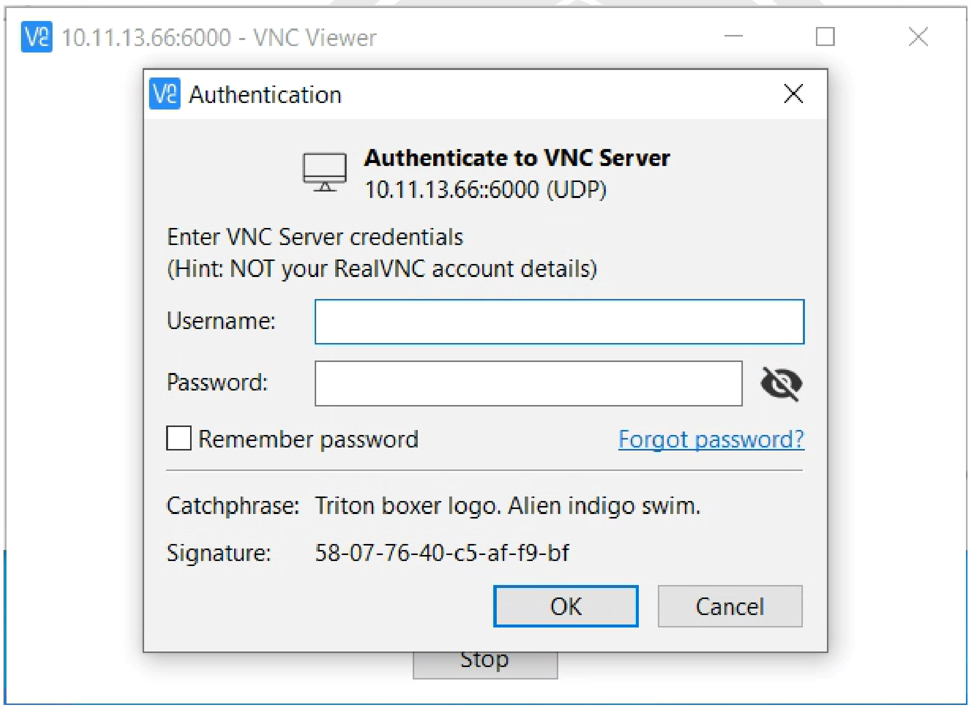


Sign in to your RealVNC account to automatically discover team computers.

Alternatively, enter the VNC Server IP address or hostname in the Search bar to connect directly.

Figure 11: VNC Viewer login.

4. In the Authentication dialog, enter your Username and Password for your SSH server and click OK.



Enter VNC Server credentials
(Hint: NOT your RealVNC account details)

Catchphrase: Triton boxer logo. Alien indigo swim.

Signature: 58-07-76-40-c5-af-f9-bf

Figure 12: VNC Viewer Authentication.

2.3 TENEX C Shell Run Commands

The server uses TENEX C Shell (tcsh), a Unix shell based on and backward compatible with the C shell (csh). It is essentially the C shell with programmable command-line completion, command-line editing, and a few other features. Unlike the other common shells, functions cannot be defined in a tcsh script, and the user must use aliases instead (as in csh). It is the native root shell for BSD-based systems such as FreeBSD. The Run Commands (RC) file `.cshrc` is not complex. It sets several variables which are not used often. It first sources `apps/env/modules.csh` which sources `/apps/systools/modules-4.7.1/init/tcsh` (here tcsh and csh files are the same) and `apps/env/cshrc.alias` which set aliases, `module load` python and htop, and set `MODULEPATH` environment. Then it sources the seven sub csh files for ANSYS, KeySight, Cadence, Synopsys, Mentor, Mathworks, and MC2 software. In the end, it appends the paths of all the available software to the system PATH variable.

The sub csh files of EDA software set license files and `module load` the tools of the software.

In this lab, we will use Synopsys Design Compiler Version R-2020.09-SP3 for linux64, which has a much better graphical user interface, so change the corresponding entry in the PATH to `/apps/synopsys/syn-2020.09/bin`.

3 Introduction to Synopsys EDKs

3.1 Overview

In this lab, you will use Synopsys 90nm Process Design Kit (PDK) or educational purpose 32/28nm Generic Library as a part of Synopsys Educational Design Kits (EDKs) under /home/pdk/synopsys_edk/edk90/synopsys_design_flow_using_90nm_library and /home/pdk/synopsys_edk/edk32. The directory structure of Synopsys 90nm PDK is

- dc - directory for logic synthesis
- fm - directory for formal verification
- hercules - directory for post-layout DRC, LVS checks
- icc - directory for physical synthesis
- models - directory for technology files for standard cells in Liberty syntax
- ptpx - directory for power analysis
- ref - all needed information for physical synthesis
- src - RTL directory
- sta - directory for static timing analysis
- starrc - directory for post-layout parasitic extraction
- tmax - directory for ATPG generation

- vcs -directory for functional verification
- verilog - Verilog codes of library cells

A process design kit (PDK) is a set of files used within the semiconductor industry to model a fabrication process for the design tools used to design an integrated circuit. The PDK is created by the foundry defining a certain technology variation for their processes. It is then passed to their customers to use in the design process. The customers may enhance the PDK, tailoring it to their specific design styles and markets. The designers use the PDK to design, simulate, draw and verify the design before handing the design back to the foundry to produce chips. The data in the PDK is specific to the foundry's process variation and is chosen early in the design process, influenced by the market requirements for the chip. An accurate PDK will increase the chances of first-pass successful silicon.

3.2 Theoretical parts

3.2.1 Design Compiler

1. Design Compiler optimizes designs to provide the smallest and fastest logical representation of a given function. It comprises tools that synthesize HDL designs into optimized technology-dependent, gate-level designs. It supports a wide range of flat and hierarchical design styles and can optimize both combinational and sequential designs for speed, area, and power.

2. DC uses information from the setup file for normal operation. In this file, DC searches technology libraries in search_path. In the setup file target_library, link_library, and synthetic library are technology files in liberty syntax. DC used the mentioned files while transferring the RTL design to the gate-level design. The link and target libraries are technology libraries that define the semiconductor vendor's set of cells and related information, such as cell names, cell pin names, delay arcs, pin loading, design rules, and operating conditions. Here variable lib_path can be found, which shows the path to milkyway reference library for the physical synthesis tool. By means of using `set mw_reference_library` command, reference milkyway library can be given to the physical synthesis tool.

3. The first stage of the synthesis process is to analyze and elaborate RTL. In elab.tcl, command `define_design_lib` is used to show the path to the design library, which contains the analyzed form of RTL. In this script, the source command is used to execute the setup script. After sourcing the setup file, the design can be elaborated. For this command, elaborate is used. Then, the design is uniquified. Afterward, the write command is used to save the elaborated design.

4. In order to perform the next steps, compile.tcl is used. Here first setup file is sourced again. Then, design is read using `read_ddc` command. After reading the design, the current design must be set for which `current_design` command is used. Then it is necessary to link the design to its references. Link command is used. Design constraints are given in another file which is read using the source command. Here gate level netlist with GTECH gates becomes a gate-level netlist with gates from

a given technology library by compile command. To see the results, different types of write and report commands are used.

3.2.2 Formality

1. Formality is an application that uses formal techniques to prove or disprove the functional equivalence of two designs or two cell libraries. For example, users can use Formality to compare a gate-level netlist to its register transfer level (RTL) source or to a modified version of that gate-level netlist. After the comparison, Formality reports whether the two designs or cell libraries are functionally equivalent.
2. Formality creates FM_WORK directory upon invocation. It contains containers and shared technology libraries.
3. A container is a complete, self-contained space into which Formality reads designs. It is typical for one container to hold the reference design while another holds the implementation design. In general, a user does not need to concern himself with containers. Design is loaded either as a reference or implementation.
    - `read_vhdl -c ref Ref_file_path_name` – the mentioned file is read in container as a reference
    - `read_verilog -c impl Impl_file_path_name` – the mentioned file is read in container as implementation
    - `set_top ref: /*/Top_design` - sets the top-level design for the reference
    - `set_top impl: /*/Top_design` - sets the top-level design for the implementation
    - `read_db -c impl lib_db_file -technology_library` – loads in container technology library for implementation
    - `report_containers` - Produces a list of containers
    - `current_container` – sets or gets a current container
    - `cputime` - Returns the CPU time used by the Formality shell
    - `verify ref: /WORK/top_design impl:/WORK/top_design` – verifies reference and implementation design equivalency
    - `report_status` – reports verification results

3.2.3 IC Compiler

1. IC Compiler is a single, convergent netlist-to-GDSII or netlist-to-clock-tree-synthesis design tool for chip designers developing very deep submicron designs. Gate-level netlist, a detailed floorplan, timing constraints, physical and timing libraries, and foundry-process data are given to its input, and it generates either a GDSII-format file of the layout or a Design Exchange Format (DEF) file of placed netlist data ready for a third-party router as an output. IC Compiler can also output the design

as a binary Synopsys Milkyway database at any time for use with other Synopsys tools based on Milkyway or as ASCII files (Verilog, DEF, and timing constraints) for use with not Synopsys tools.

2. ICC uses information from the setup file for normal operation. In this file, search_path is mentioned where ICC searches technology libraries. In the setup file target_library, link_library and synthetic library are technology files in liberty syntax. The link and target libraries are technology libraries that define the semiconductor vendor's set of cells and related information, such as cell names, cell pin names, delay arcs, pinloading, design rules, and operating conditions. Here variable `lib_path` can be found, which shows the path to milkyway reference library. By means of using `set mw_reference_library` command, reference milkyway library is given to ICC.

3. Other design steps are done using synthesis.tcl script.

   (a) Here first setup file is sourced for which source command is used.

   (b) Then in order to start physical synthesis, it is necessary to create milkyway library with `create_mw_lib` command. These command options with their descriptions are given below:
   library name – name of created library
   -tech techfile_name.tf – give technology file
   -mw_reference_library "$ref_lib_name" – give milkyway reference library

   (c) After creating the library, it is necessary to open the library. In order to do this, `open_mw_lib` command is used.

   (d) Then, the design obtained by DC must be read. To read the design, `read_ddc` command is used.

   (e) After reading the design, the floorplan must be done. In order to do this, `initialize_floorplan` command with all necessary options is used.

   (f) Then power and ground connection must be done with `derive_pg_connection` command.

   (g) The next step in the physical synthesis process is to create rectangular rings for power and ground supplies. In order to do this, `create_rectangular_rings` command with all options is used.

   (h) To create power and ground straps `create_power_straps` command is used.

   (i) Afterwards FP placement (create_fp_placement) is done.

   (j) Then prerouting of standard cells follows (preroute_standard_cells).

   (k) Afterwards, placement is done (place_opt).

   (l) Final steps in the physical synthesis process are CTS (clock_opt) and routing (route_opt).

3.2.4 Hercules

1. Hercules can verify layout Design Rule Checks (DRC), perform Electrical Rule Checks (ERC), extract layout structures, and compare them to an original design netlist by using the Layout versus Schematic (LVS) application and generate or modify data for mask preparation. The hierarchical checking of algorithms makes Hercules particularly well-suited for large and complex IC verification.

2. While working with Hercules user uses Runset files. Runset files are ASCII (text) files containing instructions for determining where Hercules gets its files and how it runs. The Runset file contains the following sections.

   - Runset Header information
   - Runset options
   - Preprocessing options
   - Layer assignments
   - SNAP command
   - GRID checks
   - DRC checks

3. DRC Runset Header information section is defined by the keyword HEADER, found in the left margin of the file. The HEADER section contains variables that define where Hercules can find the layout libraries and other input files. The HEADER section also contains information about where to write intermediate processing files and output files.

   - The INLIB variable tells Hercules the name of the input library that the user wants to check. If the user uses GDSII formatted data, the INLIB line needs to specify a complete path and GDSII file name.
   - The BLOCK tells Hercules the name of the top-level input structure which the user wishes to verify.
   - The OUTLIB setting tells Hercules the name of the output library, which contains all the permanent output layers (created by Hercules), error output, and newly-created graphical layers.
   - The OUTPUT_BLOCK setting tells Hercules what to call the top-level output structure that holds all of your error hierarchy and permanent output layer placements.
   - The GROUP_DIR setting tells Hercules where to place all the group files it temporarily creates. Group files contain the data on which Hercules works during runset execution.

4. DRC Runset OPTIONS section allows specifying global assignments for various Hercules processes. Hercules also has a DRC_OPTIONS section. All variables in the DRC_OPTIONS section may also be placed in the OPTIONS section.

5. The PREPROCESS_OPTIONS section allows users to specify the output of information and the setting of path grid checking options. Users can set options for increased information in the block. LAYOUT_ERRORS file and the tree files. Users can also set options for path grid checking, which is done when the layers are read during the ASSIGN section. For example, setting the CHECK_PATH_90 option to FALSE allows users to have 45-degree path data in design.
6. The ASSIGN section assigns names to the database layers found in the design.
7. The SNAP command section forces data to conform to a grid resolution during a run.
8. The GRID_CHECK command performs grid checking after group file creation and appears in the runset after the ASSIGN section.
9. DRC Checks - here user writes commands to check DRC violations. For the results, see Block_name.LAYOUT_ERRORS
10. LAYOUT_PATH, INLIB, BLOCK, OUTLIB, FORMAT, and GROUP_DIR variables in LVS runset files HEADER are the same as in DRC runset file. Four variables in HEADER, which are not the same as in DRC runset HEADER, are given below.
    - COMPARE_DIR - This setting tells Hercules where to place all the COMPARE output files it temporarily or permanently creates. The default value for the COMPARE_DIR variable is run_details/compare/.
    - EQUIVALENCE - This variable specifies the location and name of the equivalence file. The equivalence file is a comparison input file that lists the structures to be compared. It is generally referred to as the EQUIV file.
    - SCHEMATIC - This variable is set to the file name, including the path, of the schematic netlist file. This variable is required to execute a Hercules LVS run. If a user does not specify this file, or if Hercules cannot find the file user specifies, the Hercules job terminates with an error indicating that the schematic file is not specified or that Hercules cannot open the schematic netlist for reading.
    - SCHEMATIC_FORMAT - This variable is set to the format type of the input schematic netlist. Hercules supports CDL, Hercules, SPICE, EDIF, EDIF3, VERILOG, and SILOS. This variable is required if the specified input schematic netlist is not in the Hercules format. If this format is set to something other than Hercules, Hercules calls NetTran, the netlist translation utility.

    For results, see Block_name.LVS_ERRORS.

3.2.5 PrimeTime

1. PrimeTime is a full-chip, gate-level static timing analysis tool that is an essential part of the design and analysis flow for today's large chip designs. PrimeTime

exhaustively validates the timing performance of a design by checking all possible paths for timing violations without using logic simulation or test vectors.
PrimeTime fits ideally into the Synopsys physical synthesis flow because it uses many of the same libraries, databases, and commands as other Synopsys tools such as Design Compiler. It can also operate as a stand-alone static timing analyzer in other design flows. It accepts design information in a wide range of industry-standard formats, including gate-level netlists in .db, Verilog, and VHDL formats; delay information in Standard Delay Format (SDF); parasitic data in Reduced Standard Parasitic Format (RSPF), Detailed Standard Parasitic Format (DSPF), Standard Parasitic Exchange Format (SPEF), and Synopsys Binary Parasitic Format (SBPF) formats and timing constraints in Synopsys Design Constraints (SDC) format.
If PrimeTime finds any timing violations, the design needs to be resynthesized using new timing constraints (generated by PrimeTime) to fix the conditions that are causing the timing errors.
The GUI offers some visual analysis capabilities that are not available in pt_shell. For example, users can view schematics of the design, display clock waveforms, and generate histograms of analysis results, such as path slack, net capacitance, and bottleneck cost. The console window within the top-level window lets users enter commands and view the text response, just like pt_shell.

2. The first step is to read the gate-level design description and associated technology library information. PrimeTime accepts design descriptions and library information in .db format and .ddc and gate-level netlists in Verilog and VHDL formats. The corresponding commands for reading design files are read_db, read_ddc, and read_verilog, read_vhdl.
   After reading a set of hierarchical design files, the link_design command resolves all references between different modules in the hierarchy and builds an internal representation of the design for timing analysis.
   To back-annotate the design with parasitic capacitance and resistance information, the read_parasitics command is used. PrimeTime accepts detailed parasitic data in RSPF, DSPF, SPEF, and SBPF formats.

3. These are some of the more common report commands:

   - `report_port` - Lists the ports and shows port information such as the direction, pin capacitance, wire capacitance, input delay, output delay, related clock, design rules, and wire load information.

   - `report_clock` - Generates a report on the clocks defined for the design, showing for each clock the name, period, rise and fall times, and timing characteristics such as latency and uncertainty.

   - The `report_timing` command is perhaps the most flexible and powerful PrimeTime analysis command. It provides general or more information about the timing of the whole design, a group of paths, or an individual path. The command options let users specify the types of paths reported, the scope of the design to search for the specified paths, and the type of information included in the path reports.

- The `report_constraint` command reports the results of constraint checking done by PrimeTime. Users can obtain information such as the location of the worst violation, the amount by which the constraint is met or violated, and the calculation of the delays used for checking the constraint.

3.2.6 PrimeTimePX

1. PrimeTime PX is an add-on feature to PrimeTime that accurately analyzes the power dissipation of cell-based designs. It is intended as an advanced solution for ASIC and structured custom circuit designers who are developing products for power-critical applications such as portable computing and telecommunications. PrimeTime PX builds a detailed power profile of the design based on the circuit connectivity, the switching activity, the net capacitance, and the cell-level power behavior data in the Synopsys database format (.db) library, which can be either a nonlinear power model (NLPM) or a Composite Current Source (CCS) library. It then calculates the power behavior for a circuit at the cell level and reports the power consumption at the chip, block, and cell levels. PrimeTime PX supports a gate-level netlist only.

2. Set the power_enable_analysis variable to true to enable power analysis and see power data. The default value of this variable is false. If a user does not set this variable to true, he cannot see power data. A PrimeTime PX license is required to use this variable.
   Power analysis mode can be selected by using the power_analysis_mode variable. The syntax of this variable is as follows:
   `set power_analysis_mode averaged | time_based`
   This variable must be set before using any of the power commands. If this variable is not set, PrimeTime PX, by default, performs averaged power analysis.
   The tool supports netlists in Verilog, VHDL, EDIF, .db, .ddc, and Milkyway formats. The technology library must be in the .db (Synopsys database) format. Both NLPM and CCS libraries are supported. `Read_verilog` command can be used to read the design.
   In order to read the VCD file that contains the switching activity information read_vcd command can be used. The `-rtl` option of the `read_vcd` command is used to specify that the VCD is generated from an RTL simulation. If the VCD file is generated from a zero-delay simulation, the `-zero_delay` option is used. When neither option is specified, the tool assumes that the file is a gate-level VCD file.
   Power analysis is triggered consistently for the various power analysis modes during the execution of the update_power command. After the analysis, the `update_power` command generates the power data.
   In order to generate an averaged or time-based power report that contains the power consumption for the design `report_power` command can be used.

3.2.7 TetraMAX

1. TetraMAX is a high-speed, high-capacity automatic test pattern generation (ATPG) tool. It can generate test patterns that maximize test coverage while using a minimum number of test vectors for a wide variety of design types and design flows. It is well suited for designs of all sizes up to millions of gates.

2. `read_netlist` command is used to read the design. Option delete is used to clear in-memory designs and switch to new designs. `read_netlist` command with option `-library` is used to read the library models. Building the ATPG design model takes the parts of the design which are to be part of the ATPG process, removes the hierarchy, and puts them into an in-memory image that TetraMAX can use. Design building is done by means of using `run_build_model` command. If design references undefined modules, TetraMAX sends error messages during execution of `run_build_model`. To identify all currently referenced modules, users can use the Netlist > Report Modules menu command or enter the `report_modules -undefined` command. In order to perform DRC from the command line, `run_drc` command can be used. Users can write out a template STIL file at any point after executing a `run_build_modell` command. To do so, the `write_drc_file` command or the Write tab in the DRC dialog box can be used. User can initialize the fault list for all faults using the `add_faults -all` command. Some violation IDs show an abort indicator suffix, which appears as Z7-12.A or Z6-3 (Abort). This means that the ATPG analysis of the violation was aborted. In such cases, users might want to increase the ATP abort limit. Users can specify the pattern generation effort and other ATPG settings using the Run ATPG dialog box or can use the `set_atpg` command at the command line. To run ATPG from the command line, the `run_atpg` command is entered. User can generate a fault summary report using the `report_summaries` or `report_fault` command. In order to write test patterns from the command line `write_patterns` command can be used.

3.2.8 VCS

1. VCS is a high-performance, high-capacity Verilog simulator that incorporates advanced, high-level abstraction verification technologies into a single open native platform. VCS enables analyzing, compiling, and simulating Verilog design descriptions. It also provides users with a set of simulation and debugging features to validate the design. These features provide capabilities for source-level debugging and simulation result viewing. VCS supports all levels of design descriptions but is optimized for the behavioral and register transfer levels.

2. To create an executable named simv by default, the following VCS command line is used:
   vcs source_file options. Here source_file is .v file of a design that needs verification and testbench files. Some options are described below:

   - -f filename– Specifies a file name that contains a list of absolute path names for Verilog source files and compile-time options.

   - -R–Runs the executable file immediately after VCS links it together. Any runtime option can be added to the VCS command line.

- -RI –Compiles model for interactive use, invokes the VirSim graphical user interface immediately after compilation, and pauses simulation at time zero.
- +v2k–Enables language features in the IEEE 1364-2001 standard.
- -v filename–Specifies a Verilog library file, in which VCS looks for the modules and UDP instances that are instantiated, but not defined, in the source code.
- +tetramax–Enables simulation of TetraMAX's testbench in zero delay mode.

3.2.9 StarRC

StarRC is a software tool that extracts parasitics from connected databases that represent IC layout designs.

StarRC can be used to generate netlists to conduct timing, clock, noise, or power analysis.

Users can also extract parasitics like resistors, capacitors, and inductors from a layout design database.

(*.nxtgrd) is a database containing capacitance, resistance, and layer information, which can be encrypted. StarRC uses this output database (.nxtgrd file) to calculate the parasitics of the actual layout by pattern matching.

The mapping file, which maps the database layer to the tcad_grd layer, is needed for every StarRC run. The mapping file can be written manually or by using the StarRC GUI. Mapping multiple database layers to a single process layer is valid, but the reverse is prohibited. Each logically connected database layer must be mapped in this file, even if the layer is derived or used only for intermediate connections with no real physical significance. In LEF/DEF flows, this means that each layer defined in the technology LEF file must be mapped (including VIAs).

For extraction, StarRC also uses .tf.

After extraction user gets extraction netlist in .spf format.

3.3 Design process

The overview of the design process is

1. Logical synthesis in dc.
   (a) Edit the technology-specific file to use the library
   Also edit the clock frequency targets dc/scripts/setup.tcl.
   (b) Invoke DC.
   (c) Elaborate the design
   For this use ./scripts/elab.tcl.
   (d) Synthesis of the design
   For this use ./scripts/compile.tcl

Reports: ./reports/*
DDC netlist: ./db/TOPDESIGN.ddc

Analyze the results (reports to `backend` and `db` directories)

2. Formal verification in fm.
   fm_comp.scr.gate: This script is used to make a formal comparison between the RTL and gate-level netlist.

   Procedure:
   run_fm.gate script is provided to set up environment variables for the fm_comp.scr.gate script.

   Usage:
   run_fm.gate <src path> <library file> <netlist file> <dwroot path>

   <src path> path to src directory. Under this directory, the RTL and the memory_models directories must be present.

   <library file> this specifies the absolute or relative path to the db file of the library (do not use pg pin syntax library)

   <netlist file> this specifies the absolute or relative path to the .db,.ddc, or .v file of the gate-level netlist

   <dwroot path> path to installation of Synopsys DesignWare

   Analyze the results (logs and reports directories)

3. Physical analysis in icc.
   (a) Invoke ICC.
   (b) Run script ./scripts/physical.tcl.
   (c) Write GDS and .v from ICC to directory ../hercules (use information from write_verilog_write_gds.txt).

   Analyze the results (spef and Milkyway library PARSER_ICC directory).

4. Post-layout DRC, LVS checks in Hercules.
   (a) Transfer Verilog from ICC to sp (use ./verilogtosp).
   (b) Copy GDS and sp to ./DRC for DRC and ./LVS for LVS.
   (c) Move to directory ./DRC for drc.
      i. Invoke Hercules.
      ii. Run script rules.drc.9m_saed (use Hercules rules.drc.9m_saed).

      Analyze the results (PARSER.LAYOUT_ERRORS file).
   (d) Move to directory ./LVS for lvs.
      i. Invoke Hercules.

ii. Run script rules.drc.9m_saed (use Hercules rules.drc.9m_saed).

Analyze the results (PARSER.LAYOUT_ERRORS and PARSER.LVS_ERRORS files).

5. Static timing analysis in sta. The scripts provided allow users to run static timing at many different stages. run_sta shell script is provided to enable the different options for running static timing.
run_sta.csh Switch Usage: -netlist [path to netlist] : Path+filename of netlist to time -speed [WC|BC|BC_WC] : operating condition, (default best_case/worst_case analysis if done) -spef [path] : Path+filename of spef file -dspf [path] : Path+filename of dspf file -load [path] : Path+filename of load file -sdf [path] : Path+filename of sdf file -clk_tree : post clock tree timing, otherwise ideal clocks are used -duty [45|50|55] : Clock duty cycle 45/55 or 50/50 (default) or 55/45 -mode [sync|syncbypass|capture|shift] : mode of operation (default = sync) -write_sdf : sdf file will be written out -no_reports : indicates interactive mode without generation all the timing reports -report_dir [sub-dir name] : Sub Directory to write timing reports i.e. ./reports/[sub-dir] (default = ./reports)

   The following explains each switch and describes when it is appropriate to use them. Switch Usage

   -netlist [path to netlist]: This switch is required and points to the Verilog, VHDL, or .db gate-level netlist that needs to be timed.

   -speed [WC|BC|BC_WC]: This switch is optional. The default is to run PrimeTime in best case / worst case mode, meaning that setup times are checked with respect to the worst case library and hold times are checked with respect to the best case library.

   WC: specifies that setup and hold times are checked with respect to the worst-case library BC: specifies that setup and hold times are checked with respect to the best-case library

   NOTE: The libraries are defined in the ../dc/scripts/setup.tcl

   -spef [path]: This switch must provide the path and filename to a SPEF parasitic file.

   Use 2.5D Parasitic Extraction data as soon as possible to provide the most accurate timing results. Note: Even though the Physical Compiler scripts write out a spef file, it is not meant for timing analysis; the Physical Compiler is not an extraction tool.

   -dspf [path]: This switch must provide the path and filename to a DSPF parasitic file.

   -load [path]: This switch must provide the path and filename to a Load parasitic file.

   -sdf [path]: This switch must provide the path and filename to an SDF parasitic file.

-clk_tree: This switch should only be used if clock trees have been added to your design. Specifying this switch removes the ideal attributes on the clocks. Also, detailed clock reports are generated when this switch is used.

-duty [45|50|55]: This switch specifies the duty cycle of the clock. The default is 50

-mode [sync|syncbypass|capture|shift] : The default is to run timing analysis in functional mode. shift times scan chain shift capture times scan chain capture logic

-write_sdf: This switch is optional. When used, an SDF file is generated.

-no_reports: This switch should only be used if you want to run PrimeTime interactively. With this switch active, the design will be setup with the constraints applied, and the pt_shell prompt will be available for interactive use.

-report_dir [sub-dir]: path to a directory where you want the timing reports to be written to.
The default is ./reports. If the directory doesn't exist, it is created.
If a sub-dir is specified, the reports are written to ./reports/[sub-dir].
If a sub-dir is specified, the logs are written to ./logs/[sub-dir].
Currently, the script generates timing models by default, in case timing models are not required, pls set the gen_model variable to false.

Analyze the results (logs and reports directories).

6. Power analysis in ptpx.
   The Power Analysis is run using the Prime Time tool. This is moved from Prime Power. The current version supports two types of Analysis

   (a) Vectorless Analysis

   (b) VCD Based Analysis

   Flow:
   The power analysis is run with the following command,

```
run_ptpx.tcl -netlist <path_to_netlist_file> \
             -para <path_to_parasitics_file> \
             -sa vcd -sa_file <path_to_VCD_file>
```

   Options:
   `-report_dir <path_to_report_dir>` : The Power Analysis reports are stored in this path. By default, the reports are stored in the ./reports directory

   `-extn [tag]` : To differentiate between different run, this option can be used.

   Analyze the results (logs and reports directories.

7. ATPG generation in tmax.
   PARSER_tmax.scr: This script is used for ATPG based on the fast sequential methodology of the TetraMax.
   Assumptions:

   - You have access to TetraMax.

- Your environment should have the $SYNOPSYS path set to point to the Synopsys tool installation directory.
- You have Verilog simulations libraries for your technology

Procedure:
The run_tmax.csh shell script is provided to set up the three environment variables that are required by the Tetramax script: The path to the netlist file, the path to the Verilog technology, the simulation library.
Example:

```
run_tmax.csh -netlist <Path + filename to netlist>
             -tech_lib ../verilog/ALL.v
```

Analyze the results (logs and reports directories).

8. Functional verification in vcs.
   (a) Copy files from directory `tmax` to directory `vcs`.
   (b) In PARSER_SERIAL.v, include the needed Verilog description.
   (c) In design code insert `timescale 1ns/1ps`.
   (d) Run VCS (`vcs PARSER_SERIAL.v +tetramax -v ../ALL.v`).

   Analyze the results (VCS running process).
9. Post-layout parasitics extraction in starrc.
   (a) Copy `.GDS` from ../hercules/DRC.
   (b) Run script hercules_saed90nm_rcx.sh.

   Analyze the results (PARSER.spf file).

4 Architectural choices, RTL compilation, and simulation with DVE

Since the Linux kernel on the server is very old (2.6.32), so the new versions of Synopsys VCS (2020.03 or 2020.12) are not compatible with the Linux kernel. If running new Synopsys VCS, it will fail to launch with errors

```
Process Size: 406032384 bytes
verdi detected abnormal termination.
Log information written to
/home/users/<username>/synopsys/syn_tut/dve/work/verdiLog/sysinfo_<date>_<time>.tar.
Please send this file to customer support.
/apps/synopsys/vcs-2020.12/bin/vcs: line 7983: 210579 Segmentation fault
 ↪  (core dumped) simv -gui -ucli
Note: Execution of simv exited with code 139
```

The version configuration in /apps/env/modulefiles/synopsys/vcs/.version has been changed to Synopsys VCS 2018.09 now.

4.1 Introduction

The Synopsys simulator is called VCS, but it is used through an interface called Discovery Visual Environment (DVE), which is an interactive Graphical User Interface (GUI) used for debugging SystemVerilog, VHDL, Verilog, and SystemC designs. Using DVE you can drag-and-drop signals in various views or use the menu options to view the signal source, trace drivers, compare waveforms, and view schematics. Use DVE to quickly find bugs in RTL or gate, assertions, testbench, and coverage.

4.2 Tasks

1. First, create a `synopsys/syn_tut` directory under your user directory `/home/users/<username>` if one does not already exist. Make sure to unzip the /home/Flow/Synopsys/Scripts/dve_32.zip directory in the syn_tut directory and move `dve` out. Also, unzip Tetramax_28nm.zip and move `tetramax_28` out. Start DVE graphical user interface (GUI) from the `work` directory. Bring up DVE and open Johnson_count.v and/or Johnson_count_dc.v using the following commands:

   ```
   cd synopsys
   mkdir syn_tut
   cd syn_tut
   cd dve
   cd work
   set NETLIST_DIR=/home/users/<username>/synopsys/syn_tut/tetramax_28 ⌟
    ↪  /pre_lay/ref/models
   ```

   Make sure the variable `LM_LICENSE_FILE` is set to `27080@server01` and `SNPSLMD_LICENSE_FILE` is set to `27000@server01`.

- for only Johnson_count.v

  ```
  vcs ../source/Johnson_count.v -R -gui -debug_all
  ```

- for only Johnson_count_dc.v

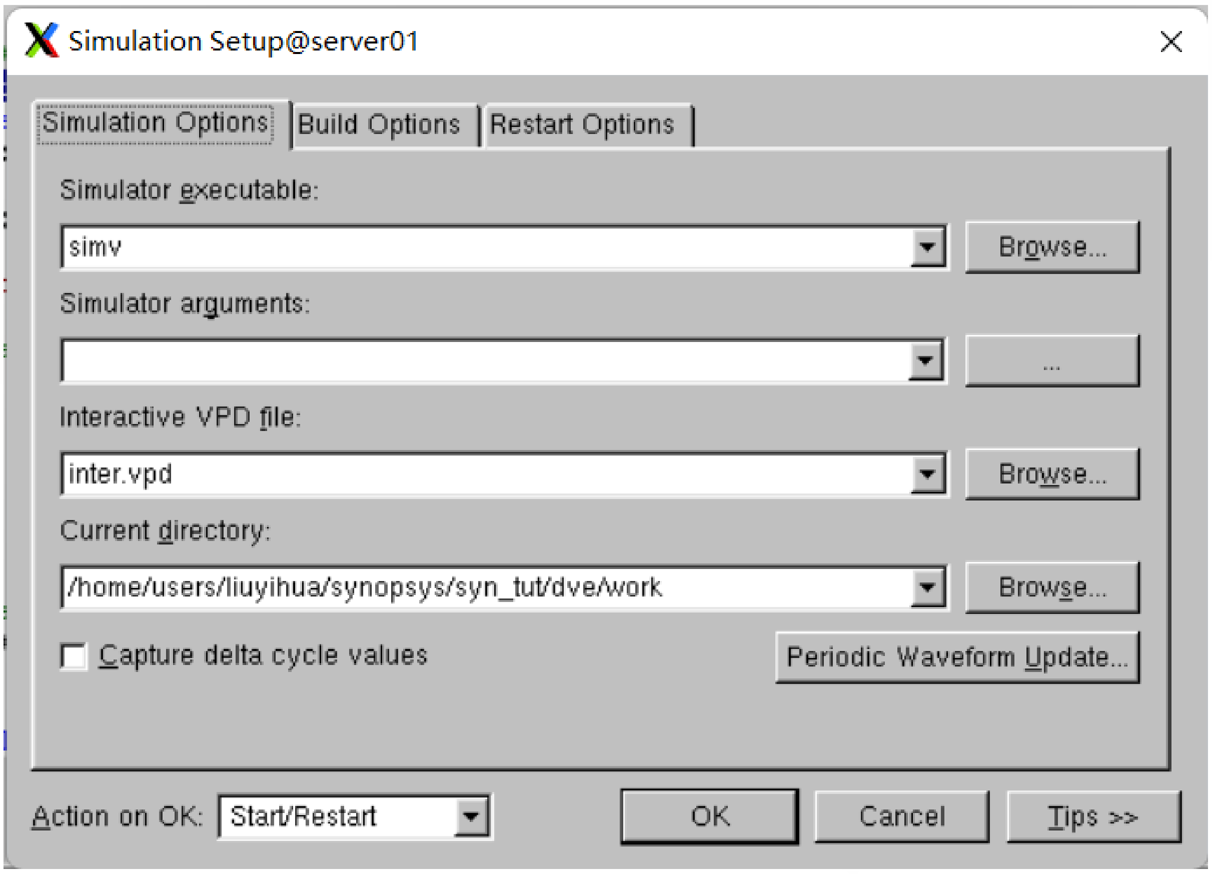
  ```
  vcs ../source/Johnson_count_dc.v -v $NETLIST_DIR/saed32nm_hvt.v
   ↪  -timescale=1ns/1ps -R -gui -debug_all
  ```

- for both Johnson_count.v and Johnson_count_dc.v together [For comparing two signals from two designs, you will need this.]

  ```
  vcs ../source/Johnson_count.v ../source/Johnson_count_dc.v
   ↪  -v $NETLIST_DIR/saed32nm_hvt.v -timescale=1ns/1ps -R
   ↪  -gui -debug_all
  ```

For Synopsys 32/28 nm EDK, just run the third command, which will parse the two design files and the library file, and compile the modules `test`, `test_dc`, and `DFFARX1_HVT`; for Synopsys 90nm EDK, just run the first command since the netlist is absent.

2. From the menu bar, select Simulator > Setup, then select click OK to start the simulation.


Simulation Setup@server01

Simulation Options | Build Options | Restart Options

Simulator executable: simv

Simulator arguments:

Interactive VPD file: inter.vpd

Current directory: /home/users/liuyihua/synopsys/syn_tut/dve/work

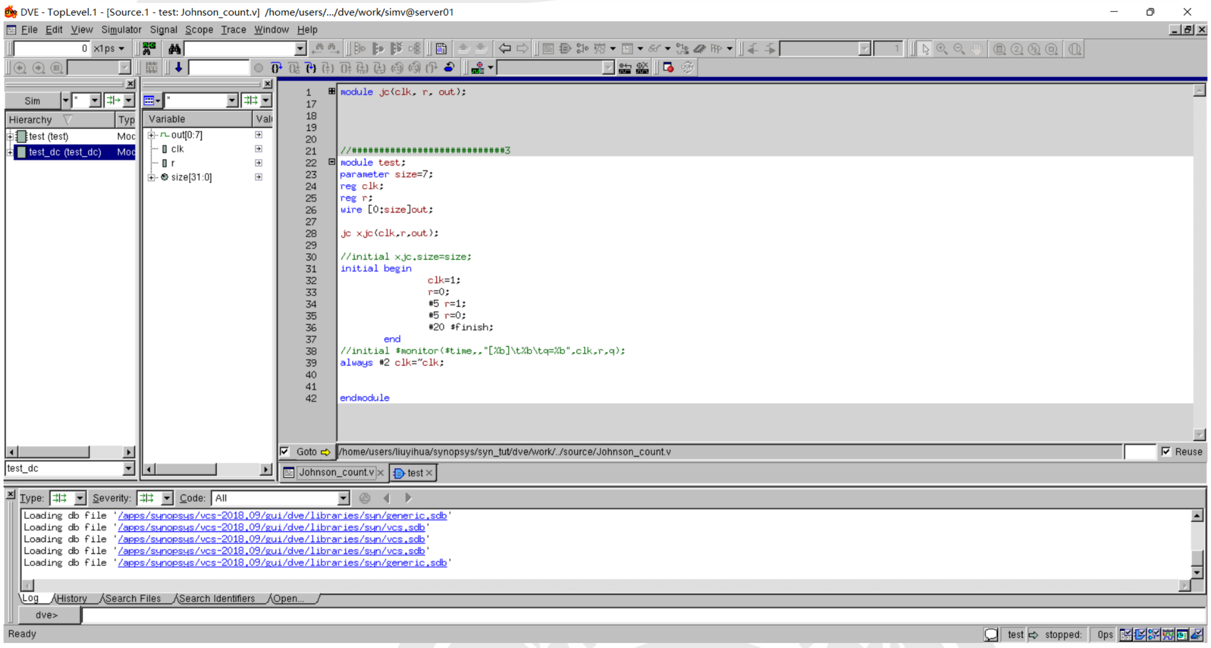
☐ Capture delta cycle values

Action on OK: Start/Restart

Figure 13: Simulation setup.

The Simulation setup has the following options:

- *Simulator Executable* - specifies the name of a simulator executable.
- *Simulator arguments* - identifies the simulator arguments.
- *Interactive VPD file* - specifies the name of the VPD file. VPD files (design database files) are platform-independent, versioned files into which you can dump the selected signals during simulation. DVE gets hierarchy, value change, and some assertion information from these files.
- *Current directory* - specifies the full path of the simulator executable.


DVE - TopLevel.1 - [Source.1 - test: Johnson_count.v] /home/users/.../dve/work/simv@server01
File Edit View Simulator Signal Scope Trace Window Help
Hierarchy
test (test)
test_dc (test_dc)
Variable
out[0:7]
clk
r
size[31:0]
Goto
/home/users/liuyihua/synopsys/syn_tut/dve/work/../source/Johnson_count.v
Johnson_count.v
test
test_dc
Type: Severity: Code: All
Log History Search Files Search Identifiers Open...
dve>
Ready
test stopped: 0ps

```
module jc(clk, r, out);




//##############################3
module test;
parameter size=7;
reg clk;
reg r;
wire [0:size]out;

jc xjc(clk,r,out);

//initial xjc.size=size;
initial begin
                clk=1;
                r=0;
                #5 r=1;
                #5 r=0;
                #20 $finish;
        end
//initial $monitor($time,,"[%b]\t%b\tq=%b",clk,r,q);
always #2 clk=~clk;


endmodule
```

```
Loading db file '/apps/synopsys/vcs-2018.09/gui/dve/libraries/syn/generic.sdb'
Loading db file '/apps/synopsys/vcs-2018.09/gui/dve/libraries/syn/vcs.sdb'
Loading db file '/apps/synopsys/vcs-2018.09/gui/dve/libraries/syn/vcs.sdb'
Loading db file '/apps/synopsys/vcs-2018.09/gui/dve/libraries/syn/vcs.sdb'
Loading db file '/apps/synopsys/vcs-2018.09/gui/dve/libraries/syn/generic.sdb'
```

Figure 14: Overview of DVE window.

A TopLevel window contains

(a) frame,

(b) menus,

(c) toolbars,

(d) status bar,

(e) pane targets.

4.3 Viewing a Waveform

- **Hierarchy Pane** - displays the scope hierarchy of the design.
- **Data Pane** - displays the variables of the selected scopes of the Hierarchy pane.

- **Source View** - Displays the source code and supports source code relative features, such as tracing driver or load, and setting line breakpoints.
- **List View** - provides a table view to display the values of signals over time.
- **Schematic View** - provides a module-based schematic to display the connectivity of the object.
- **Assertions View** - displays the summary of assertion results of simulations, including the success, failures, and incomplete ones.

To view the simulation waveform for this testbench, select all of the signals in the wave pane (clk, out, r, size), right-click, and select "Add To Waves > New Wave View". You should see an empty wave window after this step.

Using Simulator Menu->Start/Continue (shortcut is the F5 key) or the down arrow button on the toolbar, the simulation until a breakpoint is hit, the simulation finishes, or for the duration specified in the set Continue Time dialog box or toolbar time entries. When the simulation is running, the red circle Stop icon is activated, and you can click to stop the simulation.

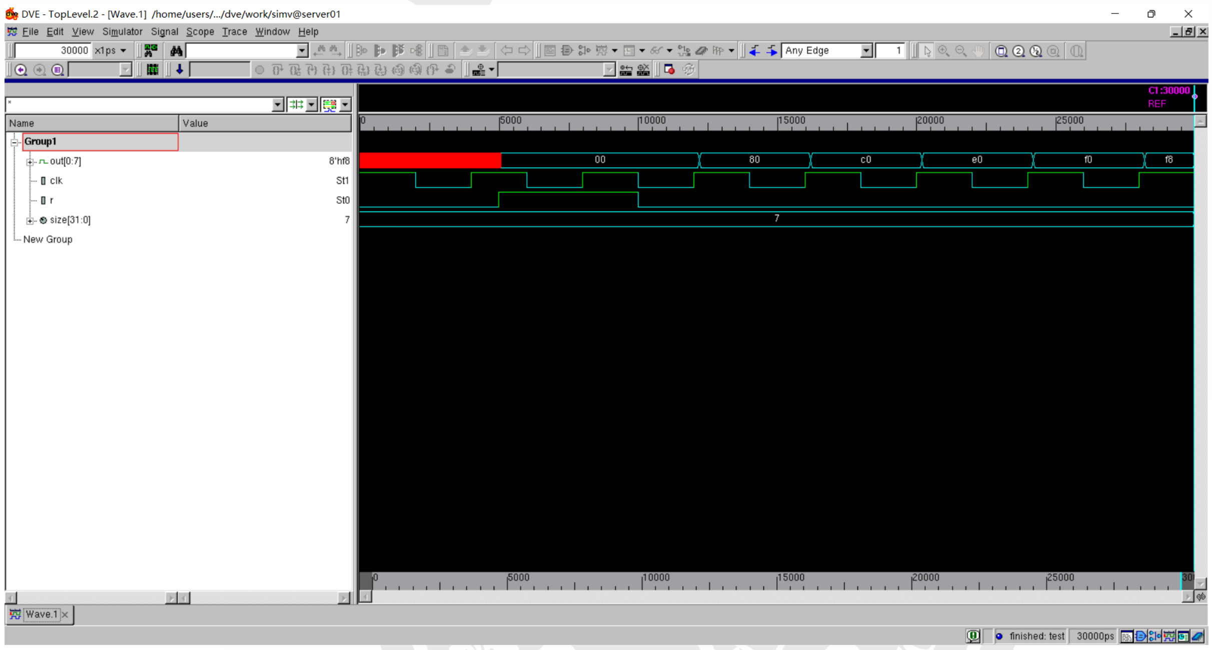

DVE - TopLevel.2 - [Wave.1] /home/users/.../dve/work/simv@server01
File Edit View Simulator Signal Scope Trace Window Help
Name
Value
Group1
out[0:7]
8'hf8
clk
St1
r
St0
size[31:0]
7
New Group
00
80
c0
e0
f0
f8
C1:30000
Any Edge
Wave.1
finished: test
30000ps

Figure 15: Wave view window.

Right-click the waveform view and click Zoom->Zoom Full to view the full range of the waveform.

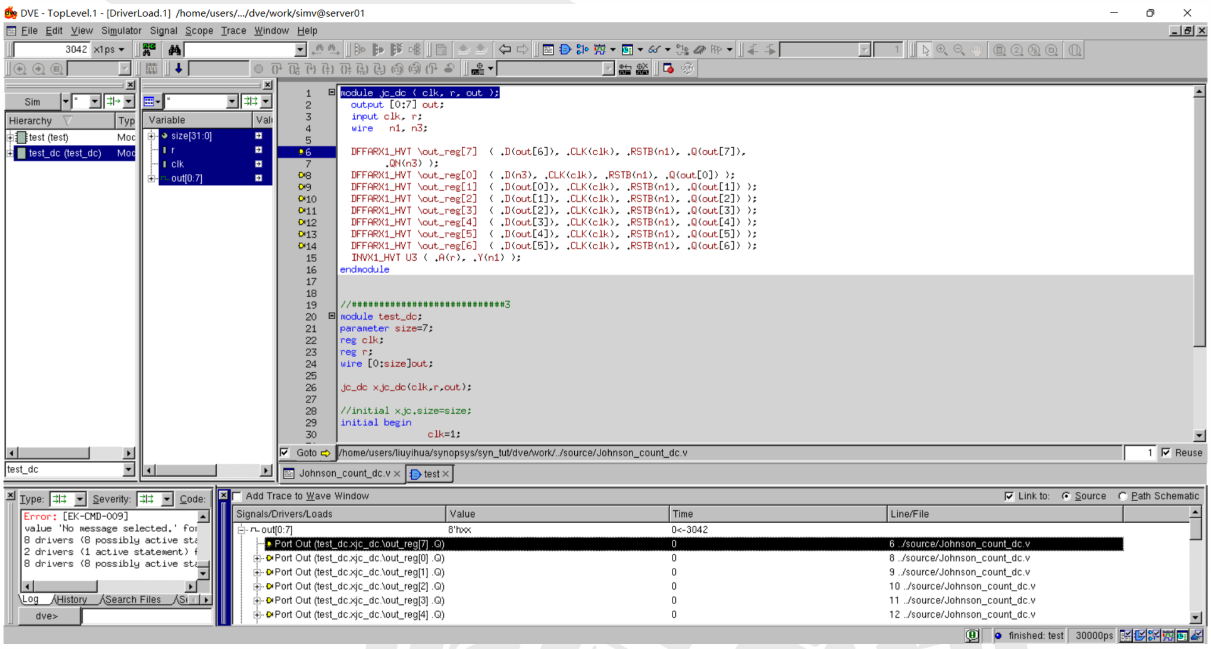
4.4 Tracing Drivers and Loads

1. Trace drivers and loads of a signal at any time to see the drivers/loads that caused a value change and see all the drivers/loads that possibly contributed to a signal value.

2. Select a signal in the signal view pane.
3. Right-click and select Trace Drivers or Trace Loads. When a driver is traced, a new Driver pane will be created if none exists in the current top-level frame. If a driver pane exists, the driver information will be added to the top of the list.

Additionally, the first driver will be highlighted in the Source view and annotated with a blue node in the gutter. In the Wave view, double-click on a waveform to see its drivers (Figure 16).

Link the Driver panes to the Source view in the same top-level frame and Path Schematic view. The link to the radio buttons at the top right of the pane shows the current linked windows. By linking a Source and Schematic view, when the object is selected in the Drivers pane, the object will also be selected in the linked views.


DVE - TopLevel.1 - [DriverLoad.1] /home/users/.../dve/work/simv@server01
File Edit View Simulator Signal Scope Trace Window Help
Sim
Hierarchy
test (test)
test_dc (test_dc)
Variable
size[31:0]
r
clk
out[0:7]
test_dc
Log History Search Files
dve>

```
module jc_dc ( clk, r, out );
  output [0:7] out;
  input clk, r;
  wire   n1, n3;

  DFFARX1_HVT \out_reg[7]  ( .D(out[6]), .CLK(clk), .RSTB(n1), .Q(out[7]),
        .QN(n3) );
  DFFARX1_HVT \out_reg[0]  ( .D(n3), .CLK(clk), .RSTB(n1), .Q(out[0]) );
  DFFARX1_HVT \out_reg[1]  ( .D(out[0]), .CLK(clk), .RSTB(n1), .Q(out[1]) );
  DFFARX1_HVT \out_reg[2]  ( .D(out[1]), .CLK(clk), .RSTB(n1), .Q(out[2]) );
  DFFARX1_HVT \out_reg[3]  ( .D(out[2]), .CLK(clk), .RSTB(n1), .Q(out[3]) );
  DFFARX1_HVT \out_reg[4]  ( .D(out[3]), .CLK(clk), .RSTB(n1), .Q(out[4]) );
  DFFARX1_HVT \out_reg[5]  ( .D(out[4]), .CLK(clk), .RSTB(n1), .Q(out[5]) );
  DFFARX1_HVT \out_reg[6]  ( .D(out[5]), .CLK(clk), .RSTB(n1), .Q(out[6]) );
  INVX1_HVT U3 ( .A(r), .Y(n1) );
endmodule


//##############################3
module test_dc;
parameter size=7;
reg clk;
reg r;
wire [0:size]out;

jc_dc xjc_dc(clk,r,out);

//initial xjc.size=size;
initial begin
                clk=1;
```

Goto /home/users/liuyihua/synopsys/syn_tut/dve/work/../source/Johnson_count_dc.v 1 Reuse

Johnson_count_dc.v test

```
Error: [EK-CMD-009]
value 'No message selected.' fo
8 drivers (8 possibly active st
2 drivers (1 active statement) f
8 drivers (8 possibly active st
```

Add Trace to Wave Window   Link to: (● Source) ○ Path Schematic

Signals/Drivers/Loads	Value	Time	Line/File
out[0:7]	8'hxx	0<-3042	
Port Out (test_dc.xjc_dc.\out_reg[7] .Q)		0	6 ../source/Johnson_count_dc.v
Port Out (test_dc.xjc_dc.\out_reg[0] .Q)		0	8 ../source/Johnson_count_dc.v
Port Out (test_dc.xjc_dc.\out_reg[1] .Q)		0	9 ../source/Johnson_count_dc.v
Port Out (test_dc.xjc_dc.\out_reg[2] .Q)		0	10 ../source/Johnson_count_dc.v
Port Out (test_dc.xjc_dc.\out_reg[3] .Q)		0	11 ../source/Johnson_count_dc.v
Port Out (test_dc.xjc_dc.\out_reg[4] .Q)		0	12 ../source/Johnson_count_dc.v

finished: test 30000ps

Figure 16: TopLevel window with tracing drivers and loads.

4.5 Comparing Signals, Scopes, and Groups

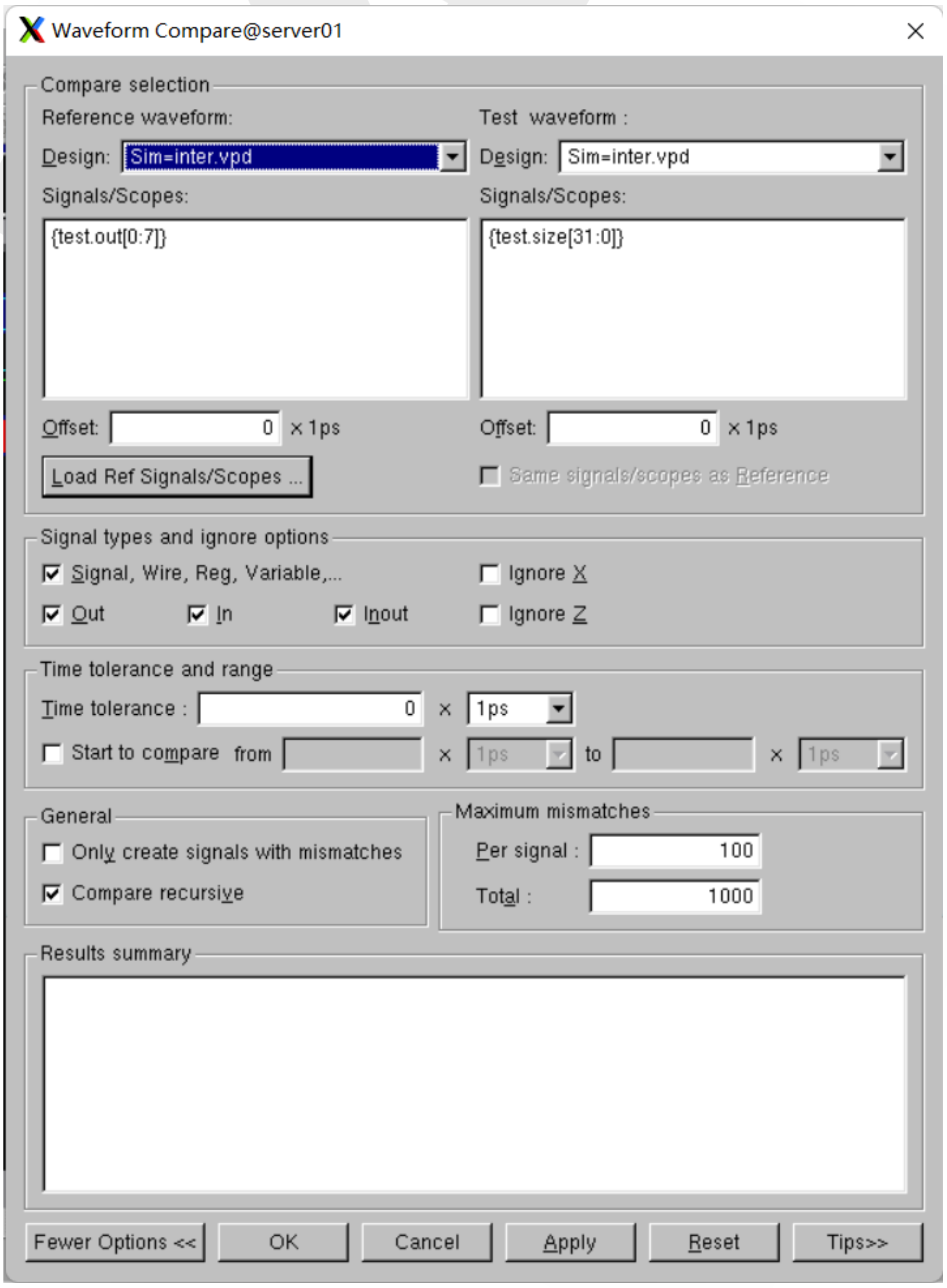
For comparing individual signals with the same bit numbers, scopes (for comparing variable children), buses, or groups of signals from one or two designs.

To view a comparison

1. Select one or two signals, signal groups, scopes, or buses from the Signal pane of the Wave view.
2. Right-click and select Compare...
3. Click Load Reference Signals/Scopes and select the text file with the signals and scopes to reference.

- Note 1: If comparing two designs from the root, then the reference waveform region and test waveform region can be empty.
- Note 2: If you don't have a reference file, you can skip this step and still compare the signals you have in your design.

4. Click the More Options button (Figure 17)
   The dialog box is expanded, and additional options are displayed.



Waveform Compare@server01

Compare selection

Reference waveform:
Design: Sim=inter.vpd
Signals/Scopes:
{test.out[0:7]}
Offset: 0 × 1ps
Load Ref Signals/Scopes ...

Test waveform :
Design: Sim=inter.vpd
Signals/Scopes:
{test.size[31:0]}
Offset: 0 × 1ps
☐ Same signals/scopes as Reference

Signal types and ignore options
☑ Signal, Wire, Reg, Variable,...
☑ Out ☑ In ☑ Inout
☐ Ignore X
☐ Ignore Z

Time tolerance and range
Time tolerance : 0 × 1ps
☐ Start to compare from × 1ps to × 1ps

General
☐ Only create signals with mismatches
☑ Compare recursive

Maximum mismatches
Per signal : 100
Total : 1000

Results summary

Figure 17: The Waveform Compare dialog box.

5. In the Signal types and ignore options section, select the signal types to compare

and select ignore options.
For example, if Ignore X is selected and if the reference signal value is X, there is always a match, whatever the values of the Test Signal.

6. Enter a Time Tolerance to filter out mismatch values that have time ranges smaller than the tolerance range.
7. In the General section, select to compare recursively or to only create signals with mismatches.
8. Click Apply to start the comparison and keep the dialog box open. Or click OK to start the comparison and close the dialog box (to open it at any time from the Signal pane CSM). Results are displayed in the current Wave view.

4.6 (Optional) Selecting signal in the Schematic View

You can open the Schematic View by right-clicking the module in the Hierarchy view and clicking "Show Schematic" or clicking the "Show Schematic" button on the toolbar or pressing Ctrl+2.

To select a signal in the Schematic view

1. Enter the signal name in the Find toolbar box in the Schematic view and then click the Find Next toolbar button. The signal is highlighted in the schematic.
2. With the signal selected, click the Trace Drivers toolbar button or select the Trace Drivers menu item. The signal is highlighted in purple.

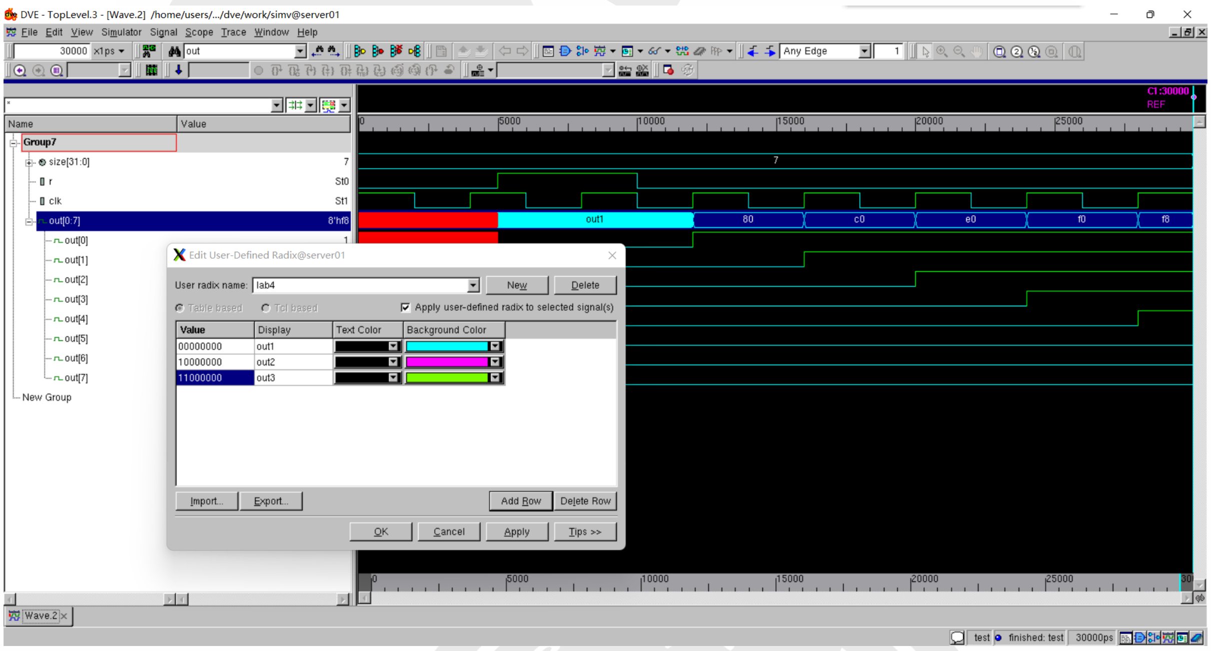
4.7 Using User-defined Radixes

This section describes how to create, edit, import, and export user-defined radices. You can define a custom mnemonic mapping from values to strings for display in the Wave view.

To create, delete, import, and export a user-defined radix

1. Select Signal->Set Radix->User-Defined->Edit...
2. Click New, enter a radix name and then hit the Enter key on your keyboard. All buttons on the Edit User-defined Radix get enabled. *Remember to switch Tcl based to Table based.*
3. Click Add Row to activate a row for the user-defined radix and perform the following steps:
   - Select the text and background colors for each row entry.
   - Select the radix, click a cell in the Value and Display column, then enter the values. The radix is edited.
4. Select a row and then click Delete Row. The row is deleted.

5. Select a radix from the Radix Table Name drop-down and click the Delete button. The radix is deleted.
6. Click Import, then browse and select the desired radix. The radix is imported.
7. Click Export, select the radix and then enter a radix name. The radix is exported.
8. Select the Apply user-defined radix to selected signal(s) checkbox (Figure 18). The user-defined radix is applied to the selected signal in the Wave view.
9. Click OK or Apply to save the user-defined radix.



Edit User-Defined Radix@server01

User radix name: lab4

Table based · Tcl based · [✓ Apply user-defined radix to selected signal(s)]

Value	Display	Text Color	Background Color
00000000	out1	⬛	🟦
10000000	out2	⬛	🟪
11000000	out3	⬛	🟩

Figure 18: The Edit User-defined Radix dialog box with Wave view window.

Export the User-Defined Radix as a TXT file.

To exit DVE write exit in the command line on the console.

```
DVE> exit
```

5 Logic Synthesis

Synopsys Design Compiler (DC) is composed of

- Design Compiler Graphical
- DC Ultra (TM)
- DFTMAX (TM)
- Power Compiler (TM)
- DesignWare (R)
- DC Export (TM)
- Design Vision (TM)
- HDL Compiler (TM)
- VHDL Compiler (TM)
- DFT Compiler
- Design Compiler (R)

5.1 Design Compiler

5.1.1 Introduction

Synopsys provides an integrated RTL synthesis solution. Using Design Compiler tools, you can:

- Produce fast, area-efficient ASIC designs by employing user-specified or standard-cell libraries
- Translate designs from one technology to another
- Explore design tradeoffs involving design constraints such as timing, area, and power under various loading, temperature, and voltage conditions
- Synthesize and optimize finite state machines
- Integrate netlist inputs and netlist or schematic outputs into third-party environments while still supporting delay information and place and route constraints
- Create and partition hierarchical schematics automatically

5.1.2 Basic synthesis flow

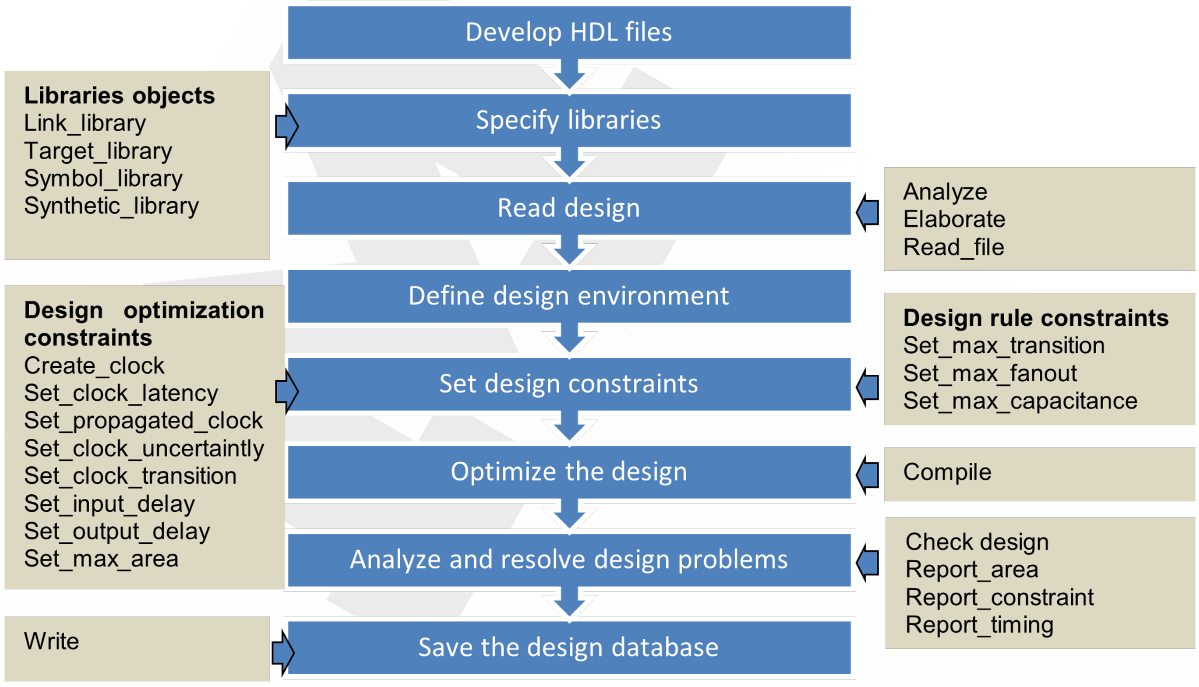


Figure 19: Basic synthesis flow.

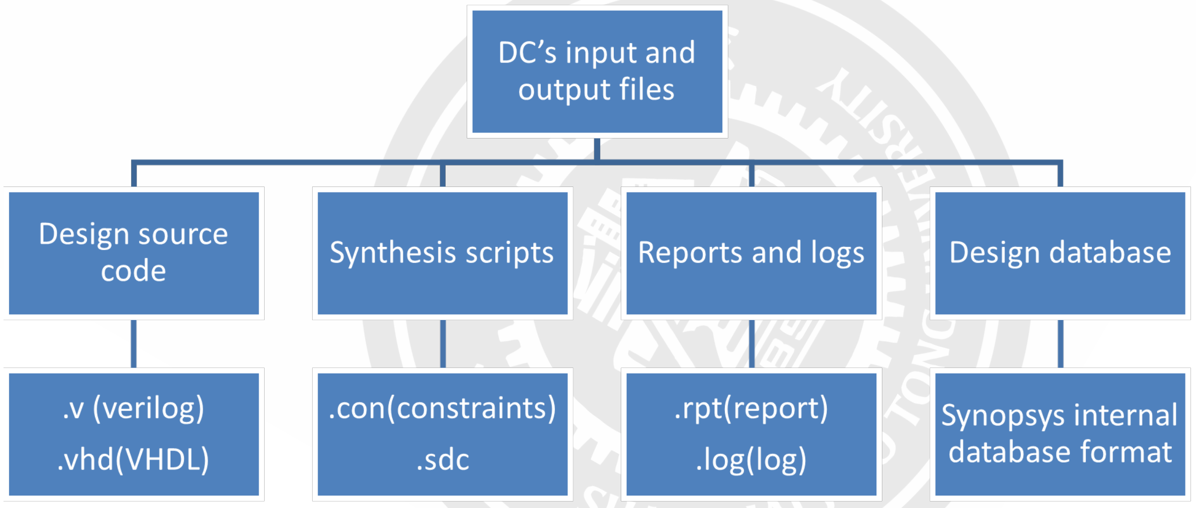


Figure 20: DC's input and output files.

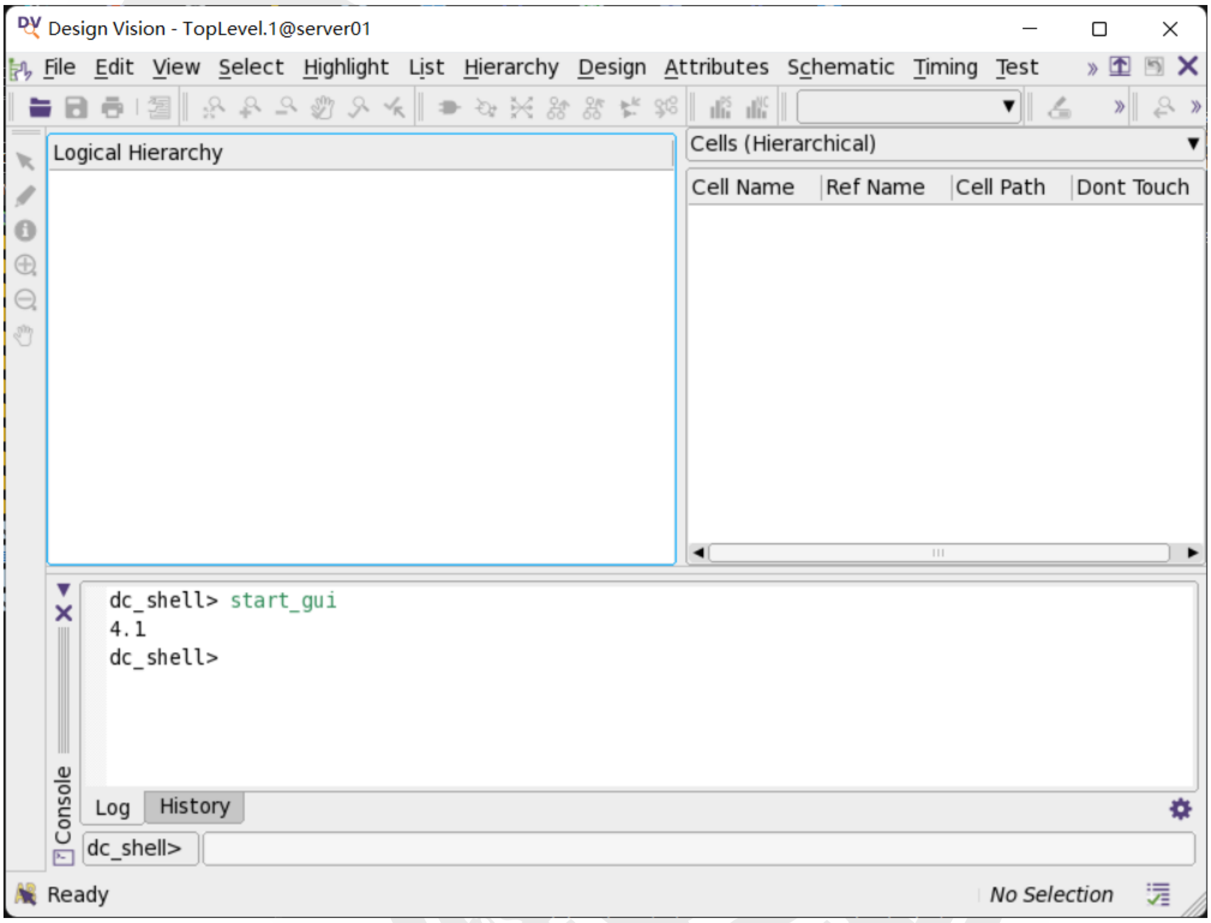
5.1.3 Tasks

1. First, create a synopsys/syn_tut directory if one does not already exist. Make sure to unzip the /home/Flow/Synopsys/Scripts/Dc_32_paths.zip directory in the

syn_tut directory and move `dc_32` out. If using Synopsys 90nm EDK, make a directory dc_90 and copy contents of dc_32 to it. Start Design Compiler graphical user interface (GUI) from the `work` directory. To start it, use the following commands:

```
cd synopsys
mkdir syn_tut
cd syn_tut
cd dc_32   # or cd dc_90
cd work
dc_shell
```

Then in the DC Shell, run `start_gui`. This opens the DC top-level GUI window:


Design Vision - TopLevel.1@server01
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test
Logical Hierarchy
Cells (Hierarchical)
Cell Name
Ref Name
Cell Path
Dont Touch
Console
Log
History
Ready
No Selection

```
dc_shell> start_gui
4.1
dc_shell>
```

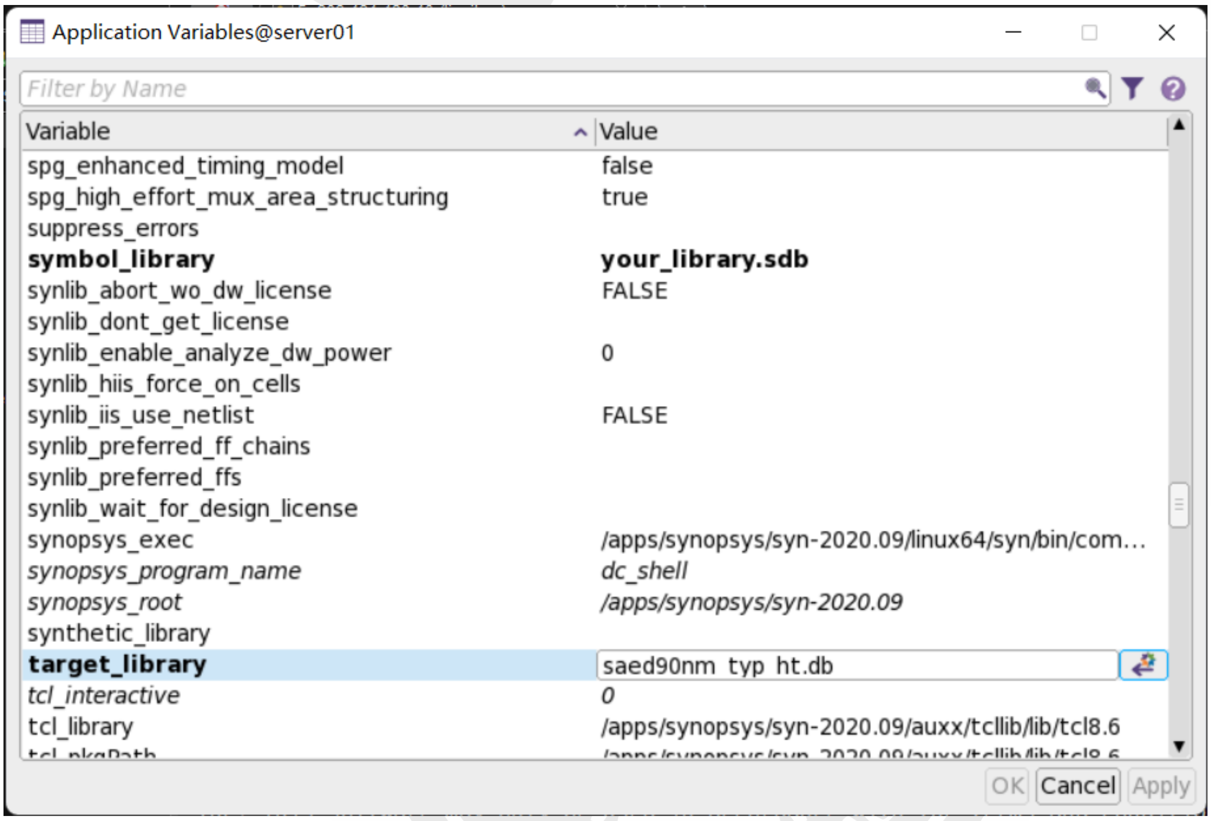
Figure 21: DC Top-Level GUI Window.

Of course, you can use `dc_shell` commands during the flow without GUI, which is even much more convenient as long as you are familiar with them.

2. Second, setup the libraries:
   Go to File -> Setup and then set the Link and Target libraries to

   - The `search_path` is set to ". /apps/synopsys/syn-2020.09/libraries/syn /apps/synopsys/syn-2020.09/dw/syn_ver /apps/synopsys/syn-2020.09/dw/sim_ver" by default. For Synopsys 90nm EDK, append /home/pdk/synopsys_edk/edk90/synopsys_design_flow_using_90nm_library/models/ to it (the `models` directory is the same as the `dc/models` directory); for Synopsys 32/28nm EDK, append /home/pdk/synopsys_edk/edk32/models (the `edk32` directory is the same as the `/home/Flow/Synopsys/Scripts/icc_32/ref` directory).

- For Synopsys 90nm EDK, write "* saed90nm_typ_ht.db" in `link_path`; for Synopsys 32/28nm EDK, write "* saed32hvt_tt1p05v25c.db".
- For Synopsys 90nm EDK, write "saed90nm_typ_ht.db" in `target_library`; for Synopsys 32/28nm EDK, write "saed32hvt_tt1p05v25c.db".
- You can leave the `symbol_library` as it is (`your_library.sdb`). When you add the libraries, also make sure you explicitly remove the "your_library.db" default; otherwise, you will get a warning message later.



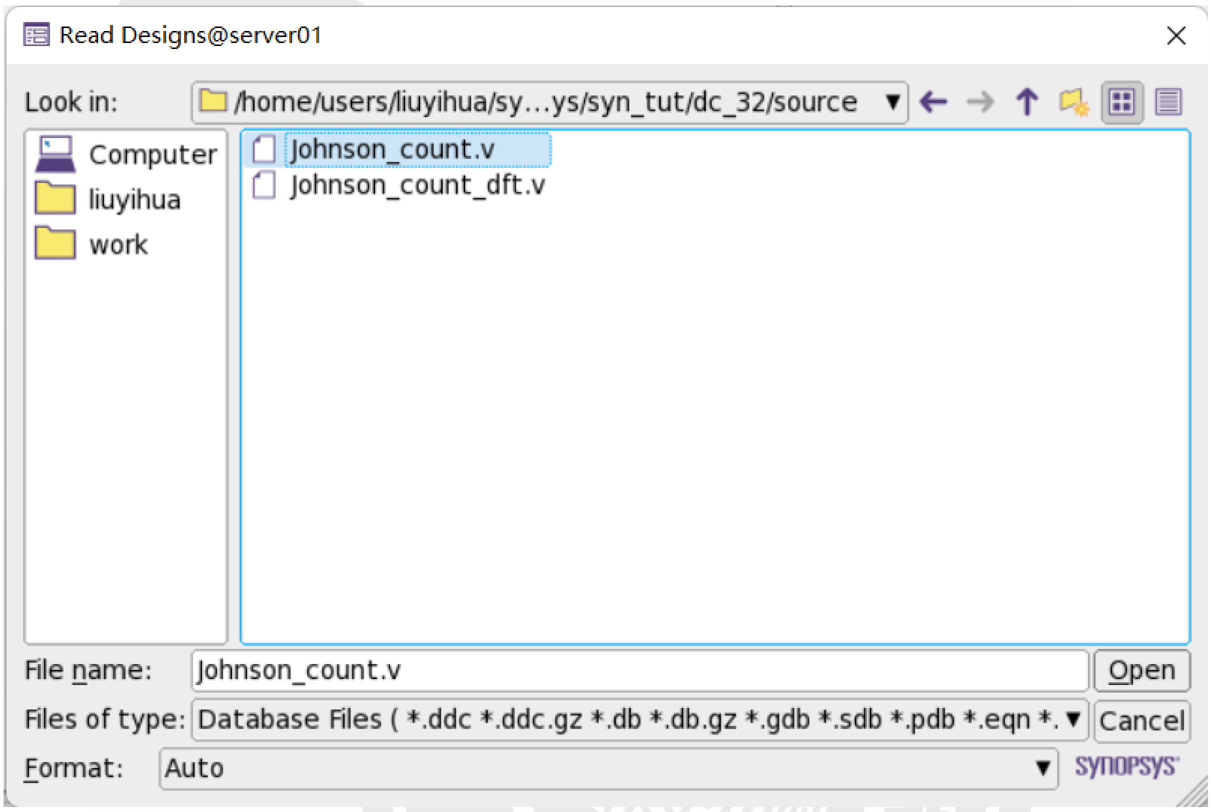
Variable	Value
spg_enhanced_timing_model	false
spg_high_effort_mux_area_structuring	true
suppress_errors	
symbol_library	**your_library.sdb**
synlib_abort_wo_dw_license	FALSE
synlib_dont_get_license	
synlib_enable_analyze_dw_power	0
synlib_hiis_force_on_cells	
synlib_iis_use_netlist	FALSE
synlib_preferred_ff_chains	
synlib_preferred_ffs	
synlib_wait_for_design_license	
synopsys_exec	/apps/synopsys/syn-2020.09/linux64/syn/bin/com…
synopsys_program_name	*dc_shell*
synopsys_root	*/apps/synopsys/syn-2020.09*
synthetic_library	
target_library	saed90nm_typ_ht.db
tcl_interactive	*0*
tcl_library	/apps/synopsys/syn-2020.09/auxx/tcllib/lib/tcl8.6

Figure 22: Application Variables Window.

```
set_app_var search_path {. /apps/synopsys/syn-2020.09/libraries/syn
↪  /apps/synopsys/syn-2020.09/dw/syn_ver
↪  /apps/synopsys/syn-2020.09/dw/sim_ver /home/pdk/synopsys_edk/ed⌋
↪  k90/synopsys_design_flow_using_90nm_library/models}
set_app_var target_library saed90nm_typ_ht.db
set_app_var link_path  {* saed90nm_typ_ht.db}
set_app_var search_path {. /apps/synopsys/syn-2020.09/libraries/syn
↪  /apps/synopsys/syn-2020.09/dw/syn_ver
↪  /apps/synopsys/syn-2020.09/dw/sim_ver
↪  /home/pdk/synopsys_edk/edk32/models}
set_app_var target_library saed32hvt_tt1p05v25c.db
set_app_var link_path  {* saed32hvt_tt1p05v25c.db}
```

The asterisk means all the libraries in the memory.

3. Third, read in the source file. Read design commands are derived from the read_file -format VHDL or read_file -format Verilog commands. In our design, we used -format Verilog:


Read Designs@server01

Look in: /home/users/liuyihua/sy…ys/syn_tut/dc_32/source

Computer
liuyihua
work

Johnson_count.v
Johnson_count_dft.v

File name: Johnson_count.v
Files of type: Database Files (*.ddc *.ddc.gz *.db *.db.gz *.gdb *.sdb *.pdb *.eqn *.
Format: Auto

Figure 23: Read Designs Box.

Go to File -> Read and set Johnson_count.v as the source file as seen in Figure 23:

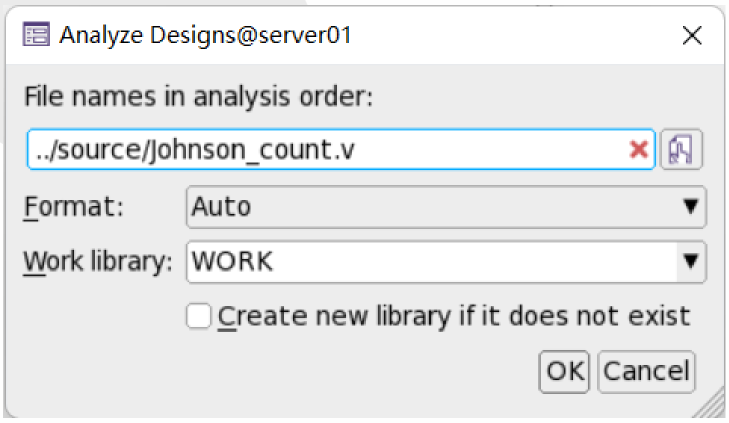
```
read_file -format verilog {/home/users/<username>/synopsys/syn_tut/
↪  dc_32/source/Johnson_count.v}
```

4. Fourth, use the `analyze` and `elaborate` commands.
To run the analyze command: Go to File -> Analyze. After Analyze is clicked, the window seen in Figure 24 should appear. The analyze command does the following:

   - Reads an HDL source file

- Checks it for errors (without building generic logic for the design)
- Creates HDL library objects in an HDL-independent intermediate format
- Stores the intermediate files in a defined location

Click the small button on the right of the textbox of "File names in analysis order". In the pop-up dialog "Specify File names in analysis order:@server01", click the "+ Add" button, click the small button on the right of the textbox, and select ../source/Johnson_count.v. Click OK. The File names in analysis order are now ../source/Johnson_count.v. Click OK.


Analyze Designs@server01

File names in analysis order: ../source/Johnson_count.v

Format: Auto

Work library: WORK

Create new library if it does not exist

Figure 24: Analyze Designs Box.

If the analyze command reports errors, fix them in the HDL source file and run `analyze` again. After a design is analyzed, reanalyze it only when it changes.

```
analyze -library WORK -format verilog {../source/Johnson_count.v}
```

The expected output is:

```
Running PRESTO HDLC
Compiling source file ../source/Johnson_count.v
Presto compilation completed successfully.
```

Click the Save button to save the design to johnson_count.ddc

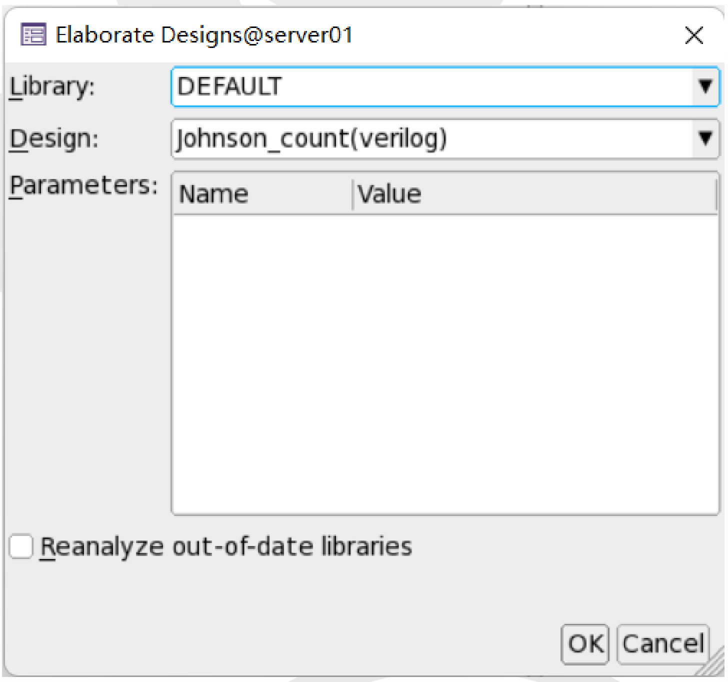
```
write -hierarchy -format ddc
```

To run the `elaborate` command: Go to File -> Elaborate. After Elaborate is clicked, the window seen in Figure 25 should appear. The `elaborate` command does the following:

- Translates the design into a technology-independent design (GTECH) from the intermediate files produced during the analysis
- Allows changing of parameter values defined in the source code
- Allows VHDL architecture selection

- Replaces the HDL arithmetic operators in the code with DesignWare components
- Automatically executes the link command, which resolves design references

In the Elaborate Designs@server01 dialog, enter "DEFAULT" to the Library textbox, then johnson_count(verilog) is automatically selected as the Design. Click OK


Elaborate Designs@server01

Library: DEFAULT

Design: Johnson_count(verilog)

Parameters:

Name	Value

☐ Reanalyze out-of-date libraries

Figure 25: Elaborate Designs Box.

```
elaborate Johnson_count -architecture verilog -library DEFAULT
```

The expected output is:

```
    Running PRESTO HDLC

Inferred memory devices in process
        in routine Johnson_count line 8 in file
                '../source/Johnson_count.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       out_reg       | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
Presto compilation completed successfully. (Johnson_count)
Warning: Design 'Johnson_count' was renamed to 'Johnson_count_1' to avoid
        a conflict with another design that has the same name but
different parameters. (LINK-17)
Elaborated 1 design.
Current design is now 'Johnson_count_1'.
```

You can switch between designs from the Design List drop-down list on the toolbar.

5. Fifth, for a design to be complete, it needs to be connected to all of the library components and designs it references. So to perform a name-based resolution of design references for the current design, use the `link` command.

```
dc_shell> link
  Linking design 'Johnson_count_1'
  Using the following designs and libraries:
  -----------------------------------------------------------------↩
  ↪  ---------
  * (2 designs)                /home/users/<username>/synopsys/syn_t↩
  ↪  ut/dc_32/work/Johnson_count_1.db,
  ↪  etc
  * (2 designs)                /home/users/<username>/synopsys/syn_t↩
  ↪  ut/dc_90/work/Johnson_count_1.db,
  ↪  etc
  saed32hvt_tt1p05v25c (library)
  ↪  /home/pdk/synopsys_edk/edk32/models/saed32hvt_tt1p05v25c.db
  saed90nm_typ_ht (library)    /home/pdk/synopsys_edk/edk90/synopsys↩
  ↪  _design_flow_using_90nm_library/models/saed90nm_typ_ht.db

1
```

6. Sixth, The `check_design` command needs to be run. The `check_design` command checks the internal representation of the current Synopsys Design Constraints for consistency and issues errors and warning messages as appropriate.

7. Seventh, set design optimization constraints, source the file constr.sdc. The optimization constraints comprise:

   - Timing constraints (performance and speed)
   - Input and output delays (synchronous paths)
   - Minimum and maximum delay (asynchronous paths)
   - Maximum area (number of gates)

   Synopsys Design Constraints (SDC) include many commands to indicate designers' constraints. It is a set of commands. In addition to resources in this lab, please refer to the documents for details:

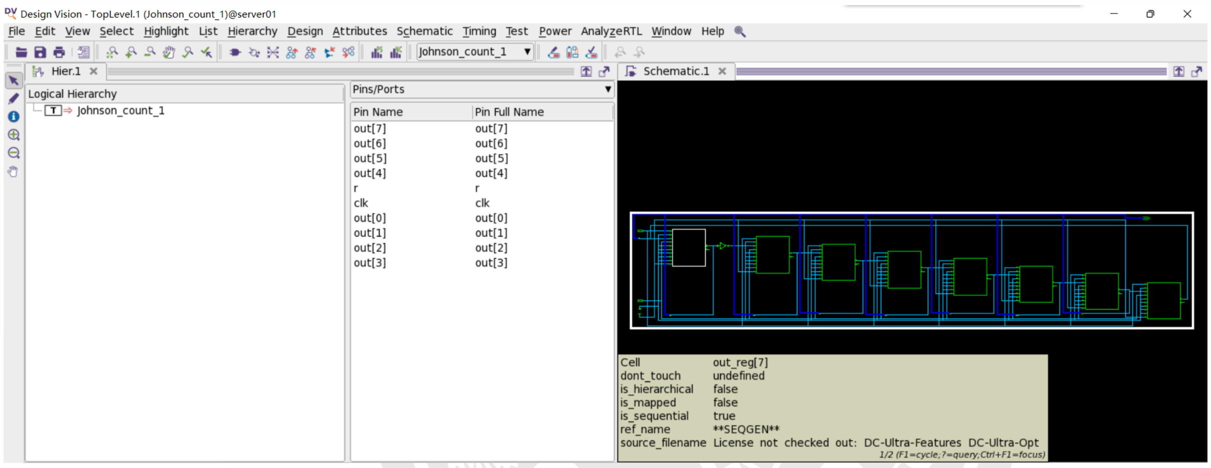
   - http://www.vlsi-expert.com/2011/02/synopsys-design-constraints-sdc-basics.html
   - http://www.cs.utah.edu/~elb/cadbook/color-figs/Chapter13-Example.pdf
   - https://free-online-ebooks.appspot.com/enc/14.17/fetxtcmdref/

   Using SDC, we can set timing constraints. We want to design a circuit, but we want

it to be able to communicate with its neighbor circuits efficiently. Therefore we kind of required to have (for design purposes) outputs loads (as if there is another circuit attached) and input buffer/inverter cell (that drives the input). Since we use cells for the input, we don't need to define the input loads. Those cells already have output loads (cells' outputs are our inputs). We use registers for inputs other than the clock. In this case, we used d-flip-flop.
Synopsys 32/28nm Generic Library contains predefined cell/gates/blocks. You can use `gedit` to open the lib file `models/saed90nm_typ_ht_pg.lib`. This library is for high threshold voltage (HVT) cells. NBUFFX16_HVT is a buffer for clock signals. A is its input, and Y is its output. A is going to be used by ScanMasterClock. SDFFARX1_HVT is a Scan Pos Edge DFF (D flip-flop) w/Async Low-Active Reset. D is its input, and Q is its output. You can see that we assign these pins with our signal. These cells drive our input signals, so their output will be our input. Normal inputs need to come to our circuit in an order with the help of the clock; also, they need to be kept in short-term memory. Register cells are for this.
To source constraints with .sdc (Synopsys Design Constraints) format:

```
dc_shell> source ../source/constr.sdc
```

8. After all steps mentioned above are completed, right-click on the Johnson_count_1 located under the Logical Hierarchy tab. Then select Schematic View, *double-click the Johnson_count_1 cell to expand the hierarchy, switch the "Cells (Hierarchical)" view to the "Pins/Ports" view*, and a Schematic seen in Figure 26 should appear.


Design Vision - TopLevel.1 (Johnson_count_1)@server01
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Window Help
Johnson_count_1
Hier.1
Schematic.1
Logical Hierarchy
Johnson_count_1
Pins/Ports

Pin Name	Pin Full Name
out[7]	out[7]
out[6]	out[6]
out[5]	out[5]
out[4]	out[4]
r	r
clk	clk
out[0]	out[0]
out[1]	out[1]
out[2]	out[2]
out[3]	out[3]

```
Cell             out_reg[7]
dont_touch       undefined
is_hierarchical  false
is_mapped        false
is_sequential    true
ref_name         **SEQGEN**
source_filename  License not checked out: DC-Ultra-Features DC-Ultra-Opt
                 1/2 (F1=cycle;?=query;Ctrl+F1=focus)
```

Figure 26: Design View From DC.

9. The `compile` command performs logic and gate-level synthesis and optimization on the current design. This command specifies that the sequential elements in the final design must exactly match the descriptions specified in the HDL. The command uses attributes/directives with the -exact_map option to ensure that the results of compile are predictable. Optimization is controlled by user-specified constraints on the design. To perform this command use the following:

```
dc_shell> compile -exact_map
```

The compiled design produces a gate-level description seen in Figure 27. To retrieve this image, once again right-click on the Johnson_count_1 located under the Logical Hierarchy tab, *expand the hierarchy*, then select Schematic View.

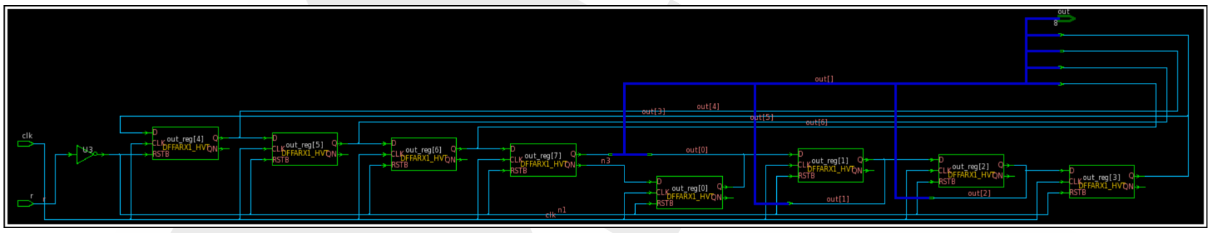


Figure 27: Design View From DC After Compile.

10. Design Compiler can generate numerous reports on the results of design synthesis and optimization. Reports are used to analyze and resolve any design problems or to improve synthesis results.
    In `dc_shell`, run `report_area`, `report_constraint`, and `report_timing`.
11. Write a design from memory to .ddc (Synopsys internal database format) by using the command below (format is still .v and .vhdl):

    ```
    dc_shell> write -hierarchy -format verilog -output
     ↪  ../results/Johnson_count_dcl.v
    ```

    At any time, for permission issues like "Error: Can't open export file" ... (EXPT-4) "Error: Write command failed. (UID-25)", you can write to another path.
12. The next step will be to run DFT Compiler. In order to start DFT Compiler (Design for Test Compiler), reset the design using the following command:

    ```
    dc_shell> reset_design
    ```

5.2 DFTC (Design for Test Compiler)

5.2.1 Introduction

DFT Compiler is used to:

- check RTL and mapped designs for DFT violations
- insert scan chains into designs
- export all the required files for downstream tools

DFT Compiler requires the user to specify a scan style in order to perform scan synthesis. A scan style dictates the appropriate scan cells to insert during optimization. The scan style, which is selected is used on all modules of design. There are three types of scan styles available in DFT Compiler:

- Multiplexed flip-flop
- Clocked scan

- Level-sensitive scan design

To specify the scan style of this design, the following command is used:

```
dc_shell> set test_default_scan_style
```

Using this command sets the scan style to the default multiplexed_flip_flop scan style.

5.2.2 Basic Scan Synthesis Flow

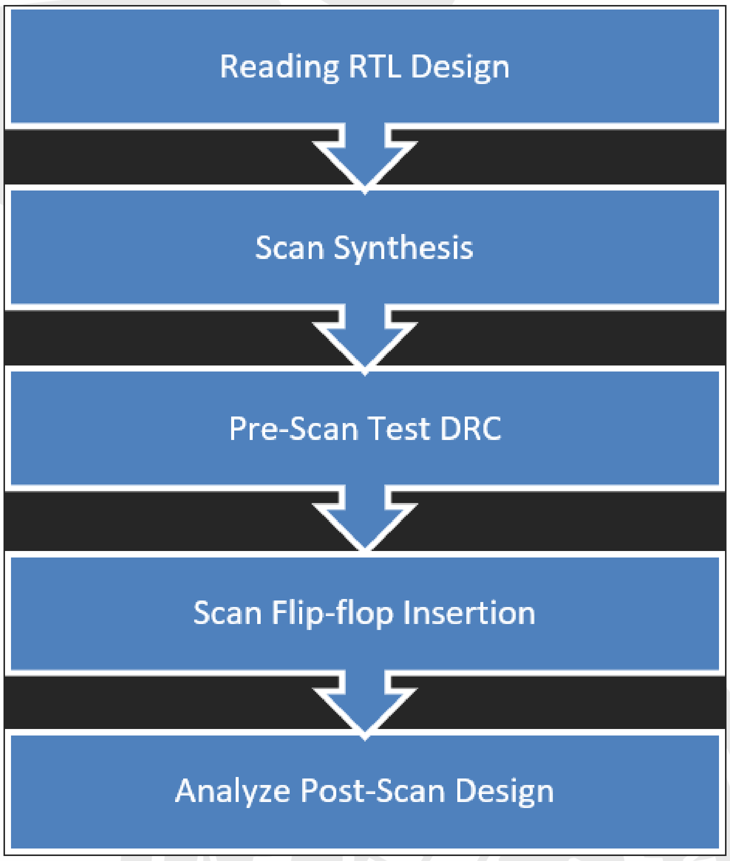


Figure 28: Basic Scan Synthesis Flow.

5.2.3 Tasks

1. First, specify the scan style. To specify the scan style of this design, use the command introduced above.
2. Second, read the design by using the following command:

   ```
   dc_shell> read_verilog ../source/Johnson_count_dft.v
   ```

   After reading the design, see the schematic view of Johnson_count by clicking `johnson_count` in the Logical Hierarchy and click Menu->Schematic->New Schematic View. *Still, expand the hierarchy. You can right-click the schematic view and click "Zoom Fit All" to have a better view.*

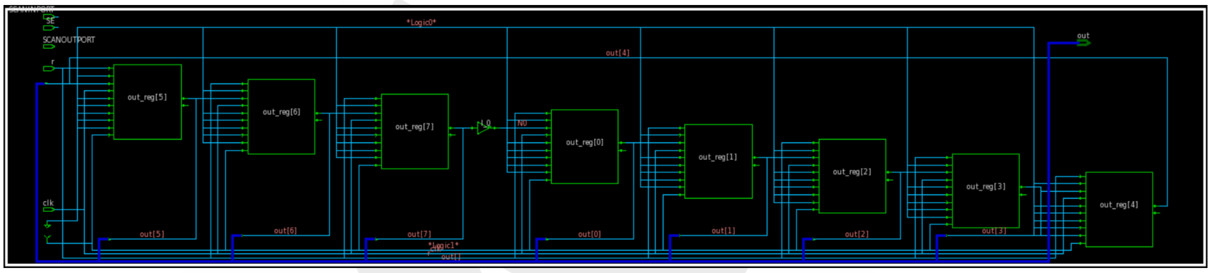


Figure 29: Schematic view of Johnson_count before DFT.

The intended schematic view is shown in Figure 29.

3. Third, source constraints with .sdc format by using the command below:

   ```
   dc_shell> source ../source/constr.sdc
   ```

4. Fourth, assign the port types to the signals that will be used in DFT. For example, ScanMasterClock, Reset, and ScanEnable. To give port types to signals, use the following commands:

   ```
   set_dft_signal -view exist -type ScanMasterClock -timing {45 55}
   ↪  -port clk
   set_dft_signal -view exist -type Reset -active 1 -port r
   set_dft_signal -view exist -type ScanEnable -active 1 -port SE
   report_dft_signal -view exist
   ```

   - set_dft_signal specifies the DFT signal types for DRC and DFT insertion.
   - -view exist option shows that ports which already exist in design, and if the port must use first time, apply the `-view spec` option.
   - -active option specifies the active sense of the port (high or low).

5. Sixth, create a test protocol to configure the design for scan test by using the command:

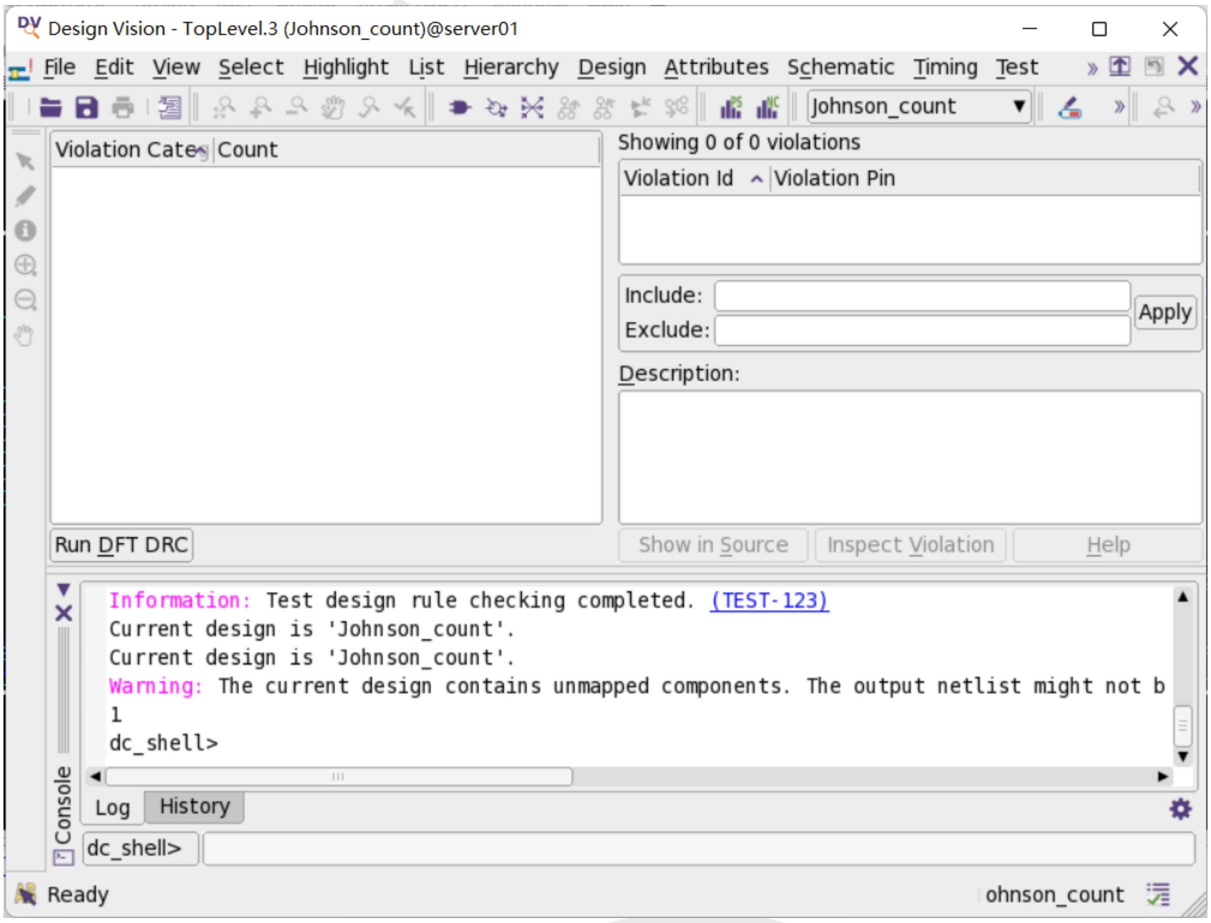
   ```
   dc_shell> create_test_protocol
   ```

   This creates a test protocol that DFT Compiler uses for test design rule checking as well as pattern generation and vector formatting steps.

6. Seventh, check the design respective to the multiplexed-flip-flop scan style, which is defined using set test_default_scan_style. Run design rule checking by using the following command:

   ```
   dc_shell> dft_drc
   ```

   This command checks violations. If violations are reported, change the RTL code and repeat steps 5 and 6. If no violations are reported, proceed to scan synthesis.


Design Vision - TopLevel.3 (Johnson_count)@server01

```
Information: Test design rule checking completed. (TEST-123)
Current design is 'Johnson_count'.
Current design is 'Johnson_count'.
Warning: The current design contains unmapped components. The output netlist might not b
1
dc_shell>
```

If there are no violations after dft_drc, it means that the protocol is working, so write out the complete protocol with .spf format for later use in the Mapped Flow.

```
dc_shell> write_test_protocol -out ../source/Johnson_count_test.spf
```

7. Eighth, create a clock object and define its waveform in the current design. 1000 specifies the period of the clock waveform in library time units. Then set input delays on SCANINPORT and on SE relative to a clock signal.

```
create_clock clk -period 1000
set_input_delay 250 SCANINPORT -clock clk
set_input_delay 150 SE -clock clk
```

8. Scan Insertion will change the design to escape from negative effects (such as Scan Registers' possible influence that may increase the area of the schematic and increase the fan-out on the nets), which must be taken into consideration during the synthesis. That's why synthesis uses scan triggers to prevent schematic feature changes. To realize this, use the following command:

```
dc_shell> compile -scan -ungroup_all
```

9. For a design to be complete, it needs to be connected to all of the library components and designs it references. So to perform a name-based resolution of design references for the current design, use the link command. The references must be located and

linked to the current design in order for the design to be functional. The purpose of this command is to locate all of the designs and library components referenced in the current design and connect (link) them to the current design.

```
dc_shell> link
```

10. To read a test protocol file into memory.

```
dc_shell> read_test_protocol ../source/Johnson_count_test.spf
```

The output is:

```
Warning: a protocol already exists in mode 'all_dft' and will be
 ↪  deleted. (TEST-1401)
... deleting protocol from the mode 'all_dft' ...

... Reading /home/users/<username>/synopsys/syn_tut/dc_90/source/Jo⌋
 ↪  hnson_count_test.spf
 ↪  ...
... STIL version 1.0 ( Design 2005 CTL P2001.10) ...
... SignalGroups  ...
... Timing  ...
... Procedures all_dft_protocol: "multiclock_capture"
 ↪  "allclock_capture" "allclock_launch" "allclock_launch_capture"
 ↪  ...
... MacroDefs all_dft_protocol: "test_setup" ...
Warning: Reading in a test protocol after specifying signals will
 ↪  cause the signals to be cleared from the protocol and the
 ↪  model.  (TEST-1714)
Read protocol for mode all_dft
1
```

11. Use -view spec in order to define the scan structure.

```
set_dft_signal -view spec -port SCANINPORT -type ScanDataIn
set_dft_signal -view spec -port SCANOUTPORT -type ScanDataOut
set_dft_signal -view exist -port SE -type ScanEnable
```

12. Set the DFT insertion configuration for the current design.

```
dc_shell> set_dft_insertion_configuration -preserve_design_name true
```

`-preserve_design_name true` specifies whether to preserve the design name when writing from the tool to the database during DFT insertion. Valid values are true to preserve the design name and false to permit renaming the design. The default value is false.

13. To specify the scan chain design.

```
dc_shell> set_scan_configuration -chain_count 1
```

`-chain_count 1` specifies a positive integer for the number of chains that insert_dft is to build. If not specified, insert_dft builds the minimum number of scan chains consistent with clock mixing constraints.

14. Before performing scan insertion, you can preview the scan design by specifying the `preview_dft` command. This command generates a scan chain that satisfies the scan specifications on your current design and displays the scan chain design. Previews, but does not implement, the test points, scan chains, and on-chip clocking control logic to be added to the current design.

    ```
    dc_shell> preview_dft
    ```

15. After previewing the design, it is ready to assemble the scan chains by using `insert_dft` command. It adds scan circuitry (either internal scan or boundary-scan) to the current design.

    ```
    dc_shell> insert_dft
    ```

16. Set the `fix_multiple_port_nets` attribute to a specified value on the current design. To buffer logic constants, use the `-buffer_constants` option with the `-all` option. `-buffer_constants` option buffers logic constants instead of duplicating them.

    ```
    set_fix_multiple_port_nets -all -buffer_constants
    compile -scan -incremental
    ```

17. Run design rule checking again and display area information for the current design.

    ```
    dft_drc -coverage_estimate
    report_area
    ```

    `-coverage_estimate` generates test coverage estimate at the end of design rule checking.

18. Write a design from memory to a file with format .sdc (Synopsys internal database format), also format .v and .vhdl.

    ```
    dc_shell> write -format verilog -h -o ../results/Johnson_count_dc.v
    ```

19. Write a design with Standard Delay Format (SDF).

    ```
    dc_shell> write_sdf ../results/Johnson_count_dc.sdf
    ```

20. To exit DFT Compiler write exit in the command line.

    ```
    dc_shell> exit
    ```

6 Getting Started with OpenLane and SkyWater PDK

6.1 Introduction

The OpenROAD Project is founded by the University of California, San Diego VLSI CAD Laboratory, cooperated with the University of Michigan, ARM, University of Minnesota, Arizona State University, etc. The main product of the OpenROAD project is the OpenROAD App. OpenROAD is a foundational building block in open-source digital flows like OpenROAD-flow-scripts, OpenLane from Efabless, Silicon Compiler Systems, as well as OpenFASoC for mixed-signal design flows. The OpenROAD project supports two main flow controllers:

- OpenROAD-flow-scripts - Supported by the OpenROAD project
- OpenLane - Supported by Efabless

In this lab, we will mainly focus on OpenLane.

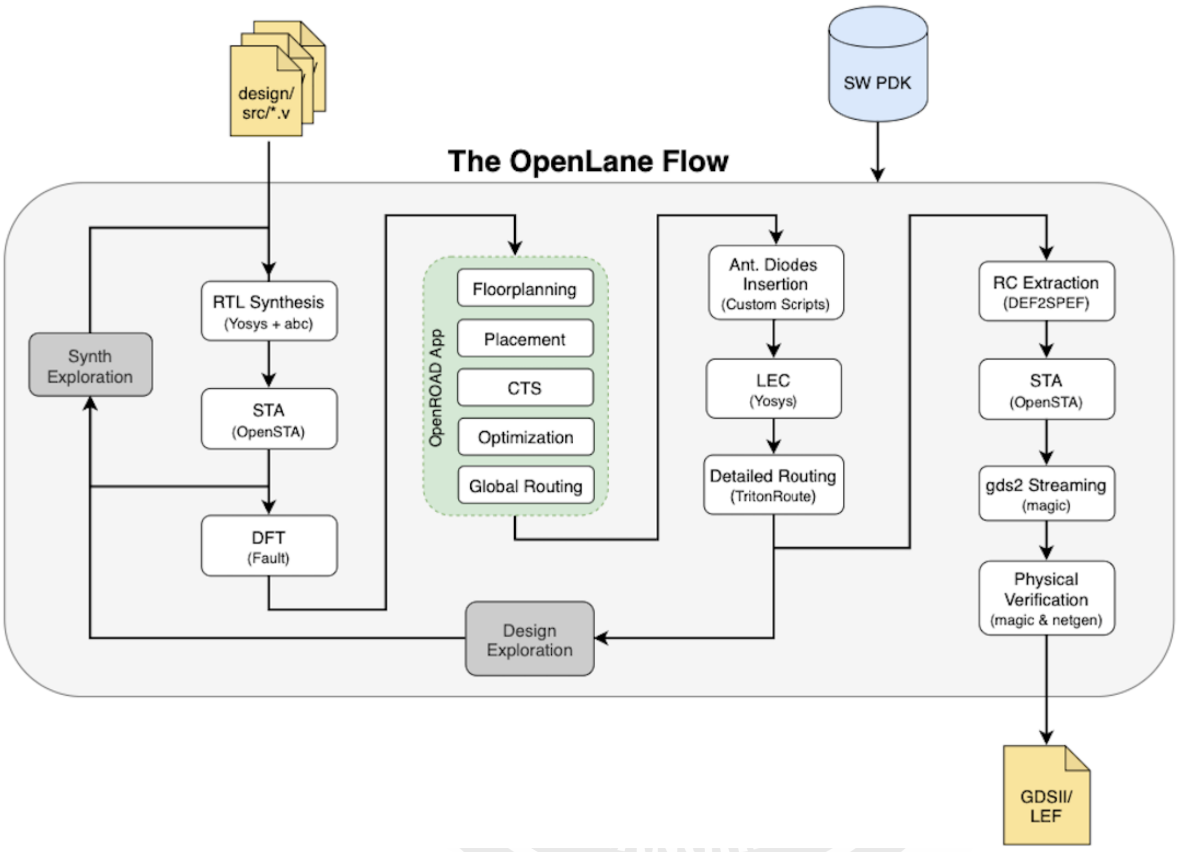


Figure 30: OpenLane Architecture.

OpenLane is an automated RTL to GDSII flow based on several components, including OpenROAD, Yosys, Magic, Netgen, CVC, SPEF-Extractor, KLayout, and a number of custom scripts for design exploration and optimization. The flow performs all ASIC implementation steps from RTL all the way down to GDSII.

The SkyWater Open Source PDK is a collaboration between Google and SkyWater Technology Foundry to provide a fully open-source Process Design Kit and related resources, which can be used to create manufacturable designs at SkyWater's facility.

There are a few introductory presentations made by various key contributors to help you understand the project [9]:

- [FOSSi Dial-Up] Tim Ansell - Skywater PDK: Fully open source manufacturable PDK for a 130nm process
- [FOSSi Dial-Up] Mohamed Shalan - OpenLane, A Digital ASIC Flow for SkyWater 130nm Open PDK
- [FOSSi Dial-Up] Mohamed Kassem - The striVe RISC-V SoC Family on SkyWater 130nm
- [FOSSi Dial-Up] James Stine - Designing new 130nm cells for SkyWater 130nm

The repository provides five sky130 standard cell libraries:

- sky130_fd_sc_hd
- sky130_fd_sc_hs
- sky130_fd_sc_ms
- sky130_fd_sc_ls
- sky130_fd_sc_hdll

6.2 Setting up the OpenLane flow

Please follow the official documentation `https://openlane.readthedocs.io/en/latest/getting_started/installation.html`. For Windows users, deploying OpenLane on WSL2 is recommended: `https://docs.docker.com/desktop/windows/wsl/` `https://learn.microsoft.com/en-us/windows/wsl/tutorials/wsl-containers` because Windows can dynamically allocate memory for WSL. Some operations need a huge amount of memory, so you can automatically make full use of your memory during the process without many trials and errors. If using virtual machines, please allocate enough memory (at least 4 GiB) for your virtual machines. If you failed to `make` or start docker "docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?.", please start Docker Desktop for Windows first. If there are errors stopping `make`

```
requests.exceptions.ConnectionError: ('Connection aborted.',
↪  RemoteDisconnected('Remote end closed connection without response'))
```

Please read the Troubleshooting B.4. After `make`, the sky130A and sky130B PDKs are prepared under the `pdks/volare` directory. Since OpenLane and SkyWater SKY130 PDK are under active development and the latter one is still an experimental preview or alpha release, there may be versioning issues. Even worse, the developers of OpenLane and SkyWater SKY130 PDK have very bad habits of using Git - they push all commits to the master/main branch every time without previously testing them rather than separate several branches or at least maintain a "develop" branch. Even worse, they instruct users to always start from the master/main branch rather than a tag or a release. The version of OpenLane that is tested by the TA is the commit 32da932761213af689f10088d671e1e3dc38f273 (tag: 2022.10.13). The Sky130 PDK version is de752ec0ba4da0ecb1fbcd309eeec4993d88f5bc. The OpenLane Docker image is tag 32da932761213af689f10088d671e1e3dc38f273-amd64 and the corresponding digest is sha256:92390c610a2204df6551b6cf3f9226e59f30ab17966b0677cd756cda74cb8de6. Unfortunately, during testing, the TA found problems with this version and created pull requests, so this part was cut; just use the latest version. During `make test`, please note that if you have already set up X11 for WSL 2, it will show:

```
cd /mnt/d/Resources/OpenLane && \
        docker run --rm -v /mnt/d/Resources/OpenLane:/openlane -v
        ↪   /mnt/d/Resources/OpenLane/designs:/openlane/install -v
        ↪   /mnt/d/Resources/OpenLane/pdks:/mnt/d/Resources/OpenLane/pdks
        ↪   -e PDK_ROOT=/mnt/d/Resources/OpenLane/pdks -e PDK=sky130A
        ↪   --user 1000:1000 -e DISPLAY=:0 -v
        ↪   /tmp/.X11-unix:/tmp/.X11-unix -v
        ↪   /home/<username>/.Xauthority:/.Xauthority --network host
        ↪   efabless/openlane:32da932761213af689f10088d671e1e3dc38f273-amd64
        ↪   sh -c "./flow.tcl -design spm -tag openlane_test -overwrite"
```

After that, if you run `klayout`, it can open the KLayout window. If not, please set up X11 for WSL 2 first [2].

6.3 Adding your designs

Follow Adding your designs to create a new design `lab4` and copy files in `lab4_starter` under `lab4/src`. You are allowed the modify the source code if needed, as long as it is different from the existing designs in the `design` directory of OpenLane. Using a Tcl configuration file is recommended rather than using a JSON configuration file because the TA has found bugs related to JSON configuration files. You may need to prepare an SDC file under the `src` directory and configure it in `config.tcl` or `config.json`.

Once the run is complete, you should see a new directory in `${OPENLANE_ROOT}/designs/lab4/runs/`. It'll be named after the UTC time pertaining to the run. Call that subdirectory LATEST_RUN and descend into it. You should see logs, reports, results, tmp. If you see errors.log or errors.txt, that is bad.

6.4 (Optional) Regression & Exploration

Follow Regression & Exploration. Define a "regression" configuration file in `${OPENLANE_ROOT}/designs/lab4/regression.config` like:

```
CLOCK_PERIOD=(1 2 5 10)
GLB_RT_ADJUSTMENT=(0 0.15)
FP_CORE_UTIL=(10 25 50)
```

```
PL_TARGET_DENSITY=(0.05 0.1 0.2)
SYNTH_STRATEGY=(1 3 14)

extra="
"
```

There is some documentation for these parameters: https://github.com/efabless/openlane/blob/master/configuration/README.md now move to https://openlane.readthedocs.io/en/latest/reference/configuration.html. Set up the regression config and run the regression tool to find the tightest clock timing for the design.

6.5 (Optional) Tasks

1. Please modify the SDC file and create a case where setup time did not pass, then use the critical as an example and describe how you can fix the timing for that path. Please upload the full timing path and also your answers for fixing the timing violation.
2. Please modify the SDC file and find the maximum frequency that passes the timing check after routing. Please upload the screenshot of the layout after place and route, and report area and power.

7 Deliverables

Please add comments or explanations to all the deliverables.

1. Section 4.2: the output of the `vcs` command.
2. Section 4.4: the screenshot of the TopLevel window with tracing drivers and loads.
3. Section 4.5: the screenshot of the Compared Signal Groups in the Wave View.
4. Section 4.6 (Optional): the screenshot of the Highlighted Signal Groups in the Wave View.
5. Section 4.7: the exported user-defined radix TXT file and the screenshot of the Edit User-defined Radix dialog box with Wave view window.
6. Section 5.1.3: the three reports of the area, constraints, and timing in Step 10 and `Johnson_count_dcl.v` in Step 11.
7. Section 5.2.3: the Preview_dft report in Step 14, the output of the `compile` command in Step 16, the area report in Step 17, `Johnson_count_dc.v` in Step 18, and `Johnson_count_dc.sdf` in Step 19.
8. Section 6.3: the screenshot of the output of the run.
9. Section 6.4 (Optional): the `regression.config` file and the log file and CSV files in the `regression_results` directory.
10. Section 6.5 (Optional): the SDC file, the screenshot of the full timing path, the answers for Task 1; the SDC file, the screenshot of the layout after place and route, and the area and power reports for Task 2.

8 Grading policy

Factors	Percentage
Section 4	35% + 2% bonus
Section 5	35%
Section 6	30% + 18% bonus

A Peer Evaluation Form

Part	Your work	Your partner's work	Your score	Your partner's score
Section 4				
Section 5				
Section 6				

B Troubleshooting

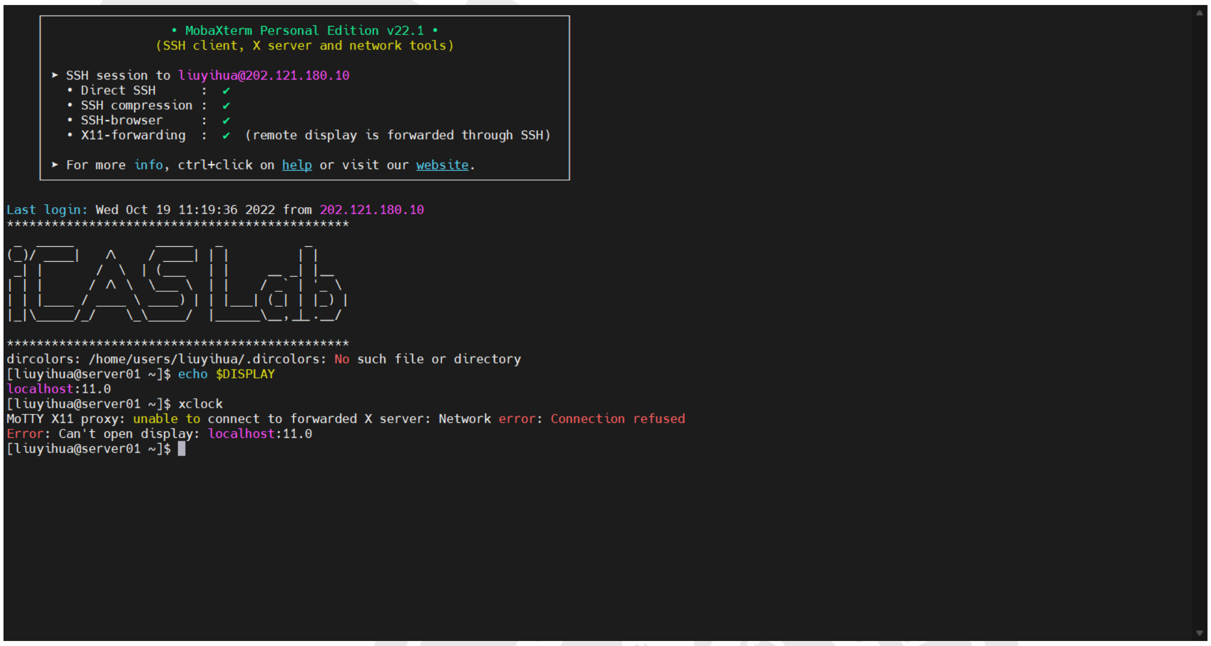
B.1 VcXsrv XLaunch not working

First, make sure there are no other X servers running. Then, please read the reference article [3].

B.2 Other X server problems

Please read the reference article [13].

B.3 MobaXterm X server



```
                • MobaXterm Personal Edition v22.1 •
              (SSH client, X server and network tools)

 ➤ SSH session to liuyihua@202.121.180.10
   • Direct SSH      :  ✔
   • SSH compression :  ✔
   • SSH-browser     :  ✔
   • X11-forwarding  :  ✔  (remote display is forwarded through SSH)

 ➤ For more info, ctrl+click on help or visit our website.

Last login: Wed Oct 19 11:19:36 2022 from 202.121.180.10
**********************************************
 _  _____          _____ _           _
(_)/ ____|   /\   / ____| |         | |
 _| |       /  \ | (___ | |     __ _| |__
| | |      / /\ \ \___ \| |    / _` | '_ \
| | |____ / ____ \ ____) | |___| (_| | |_) |
|_|\_____/_/    \_\_____/|______\__,_|_.__/

**********************************************
dircolors: /home/users/liuyihua/.dircolors: No such file or directory
[liuyihua@server01 ~]$ echo $DISPLAY
localhost:11.0
[liuyihua@server01 ~]$ xclock
MoTTY X11 proxy: unable to connect to forwarded X server: Network error: Connection refused
Error: Can't open display: localhost:11.0
[liuyihua@server01 ~]$ 
```

Figure 31: MobaXterm X11 error.

Close all your proxy or VPN connections and restart your computer. Remember that avoid opening any proxy or VPN connections before your launch MobaXterm [8].

B.4 OpenLane Make Volare Connection Error

Please read the reference [4]. Please just use the SJTU educational network during the setup. If you use other proxy or VPN networks, there are probably problems.

C Change Log

Fall 2022: Yihua Liu

- merged, simplified, and updated Lab 6 of Fall 2021 as Section 6

- created this lab

Fall 2021: Prof. Xinfei Guo

- created Lab 6

References

[1] Stuart Adamson et al. *XWIN*. Dec. 12, 2021. URL: https://x.cygwin.com/docs/man1/XWin.1.html.

[2] Microsoft Corporation. *Run Linux GUI apps on the Windows Subsystem for Linux*. Aug. 13, 2022. URL: https://learn.microsoft.com/en-us/windows/wsl/tutorials/gui-apps.

[3] Yihua Liu. *VcXsrv XLaunch failed to bind listener*. Oct. 18, 2022. URL: https://blog.csdn.net/yihuajack/article/details/127378884.

[4] Yihua Liu and Mohamed Gaber. *volare enable remotedisconnected*. Oct. 23, 2022. URL: https://github.com/efabless/volare/issues/26.

[5] Microsoft. *Get started with OpenSSH for Windows*. June 8, 2022. URL: https://learn.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse.

[6] Microsoft. *Support X11 Forwarding*. Dec. 7, 2019. URL: https://github.com/PowerShell/Win32-OpenSSH/issues/1515.

[7] Mobatek. *MobaXterm documentation*. 2022. URL: https://mobaxterm.mobatek.net/documentation.html.

[8] Pavlin. *Troubleshoot your MobaXterm X11 connection to UPPMAX*. Aug. 2022. URL: https://hackmd.io/@pmitev/UPPMAX-MobaXterm-X11.

[9] Arya Reais-Parsi. *Lab 1 - Intro to OpenLane and Skywater 130*. 2018. URL: https://inst.eecs.berkeley.edu/~cs250/fa20/labs/lab1/.

[10] Spuratic and einpoklum. *Getting "Warning: Missing charsets in String to FontSet conversion"*. Oct. 21, 2020. URL: https://superuser.com/questions/1531413/getting-warning-missing-charsets-in-string-to-fontset-conversion.

[11] Tatu Ylonen et al. *ssh(1) - Linux man page*. Apr. 14, 2013. URL: https://linux.die.net/man/1/ssh.

[12] Tatu Ylonen et al. *ssh_config(5) - Linux man page*. Apr. 14, 2013. URL: https://linux.die.net/man/5/ssh_config.

[13] Steven Zeil. *Troubleshooting X*. Aug. 2, 2018. URL: https://www.cs.odu.edu/~zeil/cs252/latest/Public/xtrouble/index.html.