X-FAB Semiconductor Foundries AG
Haarbergstraße 67
D-99097 Erfurt
Germany
phone +49 361-427-6663
fax +49 361-427-6631

XENV - X-FAB Design Kit Scripts Package for IC6.1

User Guide

Release V1.2.10

05 Dec 2014

REVISION HISTORY

NUMBER	DATE	DESCRIPTION
V1.2.10	05 Dec 2014	• cds.lib generation modified to support the correct IP library selection with older modular processes (e.g. xc06m3) • avoid unnecessary updates of pvsSetup and calibreSetup • pvs setup – temporary preset file defined with explicit path definition "./" – The implicit path definition which was used until version 1.2.9 caused issues with the pre-trigger loading, if the local skill path was prepended by non-default directory • QRC default setup: align default temperature with spectre (27degrC)
V1.2.9	29 Sept 2014	• support for Cadence EAD • IC616 compatibility • option --getversion with -t will create the setup file only for the chosen process (plus x_all, xx) • update CDL netlisting routine for XU035 UHV devices
V1.2.8	25 July 2014	minor bugfix for --clone option
V1.2.7	19 June 2014	minor bugfix for wrong call of $CUSTOM_SCRIPT_PATH script from virtuoso
V1.2.6	03 June 2014	minor bugfix for module update with .assura-, .calibre- and .pvs-Setup directories
V1.2.5	29 May 2014	support for packages which are valid for multiple process families
V1.2.4	22 April 2014	support for variables in pvtech.lib
V1.2.3	28 March 2014	enhanced PVS setup for modular processes
V1.2.2	14 February 2014	minor bugfix for cds.lib generation

REVISION HISTORY

NUMBER	DATE	DESCRIPTION
V1.2.1	31.01.2014	• more customization with additional .xfabcadrc variables. – default directory to update .cadence can now be controlled by .xfabcadrc variable XfabUpdateDotCadenceInHomeDir. – relative paths in phyiscal verification setup can be replaced by absolute paths, using .xfabcadrc variable XfabNoRelPathsInSetup • enviroment setup is split into two parts: required PDK-specific setup (1) and recommended EDA tool defaults setup (2). – (2) can be suppressed by the SKILL variable XfabCadNoToolDefaults • enhanced -c|--clone – clones PVS- and Calibre setup – works with non-modular PDK, e.g. xc06 • support for Coventor MEMS+ integration • improved spectre model setup when using --tdir option
V1.1.2	01 November 2013	default linking behaviour of xkit uses all existing major releases, similar to --allversion
V1.1.1	03 September 2013	add new option --allversions option --useversion now supports multiple version numbers improved version sorting algorithm, supports multiple digit numbers
V1.1.0	15 August 2013	change only release number - use one digit numbers
V1.0.10	01 August 2013	minor bugfix for spectre ModelPath setup
V1.0.9	11 June 2013	add new option --useversion, to be able to switch between major package versions
V1.0.8	28 May 2013	• minor bugfixes, and: – prevent error popup in vuiRcxRunForm – solve handling if setup is uncomplete • enhancements: – handling trailing slash from X_DIR – handling of symLinks in xkit – search defined technology data only, except with --getversion – improve error handling if technology does not exist, print in X_DIR available technologies

REVISION HISTORY

NUMBER	DATE	DESCRIPTION
V1.0.7	15 May 2013	• enhanced xStream setup • enhance modelsetup – full support of non-modular PDKs (XI10 etc) – param.scs at first, all others sorted alphabetically – $tech.scs inactive as last with section mc_g, for a better support of MC simulation
V1.0.6	22 April 2013	enhanced support for PVS / iPVS
V1.0.5	10 April 2013	minor bug fixes
V1.0.4	01 March 2013	more customization with xfabcadrc
V1.0.3	21 February 2013	full support for XENV-compliant Calibre runsets
V1.0.2	31 January 2013	enhanced support for 3rd party EDA tools enhanced customization with .xfabcadrc new option -q|--quiet cdsNetlistingMode set to "Analog" for GUI-based flows library SCHAFFNER is now linked in .xkit and defined in cds.lib
V1.0.1	20 December 2012	Initial release of XKIT package

Contents

List of Figures

1 Introduction

This document describes the scope and content of XENV, X-FAB's design kit scripts package for Cadence IC6.1, and the installation and usage of XENV-compliant design kit packages.

1.1 X-FAB Design Kits

X-FAB design kits close the gap between EDA software and foundry process. The different design kit packages contain all data to work with the digital, analog and mixed-signal design-flows in combination with X-FAB foundry data. Together with the XENV design kit scripts package, the design kits provide an out-of-the-box support of the latest design flows, and enable designers to start IC development without any additional settings.

1.2 XENV

XENV design kit scripts package provides an intuitive, easy to use and proven way to setup a project directory for Cadence IC6.1 using X-FAB's design kit packages. The package contains all necessary scripts and template files to create and update the project setup in a structured, reliable and reproducible way. While XENV basically supports Cadence design flows, it provides an easy way to integrate 3rd party OpenAccess EDA tools in the design flow.

The most important feature of XENV is the improved version handling:

- XENV offers support of multiple library or package versions within one root directory.
  - XENV chooses the latest available versions during the project setup (default),
  - and changes these versions only if the project admin decides to do an update (no unexpected library updates).
- XENV supports a convenient selection of library versions, which can be used to rebuild a project directory with the original library versions

2 Requirements

XENV and all XENV-compliant design kit packages have been created and verified on the following Linux RedHat version:

RHEL 5.5

Other versions of Linux RedHat or other Linux distributions might also be used with XENV but have not been verified by X-FAB.

XENV requires the following software versions:

- perl 5.8.5 or higher
- rsync 2.6 or higher
- GNU find 4.4.0 or higher
- Cadence IC6.1.5

3 Installation

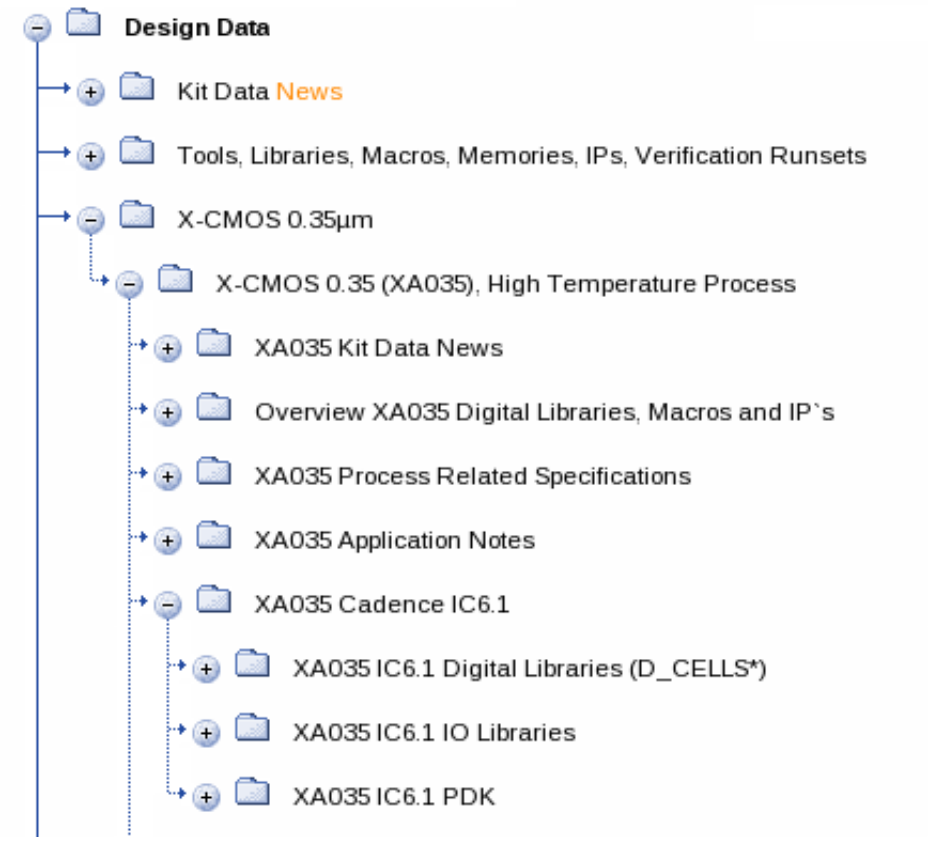
3.1 Download from X-TIC

XENV requires at least the installation of the following design kit packages:

- XENV design kit scripts package (process independent)
- process-specific PDK Kernel package
- process-specific Spectre Models package
- process-specific Verification Runset package(s)

Depending on the target design and the required design flows, it will be necessary to install additional packages (e.g. analog-, digital- , IO-libraries).

Figure 1 XENV-compliant design kit packages on X-TIC



Each design kit package tar file contains the full path, starting below the installation root directory. The tar file name clearly describes the component content, and is aligned with the contained directory structure.

Example: XA035 Spectre Models package, version 4.0.1

```
tar file name:
    xa035-cadence-spectre-v4_0_1.tar.gz
tar file content:
    xa035/cadence/v4_0/spectre/v4_0_1/mc_params
    xa035/cadence/v4_0/spectre/v4_0_1/mos
    xa035/cadence/v4_0/spectre/v4_0_1/occ
    xa035/cadence/v4_0/spectre/v4_0_1/revision_v4_0_1.info
```

Note
XENV is backward-compatible to tkit-compliant PDK- and library packages. For more information, please refer to Section 4.1.12

3.2 Package Installation

All design kit packages need to be installed at a common root directory <XKIT_ROOT_DIR>. New versions of already installed packages should be extracted at the same root directory and will be installed in parallel to the existing data. The following steps describe the installation of the minimum set of design kit packages, which is necessary to support analog-centric design flows. All other packages can be installed in this way.

3.2.1 Create XKIT root directory (only necessary for first installation)

For the first installation it will be necessary to create a new root directory. This directory should be used for all further package installations.

```
> mkdir <XKIT_ROOT_DIR>
> cd <XKIT_ROOT_DIR>
```

Example:

```
> mkdir /xfab/XKIT
> cd /xfab/XKIT
```

3.2.2 Install XENV, X-FAB design kit scripts package for Cadence IC6.1

At <XKIT_ROOT_DIR>, extract the XENV package tar file using the following command:

```
> tar -xzvf xenv-cadence-v1_0_3.tar.gz
```

The following directory structure will be created:

```
<XKIT_ROOT_DIR>/x_all/cadence/xenv/v1_0_3
```

3.2.3 Installation of design kit packages

All other design kit packages can be installed in a similar way:

```
> cd <XKIT_ROOT_DIR>
> tar -xzvf xa035-cadence-PDK-IC61-v4_0_1.tar.gz
> tar -xzvf xa035-cadence-assura-v4_0_1.tar.gz
> tar -xzvf xa035-cadence-QRC_assura-v4_0_1.tar.gz
> tar -xzvf xa035-cadence-spectre-v4_0_1.tar.gz
```

The directory structure at <XKIT_ROOT_DIR> should now look like this:

```
$X_DIR
   |- x_all/cadence/xenv/v1_0_3
   |- xa035/cadence
              |- v4_0
                   |- assura
                   |    |- v4_0_1
                   |- PDK
```

```
                    |     |- v4_0_1
                    |- QRC_assura
                    |     |- v4_0_1
                    |- spectre
                          |- v4_0_1
```

3.2.4 Installation of new versions of existing design kit packages

New versions of existing design kit packages should be installed at the same root path <XKIT_ROOT_DIR>.

Example: Installation of new PDK kernel version

```
> cd <XKIT_ROOT_DIR>
> tar -xzvf xa035-cadence-PDK-IC61-v4_0_2.tar.gz
```

Version 4.0.2 of the PDK kernel will be installed in parallel to version 4.0.1:

```
$X_DIR
   |- x_all/cadence/xenv/v1_0_3
   |- xa035/cadence
               |- v4_0
                    |- assura
                    |     |- v4_0_1
                    |- PDK
                    |     |- v4_0_1
                    |     |- v4_0_2
                    |- QRC_assura
                    |     |- v4_0_1
                    |- spectre
                          |- v4_0_1
```

3.2.5 Installation of multiple design kit packages

Multiple design kit packages can be installed simultaneously by using tar with a list of available tar files. The following example will extract all tar files which are found in /downloads/xfab:

```
> cd <XKIT_ROOT_DIR>
> echo  /downloads/xfab/*.tar.gz | xargs -n 1 tar -zxvf
```

3.3 Environment Settings

XENV requires the definition of just one environment variable: X_DIR. $X_DIR should point to the root path <XKIT_ROOT_DIR>.

It is recommended to define X_DIR in the local shell rc or module file. The path to XENV should be added to the local path variable.

Example (c-shell): X_DIR definition in .cshrc

In this example, /opt/xfab was used as <XKIT_ROOT_DIR>

```
setenv X_DIR /opt/xfab
set path = ($X_DIR/x_all/cadence/xenv $path)
```

Example (bash): X_DIR definition in .bashrc

```
export X_DIR=/opt/xfab
export PATH=$X_DIR/x_all/cadence/xenv:$PATH
```

After these definitions, the main executable of XENV, xkit, should be found in the path.

The Cadence-specific environment variables $CDSHOME and $IUSHOME should be defined and should point the valid CDS or IUS installation.

The Cadence netlisting mode needs to be defined as "Analog". It is recommended to define the netlisting mode as a shell variable rather than with Skill, to support GUI- and non-GUI-based design flows.

Example (c-shell):

```
setenv CDS_Netlisting_Mode Analog
```

Example (bash):

```
export CDS_Netlisting_Mode=Analog
```

4 Project setup with XENV xkit

The xenv scripts at $X_DIR/x_all/cadence/xenv create a setup for Cadence Virtuoso in the working directory, and define defaults for the most important Cadence tools. After the definition of X_DIR as described in Section 3.3, the main executable of XENV, xkit, should point to $X_DIR/x_all/cadence/xenv/xkit.

For a new project, it is recommended to execute xkit in an empty directory. By default, xkit will start Cadence virtuoso after the directory setup is finished.

4.1 xkit options

The current version of xkit supports the following command line options:

4.1.1 xkit -h

prints xkit help message including the supported command line options.

```
xkit - X-FAB Cadence PDK setup script

Usage: xkit [OPTION]...
Setup or update of a project directory for Cadence IC6.1 using X-FAB's design kit ↩
     packages

Mandatory arguments to long options are mandatory for short options too.

   -h|--help                      : this (help) message
   -v|--version                   : print xenv / xkit package version
   -t|--te|--tech technology      : specify X-FAB technology to use
   -m|--modules [module_code]     : PDK module update, use optional PDK
                                    module code if defined
   -u|--up|--update               : library version update
   -s|--setversion versionsfile   : library version update to the versions  ↩
       defined in versionsfile
   --getversion                   : only create versionsfile which can be used as ↩
        input for -s option
   -c|--clone master              : clone project setup from master directory
```

```
   -n|--norun                      : no start executable, only update init files
   -q|--quiet                      : suppress non-error messages
   --tdir                          : also include tkit-compliant PDK packages  ↩
       installed at $T_DIR
   --in|--internal                 : use internal DRC runset

Examples:
         xkit -t xp018 -n
         use technology xp018, starts xp018 module query loop, no start of  ↩
             virtuoso

         xkit -t xa035 --modules 1021
         use technology xa035, use module code 1021, start of virtuoso
```

4.1.2 xkit -t <technology>

xkit -t specifies the X-FAB technology to be used. Executed in an empty directory, it creates the following files and directories:

```
.assuraSetup            --> directory containing assura setup files
.avviewinit
.cdsenv
.cdsinit                --> Cadence initialisation script
.cdsinit_personal
.simrc
.xkit                   --> directory containing linked design kit structure
assura_tech.lib
cds.lib                 --> Cadence library definitions file
proj_opt.txt            --> X-FAB project options file
xa035.inc
xa035.lib
```

Except cds.lib and .cdsinit_personal, these files should not be edited manually.

xkit -t parses the installed design kit packages at $X_DIR, checks if the selected X-FAB technology exists and selects the latest version of each package. This latest version is used to create the link structure at .xkit. All setup files in $PWD reference to the linked structure.

Example: .xkit directory created with "xkit -t xa035"

```
.xkit
 |- 20120925_152453
 |- 20120925_152453.setup
 |- setup -> 20120925_152453
     |- x_all
     |- xa035
        |- analoglibs
        |- cadence
        |    |- PDK -> /xfab/XKIT/xa035/cadence/v4_0/PDK/IC61/v4_0_1
        |    |- QRC_assura -> /xfab/XKIT/xa035/cadence/v4_0/QRC_assura/v4_0_1
        |    |- assura -> /xfab/XKIT/xa035/cadence/v4_0/assura/v4_0_1
        |    |- spectre -> /xfab/XKIT/xa035/cadence/v4_0/spectre/v4_0_1
        |- diglibs.
```

xkit -t creates a timestamp directory at .xkit, and creates or updates the setup link according to the new timestamp directory. It also creates the setup log file <TIMESTAMP>.setup, which lists the used package versions.

If xkit is called in an existing project directory, the script compares the library versions defined by the existing .xkit structure with the latest versions available at $X_DIR. xkit prints the following warning if the local .xkit structure links to out-dated libraries:

```
The local PDK tree links to an outdated version of package "xa035 PDK": 4.0.1
Please consider to run
  xkit -u
  to update package "xa035 PDK" to the latest installed version 4.0.2.
```

4.1.3 xkit -u

xkit -u forces an update of the local .xkit structure. xkit -u creates a new timestamp directory at .xkit, and updates the setup link to the new timestamp directory. Existing timestamp directories will not be deleted, and can be used to easily switch back to an older setup.

Example: CAD support installed version xa035 PDK version 4.0.2. xkit informed about the available new version and suggests to update the local .xkit directory. The project admin decides to do so, using xkit -u. After the update, the local .xkit directory would look like this:

```
.xkit
 |- 20120925_152453
 |- 20120925_152453.setup
 |- 20121108_093811
 |- 20121108_093811.setup
 |- setup -> 20121108_093811
     |- x_all
     |- xa035
        |- analoglibs
        |- cadence
        |    |- PDK -> /xfab/XKIT/xa035/cadence/v4_0/PDK/IC61/v4_0_2
        |    |- QRC_assura -> /xfab/XKIT/xa035/cadence/v4_0/QRC_assura/v4_0_1
        |    |- assura -> /xfab/XKIT/xa035/cadence/v4_0/assura/v4_0_1
        |    |- spectre -> /xfab/XKIT/xa035/cadence/v4_0/spectre/v4_0_1
        |- diglibs.
```

xkit created the new timestamp directory 20121108_093811, which links to xa035 PDK v4.0.2. The setup link has been updated and points now to the new timestamp directory.

xkit does not remove or replace existing timestamp directories or log files. This makes it very easy to switch back to an old setup, either by manually changing the setup link, or by using one of the log files as an input for xkit -s.

xkit -u will update all out-dated packages. If it is necessary to update only a specific set of packages and keep the other packages unchanged, xkit -s should be used.

xkit -u will only trigger the PDK module selection (-m option), if the existing module selection does not longer exist after the PDK version update.

4.1.4 xkit -s <versionsfile>

xkit -s creates a new timestamp directory at .xkit, using the library versions defined in <versionsfile>. xkit -s accepts the log files from xkit as input, which allows to re-create a library setup with the defined library versions.

xkit -s can also be used to create a setup with a specific set of package versions.in this case, the package versions need to be defined in the <versionsfile>.

4.1.5 xkit --getversion

xkit --getversion only creates a version file, which can later be used as an input for xkit -s.

--getversion can be used without --tech option. In this case, the version file will include all packages of all process families installed at $X_DIR. Used with --tech, it will only inlcude the chosen technology and x_all.

--getversion can be combined with option --tdir, to include the package found at $T_DIR.

4.1.6 xkit --useversion <version> [--only]

xkit --useversion forces the usage of the defined version number for PDK packages. Packages which exist under this 2-digits version number are used for the setup. Packages which do not exist under the given version number are linked from the latest version number. The option --only might be added if only packages from the given version number should be used, even if some packages do not exist.

--useversion accepts multiple version numbers as input. Multiple version numbers need to be enclosed in single- or double quotes, to be transfered as a single parameter. xkit uses the given version numbers for package version selection, starting with the first given version number.

Examples:

```
xkit -t xh035 --useversion 5.0
```

will link all PDK packages which exist for the major version 5.0. All remaining packages are linked from the latest major release.

```
xkit -t xh035 --useversion 5.0 --only
```

will link all PDK packages which exist for the major version 5.0. Packages from the latest release are ignored.

```
xkit -t xh035 --useversion "5.0 6.0"
```

will link all PDK packages which exist for the major version 5.0. All remaining packages are linked from major version 6.0.

4.1.7 xkit --allversions

xkit --allversions uses all existing major versions for the linking. The major versions are sorted, and the linking starts with the latest version. --allversions can be comined with --useversion. In this case, the linking starts with the versions defined by --useversion.

Starting with XENV V1.1.2, the default linking behaviour of xkit uses all existing major releases, similar to --allversion being set explicitly.The linking can be restricted to the latest major release by setting the option --only.

4.1.8 xkit -m [<module_code>]

xkit -m forces an update of the chosen PDK modules. This option is only supported by modular PDKs.

Example: PDK module selection loop for xa035

```
You are now prompted to select the main modules of XA035 process.

This selection defines sets of devices which can be used together in one
design without design rule violations. The specific process variant and
the number of necessary mask layers depend on the devices which are used
in the design.

Core module is 1 - MOS (Standard 3.3V)
```

```
Please select MOS5A / HVMOSMID module.
0 - no MOS5A / HVMOSMID
1 - MOS5A (5.0V MOS) or HVMOSMID (14nm HV MOS)
Current setting is 0. To keep current setting, please press ENTER!
:
```

xkit -m updates .cdsinit, .cdsenv, .simrc, .avviewinit, assura_tech.lib, pvtech.lib, xa035.lib, xa035.inc, proj_opt.txt and the files in .assuraSetup and .pvsSetup. It does not overwrite user-defined setting in cds.lib and .cdsinit_personal.

The optional <module_code> forces an update of the chosen PDK modules to the given module code. xkit checks if the module code is valid, and updates all setup files accordingly without using the selection loop. This option can be used to implement xkit in a completely script-based flow.

4.1.9 xkit -n

xkit -n blocks the executions of Cadence Virtuoso. This option can be combined with all other options except -h.

4.1.10 xkit -q

xkit -q suppresses all non-error messages of xkit (quiet mode). This option can be combined with all other options except -h.

4.1.11 xkit -c <master>

xkit -c creates a "clone" of an existing project directory. The setup files and the PDK link structure .xkit will be linked to the existing files and directories of the given master directory. Depending on the file permissions in the master directory, this option can be used to restrict project modifications to a defined user or group (e.g. project admin). It is recommended to use this option especially for large or distributed design teams, to make sure that all team members use the same set of libraries, the same library versions and the same process options.

Example workflow:

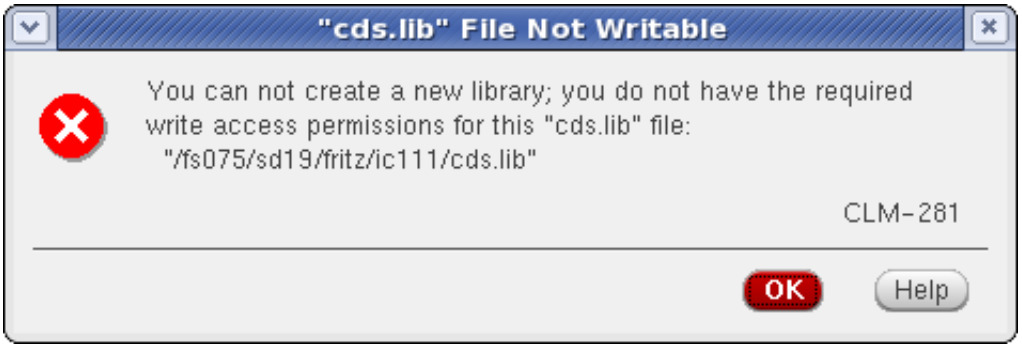
1. The project admin "Hans" creates the project setup at directory /projects/ic111 using "xkit -t xa035 -n". All files are created with permissions -rw-r--r--, all directories with drwxr-xr-x , depending on the project admin's umask settings. The project admin would also define the project-specific libraries in cds.lib, and create the project master library with write permissions for all group members. The project directory would look like this:

```
drwxr-xr-x 6 hans ic111 4096  .
drwxr-xr-x 3 hans ic111 4096  ..
drwxr-x--- 3 hans ic111 4096  .assuraSetup
-rw-r--r-- 1 hans ic111  102  .avviewinit
drwxr-xr-x 3 hans ic111 4096  .cadence
-rw-r--r-- 1 hans ic111  196  .cdsenv
-r--r--r-- 1 hans ic111 5222  .cdsinit
-rw-r--r-- 1 hans ic111 7142  .cdsinit_personal
-r--r--r-- 1 hans ic111 3546  .simrc
drwxr-x--- 4 hans ic111 4096  .xkit
drwxrwxr-x 3 hans ic111 4096  IC111
-rw-r--r-- 1 hans ic111   38  assura_tech.lib
-rw-rw-r-- 1 hans ic111  921  cds.lib
-rw-r--r-- 1 hans ic111  977  libManager.log
-r--r--r-- 1 hans ic111  216  proj_opt.txt
-rw-r--r-- 1 hans ic111    0  pvsUI_ipvs.log
-rw-r--r-- 1 hans ic111 1045  xa035.inc
-rw-r--r-- 1 hans ic111 2284  xa035.lib
```

2. Project member "Fritz" now clones the project setup using "xkit -c "/projects/ic111" . Fritz's project directory would look like this:

```
drwxr-sr-x  3 fritz ic111 4096  .
drwxr-sr-x 11 fritz ic111 4096  ..
lrwxrwxrwx  1 fritz ic111   70  .assuraSetup -> /projects/ic111/.assuraSetup
lrwxrwxrwx  1 fritz ic111   69  .avviewinit -> /projects/ic111/.avviewinit
lrwxrwxrwx  1 fritz ic111   65  .cdsenv -> /projects/ic111/.cdsenv
lrwxrwxrwx  1 fritz ic111   66  .cdsinit -> /projects/ic111/.cdsinit
lrwxrwxrwx  1 fritz ic111   75  .cdsinit_personal -> /projects/ic111/. ↩
    cdsinit_personal
lrwxrwxrwx  1 fritz ic111   64  .simrc -> /projects/ic111/.simrc
lrwxrwxrwx  1 fritz ic111   63  .xkit -> /projects/ic111/.xkit
lrwxrwxrwx  1 fritz ic111   73  assura_tech.lib -> /projects/ic111/ ↩
    assura_tech.lib
lrwxrwxrwx  1 fritz ic111   65  cds.lib -> /projects/ic111/cds.lib
lrwxrwxrwx  1 fritz ic111   70  proj_opt.txt -> /projects/ic111/proj_opt.txt
lrwxrwxrwx  1 fritz ic111   67  xa035.inc -> /projects/ic111/xa035.inc
lrwxrwxrwx  1 fritz ic111   67  xa035.lib -> /projects/ic111/xa035.lib
```

3. Fritz would be able to create new cells in the project libraries. He would not be able to create new libraries, to update the library versions or to change process options.



Note
Please note -c|--clone does not link .cadence from the master directory. It is recommended to use the .xfabcadrc option XfabUpdateDotCadenceInHomeDir = t to create the PDK-specific content of .cadence directory in $HOME, not in $CWD. Please refer to Section 4.3.3 for more information.

4.1.12 xkit --tdir

xkit --tdir runs xkit in compatibility mode with tkit-compliant PDK- and library packages.

tkit is the former version of X-FAB design kit scripts, which are now replaced by XENV/xkit. tkit offers no version handling, and tkit-compliant packages do not support multiple package versions within it's root directory $T_DIR.

With --tdir, xkit will also include the library packages found at $T_DIR. xkit will assign a dummy version number to these packages, and use its default mechanism to select the latest version number. As a result, packages which do not exist at $X_DIR will be linked from $T_DIR.

xkit will print a warning message for each library package linked from $T_DIR, if the corresponding, xkit-compliant PDK kernel exist:

```
WARNING: The local PDK tree links to the the unversioned package "xa035 D_CELLS  ↩
    cadence_IC61" at $T_DIR
         Please consider to install an xkit-compliant package version at $X_DIR.
```

In this case it is recommended to install also the xkit-compliant library packages at $X_DIR. After the next library version update with xkit -u, these libraries will be linked from $X_DIR.

4.2 xkit version sorting

The version numbers of XENV compliant design kit packages match the following pattern (perl regex syntax):

```
$version =~ /^[AB0-9]+_[AB0-9]+_[0-9]+(_[0-9]+)?$/
```

xkit uses a special sorting algorithm, which sorts the versions in the following order (oldest ... latest):

```
A.0.1 < B.0.1 < 1.B.1 < 2.A.1 < 1.0.1 < 1.0.1.1 < 1.0.9 < 1.0.10
```

4.3 customization

XENV provides several ways to customize project setup and EDA tool defaults.

4.3.1 .cdsinit_personal

xkit creates the file .cdsinit_personal in the working directory if no such file exists. xkit will not overwrite existing .cdsinit_personal files. .cdsinit_personal is loaded from .cdsinit.

Skill code defined in .cdsinit_personal will be loaded with each start of Cadence Virtuoso, after the default setup is done. By default, .cdsinit_personal contains only commented samples of skill code which a user might find helpful.

4.3.2 $CUSTOM_SCRIPT_PATH / $CUSTOM_SKILL_PATH

The shell variables $CUSTOM_SCRIPT_PATH and $CUSTOM_SKILL_PATH provide a more progressive way to customize the setup. If none of these variables is defined, the following info message is printed to CDS.log:

```
INFO: To customize this setup, please use the following environment variables:

    CUSTOM_SCRIPT_PATH - should point to the master shell or perl script
    e.g.:  setenv CUSTOM_SCRIPT_PATH /path/to/my/script/LoadMyScripts.csh

    CUSTOM_SKILL_PATH - should point to your master skill file
    e.g.:  setenv CUSTOM_SKILL_PATH /path/to/my/skill/LoadMySkills.il
```

The script defined with $CUSTOM_SCRIPT_PATH will be executed with each execution of xkit. The skill file defined with $CUSTOM_SKILL_PATH will be executed with each execution of Cadence Virtuoso.

4.3.2.1 Example for customization with $CUSTOM_SCRIPT_PATH

An example for the customization with $CUSTOM_SCRIPT_PATH can be found in the XENV package. In a project directory which was created or updated with XENV version 1.2.1 or higher, this example can be found in the /templates directory of xenv:

```
./.xkit/setup/x_all/cadence/xenv/templates/xfMemsPlus.pl
```

This perl script defines the setup for Coventor MEMS+, based on a project created with xkit. The script renames the existing .cdsinit, creates a new .cdsinit, which loads the MEMS+ specific setup followed by the original .cdsinit, and copies cdsLibMgr.il from the MEMS+ installation directory. As a result, the MEMS+ menue will be available in Cadence library manager.

The script can be used as it is or just as an example for a possible customization. It can either be defined directly as value of CUSTOM_SCRIPT_PATH, or be wrapped into another script.

4.3.3 .xfabcadrc

xkit creates .xfabcadrc in $HOME, if this file does not exist. xkit reads .xfabcadrc in $HOME and $WORK, and sets the variables that are defined (skill syntax).

Currently the following parts of the PDK customization can be modified if the corresponding variables are set in .xfabcadrc:

xkit to create no backup files in project directory	`XfabXenvNoBackupFiles = t`
absolute paths in setup files: point to local .xkit/setup	`XfabPathsToLocalXkit = t`
xkit to use absolute paths in physical verification setup files, pointing to $X_DIR	`XfabNoRelPathsInSetup = PV`
xkit to update .cadence directory in $HOME, not in $CWD	`XfabUpdateDotCadenceInHomeDir = t`
suppress all EDA tool default settings, which are not related to PDK definitions	`XfabCadNoToolDefaults = t`
suppress libManager-popup with start of virtuoso	`XfabCadNoOpenLibManager = t`
suppress loading of default bindkeys	`XfabCadNoSetBindKey = t`
suppress definition of default log filters	`XfabCadNoSetFilter = t`
default web browser for displaying data books	`XfabDefaultWebBrowser = "konqueror"`
simulation run directory	`XfabAsimenvRunDir = "/netrun/Sim"`
Assura DRC run directory	`XfabAssuraDrcRunDir = "verification/assura_drc"`
Assura LVS run directory	`XfabAssuraLvsRunDir = "verification/assura_lvs"`

PVS DRC run directory	`XfabPvsDrcRunDir = "verification/pvs_drc"`
PVS LVS run directory	`XfabPvsLvsRunDir = "verification/pvs_lvs"`
PVS FastXor run directory	`XfabPvsXorRunDir = "verification/pvs_xor"`
Calibre DRC run directory	`XfabCalibreDrcRunDir = "verification/calibre_drc"`
Calibre LVS run directory	`XfabCalibreLvsRunDir = "verification/calibre_lvs"`
Calibre PEX run directory	`XfabCalibrePexRunDir = "verification/calibre_pex"`

4.4 Migration of existing "tkit" projects

Existing tkit projects can be easily migrated to xkit, using the option -u|--update. The project directory should contain a .cdsinit file, which defines the X-FAB technology to be used.

xkit -u will create the directory .xkit and all other missing setup files. Existing setup files will be updated. xkit -u will not change cds.lib and .cdsinit_personal (see Section 4.1.3).

It is recommended to the option -m|--modules, if the PDK has also been updated. -m will update the chosen PDK modules in cds.lib and .cdsinit.

5 Support

Technical questions should be directed to X-FAB Hotline & Technical Support:

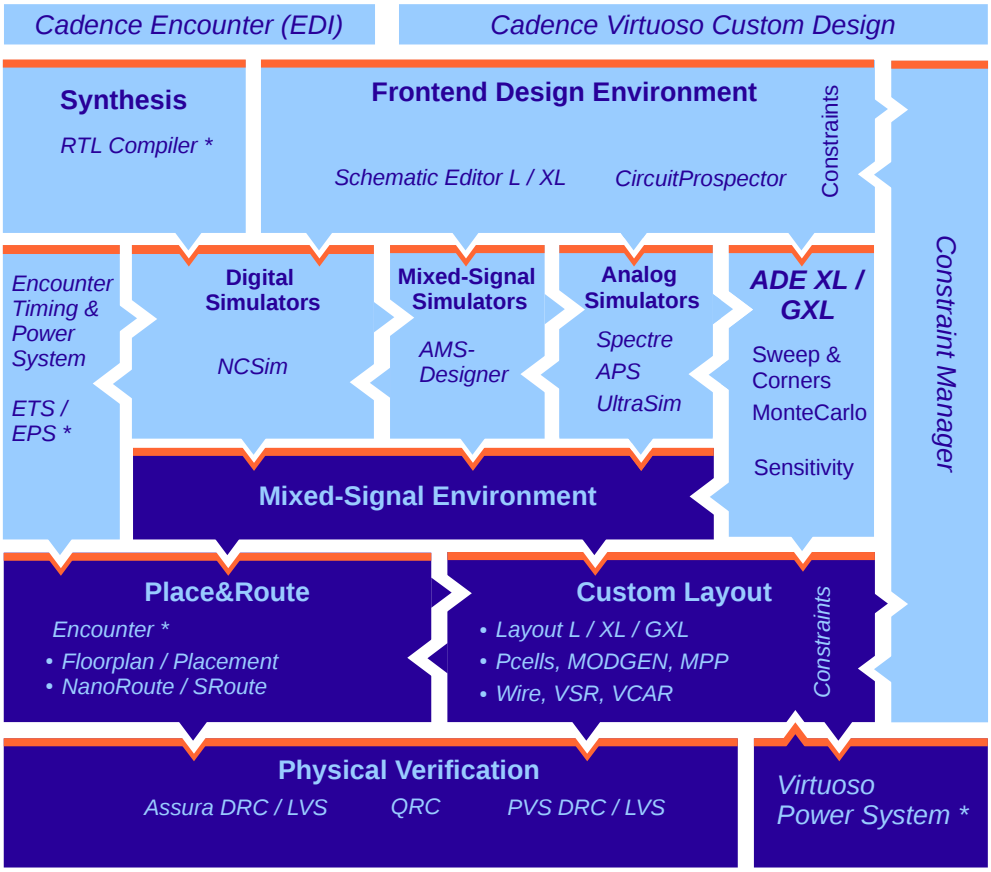
email:	hotline@xfab.com
Phone	+49 361 4276663
Fax	+49 361 4276631

Appendices

A Supported EDA tools

Figure 2 provides an overview about the Cadence design tools and flows that are supported by X-FAB design kits. XENV creates a predefined setup for most of the Cadence tools and flows, and also for some popular 3rd party OpenAccess EDA tools which can be used in a Cadence design flow.

Figure 2 Supported Cadence Design Flow



* no predefinitions by XENV, but supported by X-FAB design kit data

A.1 Cadence EDA tools

A.1.1 General setup

- default CIW log filter (can be suppressed with .xfabcadrc)
- default layer- and object map tables for xStream

A.1.2 Frontend Design Environment

- default schematic bindkeys (can be suppressed with .xfabcadrc)
- default setup for Virtuoso Circuit Prospector
- Xkit Utils schematic functions in Schematic L/XL

A.1.3 Analog Simulators

- default model path for simulators spectre, aps, UltraSim

A.1.4 Digital Simulators

- default view lists for simulator NCVerilog

A.1.5 Mixed Signal Simulators

- default model path for simulator ams
- default netlister for ams: OSS-based netlister with irun

A.1.6 Custom Layout

- default display file
- default layout bindkeys (can be suppressed with .xfabcadrc)
- Xkit Utils layout functions in Layout L/XL/GXL
- default runset for layoutEAD

A.1.7 Physical Verification

- default runsets for Assura, PVS, QRC

A.2 Other EDA tools

A.2.1 Physical Verification

- default runsets for Mentor Calibre DRC/LVS/PEX (environment variable $MGC_HOME needs to be defined)
- default setup for SiliconFrontline R3D

A.2.2 Analog Simulators

- default model path for Agilent GoldenGate
- default model path for Muneda Wicked

A.2.3 Others

- setup for Coventor MEMS+ (see Section 4.3.2.1)

B Files and directories in the project directory

B.1 Project directory created by XENV xkit

XENV xkit creates the following files and directories in the project directory (example for process family XH018):

`.assuraSetup`	• contains Assura techRuleSet- and setup files, with path definitions according to .xkit directory structure • optional; only created if Assura runset is available
`.avviewinit`	• setup file for Assura standalone flow • optional; only created if Assura runset is available
`.cadence`	• hierachy definition for Cadence library manager
`.calibreSetup`	• setup file for Calibre DRC, LVS and PEX • optional; only created if Calibre runset is available
`.cdsenv`	• default Cadence environment variable definitons
`.cdsinit`	• X-FAB PDK skill code
`.cdsinit_personal`	• customizable skill file, loaded from .cdsinit
`.cdswcd`	• default setup for Muneda Wicked • optional; only created if Muneda Wicked model package is available
`.pvsSetup`	• setup file for PVS DRC and LVS • optional; only created if PVS runset is available

`.simrc`	• default simulation settings
`.xkit`	• directory containing linked design kit structure • all path definitions in the setup files which are created by xkit are relative to ./.xkit
`assura_tech.lib`	• defines the available assura runsets • optional; only created if Assura runset is available
`cds.lib`	• Cadence library definitions file
`data.reg`	• definition of X-FAB data formats
`proj_opt.txt`	• PDK options file • lists the chosen PDK modules
`pvtech.lib`	• defines the available PVS runsets • optional; only created if PVS runset is available
`xh018.inc`	• PDK module specific verilog include file
`xh018.lib`	• PDK module specific library definitions file • included in cds.lib
`xh018_combine.lib`	• PDK module specific library combine statements • included in cds.lib

B.2 Update of the project directory

The existance of each of the files and directories which are described in Section B.1 is checked with each execution of xkit. Missing files and directories are re-created with default content.

The following files and directories which contain module-specific content are updated with each PDK module update (xkit -m). These files are also updated with each library version update (xkit -u|-s), if the variable XfabNoRelPathsInSetup is defined in .xfabcadrc (see Section 4.3.3).

```
.assuraSetup .avviewinit .cdsinit .pvsSetup .simrc assura_tech.lib ↩
    proj_opt.txt pvtech.lib xh018.inc xh018.lib xh018_combine.lib
```

The following files and directories are updated with each library version update (xkit -u|-s)

```
.xkit xh018.inc xh018.lib xh018_combine.lib
```