CEVA-XM4™

RTL V1.1.3.F Backend Reference Guide

Rev. 1.1.3.F

June 2016

DSP Division

Documentation Control

History Table

Version	Date	Description	Remarks
V1.0.0.A	26 February 2015	Initial version	
V1.0.0.F	16 April 2015	Updated RTL version	
V1.1.0.A	1 July 2015	Updated RTL version	
V1.1.0.F	6 August 2015	Updated RTL version	
V1.1.1.F	18 January 2016	Updated RTL version	
V1.1.2.F	14 March 2016	Updated RTL version	
V1.1.3.F	08 June 2016	Updated RTL version	

Disclaimer and Proprietary Information Notice

The information contained in this document is subject to change without notice and does not represent a commitment on any part of CEVA®, Inc. CEVA®, Inc. and its subsidiaries make no warranty of any kind with regard to this material, including, but not limited to implied warranties of merchantability and fitness for a particular purpose whether arising out of law, custom, conduct or otherwise.

While the information contained herein is assumed to be accurate, CEVA®, Inc. assumes no responsibility for any errors or omissions contained herein, and assumes no liability for special, direct, indirect or consequential damage, losses, costs, charges, claims, demands, fees or expenses, of any nature or kind, which are incurred in connection with the furnishing, performance or use of this material.

CEVA®, CEVA-XC™, CEVA-XC5™, CEVA-XC8™, CEVA-XC321™, CEVA-XC323™, CEVA-Xtend™, CEVA-XC4000™, CEVA-XC4100™, CEVA-XC4200™, CEVA-XC4210™, CEVA-XC4400™, CEVA-XC4410™, CEVA-XC4500™, CEVA-XC4600™, CEVA-TeakLite™, CEVA-TeakLite-II™, CEVA-TeakLite-III™, CEVA-TL3210™, CEVA-TL3211™, CEVA-TeakLite-4™, CEVA-TL410™, CEVA-TL411™, CEVA-TL420™, CEVA-TL421™, CEVA-Quark™, CEVA-Teak™, CEVA-X™, CEVA-X1620™, CEVA-X1622™, CEVA-X1641™, CEVA-X1643™, Xpert-TeakLite-II™, Xpert-Teak™, CEVA-XS1100A™, CEVA-XS1200™, CEVA-XS1200A™, CEVA-TLS100™, Mobile-Media™, CEVA-MM1000™, CEVA-MM2000™, CEVA-SP™, CEVA-VP™, CEVA-MM3000™, CEVA-MM3100™, CEVA-MM3101™, CEVA-XM™, CEVA-XM4™, CEVA-X2™ CEVA-Audio™, CEVA-HD-Audio™, CEVA-VoP™, CEVA-Bluetooth™, CEVA-SATA™, CEVA-SAS™, CEVA-Toolbox™, SmartNcode™ are trademarks of CEVA, Inc.

All other product names are trademarks or registered trademarks of their respective owners.

Support

CEVA® makes great efforts to provide a user-friendly software and hardware development environment. Along with this, CEVA provides comprehensive documentation, enabling users to learn and develop applications on their own. Due to the complexities involved in the development of DSP applications that might be beyond the scope of the documentation, an online Technical Support Service has been established. This service includes useful tips and provides fast and efficient help, assisting users to quickly resolve development problems.

How to Get Technical Support:

- **FAQs**: Visit our website http://www.ceva-dsp.com or your company's protected page on the CEVA website for the latest answers to frequently asked questions.
- **Application Notes**: Visit our website http://www.ceva-dsp.com or your company's protected page on the CEVA website for the latest application notes.
- **Email**: Use CEVA's central support email address ceva-support@ceva-dsp.com. Your email will be forwarded automatically to the relevant support engineers and tools developers who will provide you with the most professional support to help you resolve any problem.
- **License Keys**: Refer any license key requests or problems to sdtkeys@ceva-dsp.com. For SDT license keys installation information, see the *SDT Installation and Licensing Scheme Guide*.

Email: ceva-support@ceva-dsp.com

Visit us at: www.ceva-dsp.com

List of Sales and Support Centers

Israel	USA	Ireland	Sweden
2 Maskit Street P.O. Box 2068 Herzelia 46120 Israel **Tel**: +972 9 961 3700 **Fax**: +972 9 961 3800	1174 Castro Street Suite 210 Mountain View, CA 94040 USA **Tel**: +1-650-417-7923 **Fax**: +1-650-417-7924	Segrave House 19/20 Earlsfort Terrace 3rd Floor Dublin 2 Ireland **Tel**: +353 1 237 3900 **Fax**: +353 1 237 3923	Klarabergsviadukten 70 Box 70396 107 24 Stockholm Sweden **Tel**: +46(0)8 506 362 24 **Fax**: +46(0)8 506 362 20
China (Shanghai)	**China (Beijing)**	**China (Shenzhen)**	**Hong Kong**
Unit 1203, Building E Chamtime Plaza Office Lane 2889, Jinke Road Pudong New District Shanghai, 201203 China **Tel**: +86-21-20577000 **Fax**: +86-21-20577111	Rm 503, Tower C Raycom InfoTech Park No.2, Kexueyuan South Road Haidian District Beijing 100190 China **Tel**: +86-10 5982 2285 **Fax**: +86-10 5982 2284	Rm 709, Tower A SCC Financial Centre No. 88 First Haide Avenue Nanshan District Shenzhen 518064 China **Tel**: +86-755-8435 6038 **Fax**: +86-755-8435 6077	Level 43, AIA Tower 183 Electric Road North Point Hong Kong **Tel**: +852-39751264
South Korea	**Taiwan**	**Japan**	**France**
#478, Hyundai Arion 147, Gumgok-Dong Bundang-Gu Sungnam-Si Kyunggi-Do, 463-853 South Korea **Tel**: +82-31-704-4471 **Fax**:+82-31-704-4479	Room 621 No.1, Industry E, 2nd Rd Hsinchu, Science Park Hsinchu 300 Taiwan R.O.C **Tel**: +886 3 5798750 **Fax**: +886 3 5798750	1-6-5 Shibuya SK Aoyama Bldg. 3F Shibuya-ku, Tokyo 150-0002 Japan **Tel**: +81-3-5774-8250	RivieraWaves S.A.S 400, avenue Roumanille Les Bureaux Green Side 5, Bât 6 06410 Biot - Sophia Antipolis France **Tel**: +33 4 83 76 06 00 **Fax**: +33 4 83 76 06 01

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Scope

This document describes the CEVA-XM4™, which is a synthesizable and reusable IP DSP core that reduces time-to-market for DSP subsystem development. The IP source code is a process-independent Verilog RTL code that can be easily embedded in a SoC. Some modules can be instantiated as Design Ware modules. Limited usage of library cell instantiation is done for the clock-gating module.

1.2 Audience

This document is intended for frontend and backend ASIC designers who implement the CEVA-XM4 IP.

1.3 Related Documents

The following documents are related to the information in this document:

1. *CEVA-XM4 Integration Reference Guide*
2. *CEVA-XM4 Simulation Reference Guide*
3. *CEVA- XM4 Database Reference Guide*
4. *CEVA- XM4 Power Modes Guide*

1.4 Disclaimer

All of the timing constraints and setup files provided in this document are for reference **only**. These files must be updated according to the target process and library.

2. Reference Flow Scenarios

The CEVA-XM4 backend reference flow describes a full backend flow from RTL to GDSII using 28nm HPM TSMC process, an ARM standard cell library, and ARM 28nm HD memories.

The reference flow demonstrates the use of multi-VT libraries, with a specific focus on achieving timing while minimizing power leakage.

Table 2-1 summarizes the RTL configuration used in the reference flow.

Table 2-1: CEVA-XM4 Backend Flow Configuration

Configuration Option	Value
Enhanced OCEM	Enabled
NFF	Disabled
CEVA-MM3101 compatibility	Disabled
Scalar floating point	4 (one floating point per SPU)
Vector floating point	16 (eight floating points per VPU)
QMAN configuration	None
Nonlinear function support	32
Data memory	256 KB in 4 blocks
Number of DACU regions	32
Program TCM size	32 KB
Program Cache size	64 KB
Memory power gating	Disabled
AXI Masters	2 AXI Masters 128-bit width
AXI Slaves	3 AXI Slaves 128-bit width
ECC	Disabled
RTT	Disabled
Xtend	Disabled

2.1 Reference Files

The reference flow usually includes the following files:

- Synthesis logs and reports
- Floorplan examples (that is, reference DEF files)
- Place and route logs and reports (some for CLN28HM floorplanning, place, clock time synthesis (CTS), and route stages)
- PT logs and reports
- Formality logs
- Star logs

These files are provided as a TAR file, and are delivered separately from the release package.

3. Implementation Database Structure

Figure 3-1 shows the CEVA-XM4 implementation environment tree, with or without the power gating flow.

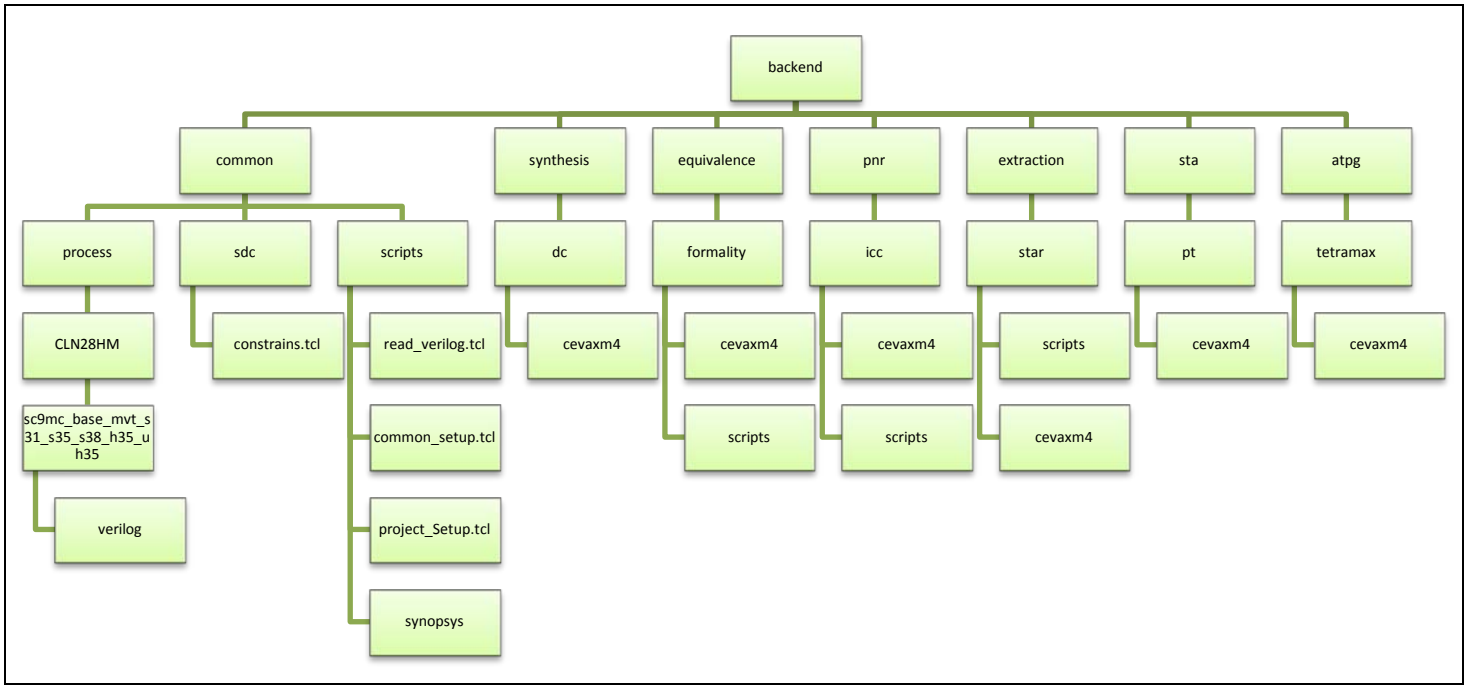

Figure 3-1: CEVA-XM4 backend/ Directory Structure

Table 3-1 and Table 3-2 describe the top-level **backend/** directories.

Table 3-1: CEVA-XM4 Top-Level backend/ Directories

Directory Name	Description
common	Root directory for common scripts and common data, for example, SDC constraints and Verilog files. Contains the following subdirectories: • **process**: Process-dependent scripts and Verilog files • **sdc**: Timing constraints files • **scripts**: Common scripts that should be sourced by all tools For more details, see Table 5-1.
synthesis	Root directory for synthesis runs. See Table 3-2 for the subdirectories.
equivalence	Root directory for equivalence check runs. See Table 3-2 for the subdirectories.
pnr	Root directory for place/CTS/route runs. See Table 3-2 for the subdirectories.
extraction	Parasitic extraction. See Table 3-2 for the subdirectories.
sta	Root directory for STA runs. See Table 3-2 for the subdirectories.
atpg	Root directory for ATPG runs. See Table 3-2 for the subdirectories.

Table 3-2: CEVA-XM4 Synopsys Database Directories

Directory Name	Description
synthesis/dc	Root directory for the CEVA-XM4 design compiler synthesis directories. For more details, see Table 5-1.
equivalence/formality	Root directory for the Formality tool for checking functional equivalence. For more details, see Table 6-1.
pnr/icc	Root directory for the CEVA-XM4 IC Compiler (place and route)
extraction/star	Root directory for the CEVA-XM4 Star-RCXT directories
sta/pt	Root directory for the CEVA-XM4 PrimeTime directories
atpg/tetramax	Root directory for the CEVA-XM4 TetraMax directories

4. Implementation Flow

The following sections describe the CEVA-XM4 implementation flow from RTL to routed netlist, as shown in Figure 4-1.

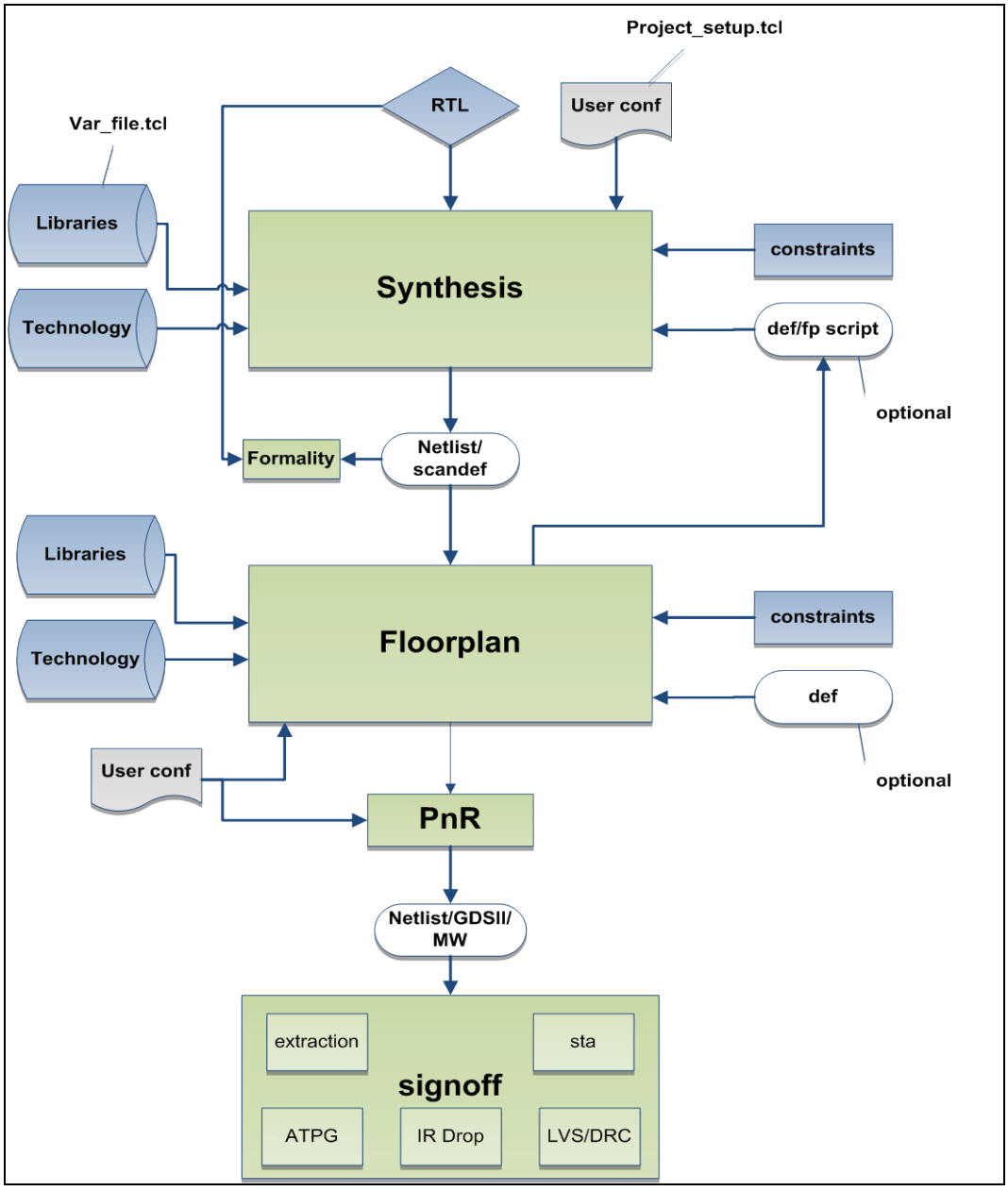

Figure 4-1: CEVA-XM4 Implementation Flow

Before running any of the stages in this section (except **star** and **tetramax**), the general **setenv** file should be sourced by typing:

```
source <install_dir>/scripts/setenv.csh
```

4.1 Stage #1 – RTL Simulation

Objectives:

Simulate and verify the RTL via assembly tests to ensure correct functionality

Tools supported in the database:

IES, VCS, or ModelSim Verilog simulator

Inputs:

- RTL code
- Simulation environment
- Test suite

4.2 Stage #2 – Synthesis

Objectives:

- Synthesize the RTL code
- Perform scan chain stitching

Tools supported in the database:

Design Compiler (DC), Design Compiler Topographical (DCT), and Design Compiler Graphical (DCG)

Inputs:

- RTL code
- Boundary constraints (load and drive cells)
- Timing and design rule constraints
- False paths, multicycles, and case analysis
- Floorplan DEF file for synthesis runs (optional)
- Scan cell guidelines for scan insertion
- Standard cell library (in .DB or .LIB format)

Outputs:

- Prelayout netlist
- Timing report
- Scandef (file defining the scan chains for the ICC)
- Database file (DDC file)
- DEF file (optional)

4.3 Stage #3 – Equivalence Checking

Objective:

Check the equivalence of the CEVA-XM4 netlist (the output of Stage #2 – Synthesis) against the RTL source file

Tool supported in the database:

Formality

Inputs:

- Post-synthesis netlist (the output of Stage #2 – Synthesis)
- RTL code
- Standard cell library

Output:

Report

4.4 Stage #4 – Floorplan Creation

Objective:

Create a floorplan

Tool supported in the database:

Synopsys IC Compiler (ICC)

Inputs:

- Post-synthesis netlist (the output of Stage #2 – Synthesis)
- DC (DCG-SPG) database file (DDC file) if running an ICC-SPG flow
- Boundary constraints (size, port location)
- Memory locations
- Keepout locations
- Standard cell and memory library
- Technology files (**tf**, **tluplus**, and **map** files)

Output:

Milkyway floorplan CEL

4.5 Stage #5 – Place

Objectives:

- Place the cells
- High fan-out net synthesis
- Reorder the scan chains

Tool supported in the database:

ICC

Inputs:

- Milkyway floorplan CEL (the output of Stage #4 – Floorplan Creation)
- Standard cell and memory library
- Boundary constraints (load and drive cells)
- Timing and design rule constraints
- Scandef (scan chain information; the output of Stage #2 – Synthesis)

Outputs:

- Placed Milkyway CEL
- Post-place netlist

4.6 Stage #6 – Clock Tree Synthesis

Objective:

Synthesize the clock tree and route clock only

Tool supported in the database:

ICC

Inputs:

- Placed Milkyway CEL (the output of Stage #5 – Place)
- Constraints (timing and CTS)
- Standard cell and memory library
- Technology files (**tf**, **tluplus**, and **map** files)

Outputs:

- Milkyway CEL with clock tree and clock route
- Post-CTS netlist

4.7 Stage #7 – Route

Objective:

Route the Milkyway CEL

Tool supported in the database:

ICC

Inputs:

- Post-CTS Milkyway CEL (the output of Stage #6 – Clock Tree Synthesis)
- Constraints
- Standard cell and memory library
- Technology files (**tf**, **tluplus**, and **map** files)

Outputs:

- Routed Milkyway CEL
- Routed (post-layout) netlist

4.8 Stage #8 – RC Extraction

Objective:

Parasitic extraction of the CEVA-XM4

Tool supported in the database:

STAR-RCXT

Inputs:

- Milkyway database, routed CEL (the output of Stage #7 – Route)
- STAR-RCXT technology file (NXTGRD file)
- Mapping file

Output:

Parasitic file (SBPF/SPEF)

4.9 Stage #9 – Equivalence Checking

Objective:

Check the equivalence of the CEVA-XM4 routed netlist (the output of Stage #7 – Route) against the RTL source file

Tool supported in the database:

Formality

Inputs:

- Routed and clocked netlist (the output of Stage #7 – Route)
- RTL code
- Standard cell library

Output:

Report

4.10 Stage #10 – Static Timing Analysis (STA)

Objective:

Check the STA (setup, hold, and clock skew)

Tools supported in the database:

PrimeTime (PT) or PrimeTime-SI

Inputs:

- Post-layout netlist (the output of Stage #7 – Route)
- Boundary constraints (load and drive cells)
- Timing and design rule constraints
- Standard cell library (.DB or .LIB format)
- Parasitic file (SBPF/SPEF; the output of Stage #8 – RC Extraction)

Outputs:

- STA reports
- Global skew reports
- Standard Delay Format (SDF) file for Gate -Level simulation, with timing

4.11 Stage #11 – ATPG

Objectives:

- Simulate scan patterns
- Verify test coverage
- Create test patterns

Tool supported in the database:

TetraMax

Inputs:

- Post-layout netlist (the output of Stage #7 – Route)
- Scan chain structure and constraints

Outputs:

- Reports
- Test patterns

4.12 Stage #12 –Gate-Level Simulation with Timing (SDF)

Objectives:

Simulate and verify the post-layout CEVA-XM4 gate-level netlist via assembly tests to ensure correct functionality

Tools supported in the database:

IES, VCS, or ModelSim Verilog simulator

Inputs:

- Post-layout netlist (the output of Stage #7 – Route)
- SDF file (the output of Stage #10 – Static Timing Analysis (STA))
- Simulation environment
- Test suite

Output:

Reports

5. Synthesis

The following sections describe the CEVA-XM4 synthesis environment configurations that must be done before synthesizing:

1. Configure the **Var_file.tcl** file.
2. Adapt the CLOCK_GATER module to the technology library.
3. Prepare the synthesis constraints.

5.1 CLOCK_GATER Module

The CEVA-XM4 design uses clock gaters to reduce power consumption. The CLOCK_GATER module contains an active low LATCH, an INVERTER, two AND gates, and an OR gate.

As shown in Figure 5-1, the clock gater module (***<install_dir>/cevaXM4_V1.1.3.F/design/top/general/ceva_clock_gater.v***), must be configured before running synthesis. Specific standard cells from the target library should be instantiated in this module.

Figure 5-1 also shows that some libraries include a clock-gating cell, which consists of an OR gate, an active low LATCH, and an AND gate. It is recommended to use this cell if it exists.

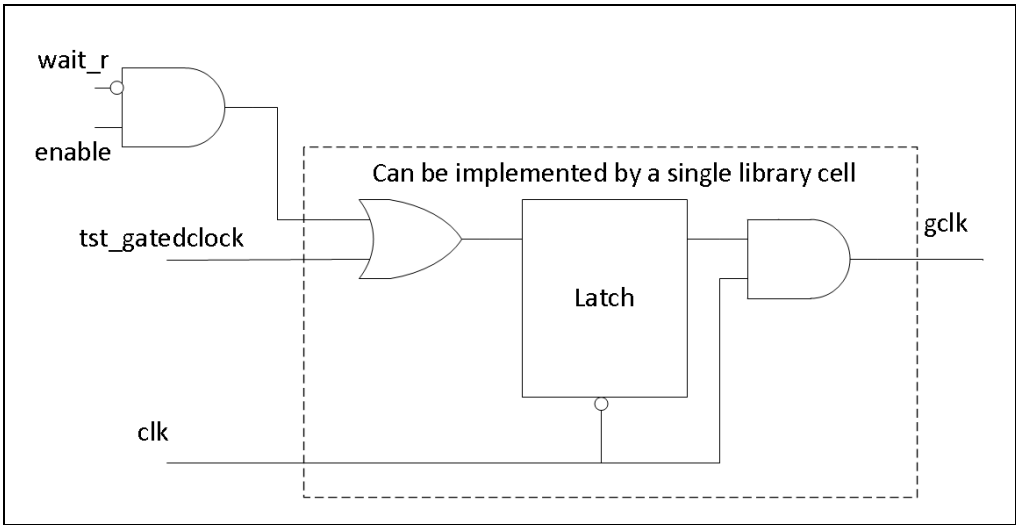

Figure 5-1: Clock Gater Module

5.2 Synthesis Flow

This section describes the synthesis process of the CEVA-XM4.

The structure of the CEVA-XM4 synthesis directory is shown in Figure 5-2. The synthesis script contains all of the scripts needed for running synthesis.

Synthesis can be run in several modes:

- Using a DC topographical with either a real floorplan DEF file or a floorplan TCL script as an input
- Using a DC topographical with physical guidance (DCG-SPG)
- Using a DC topographical without a real floorplan but with physical parameters instead (aspect ratio, utilization, and so on)

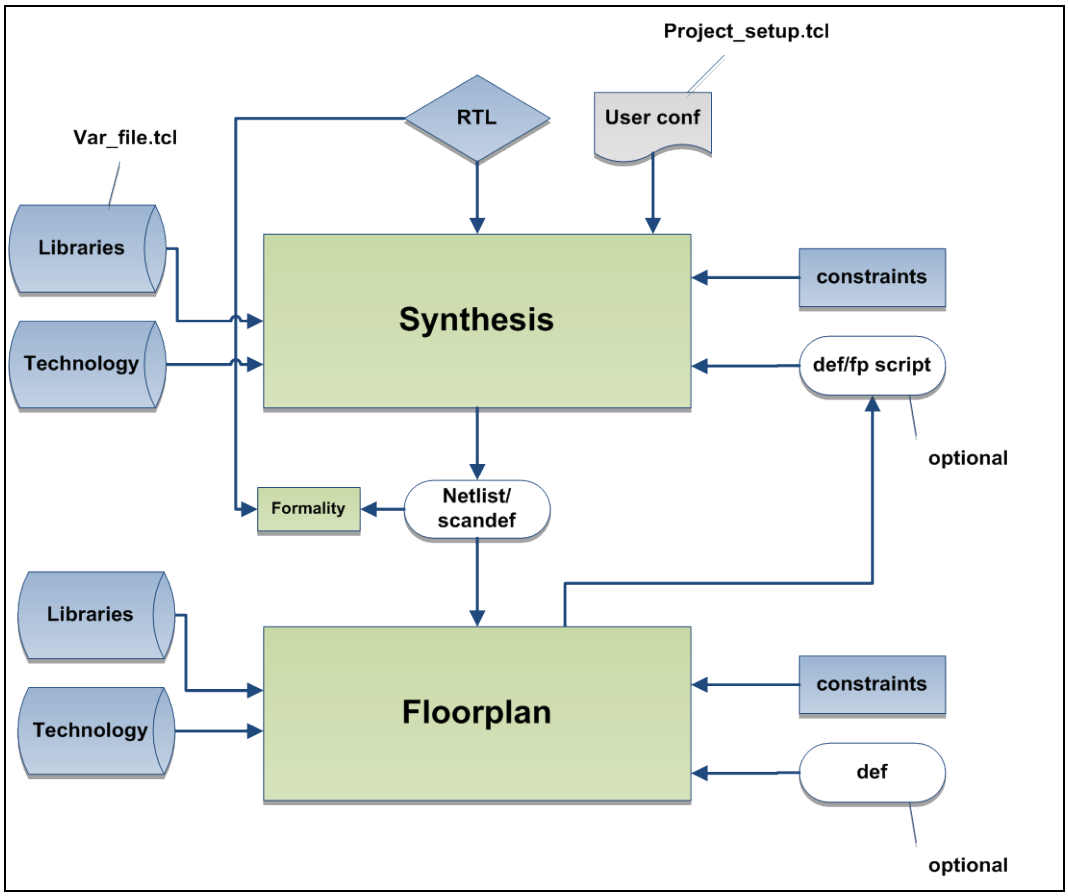

Figure 5-2: Synthesis Block Diagram

5.3 Synthesis Directory Structure

Figure 5-3 shows the synthesis directory tree.

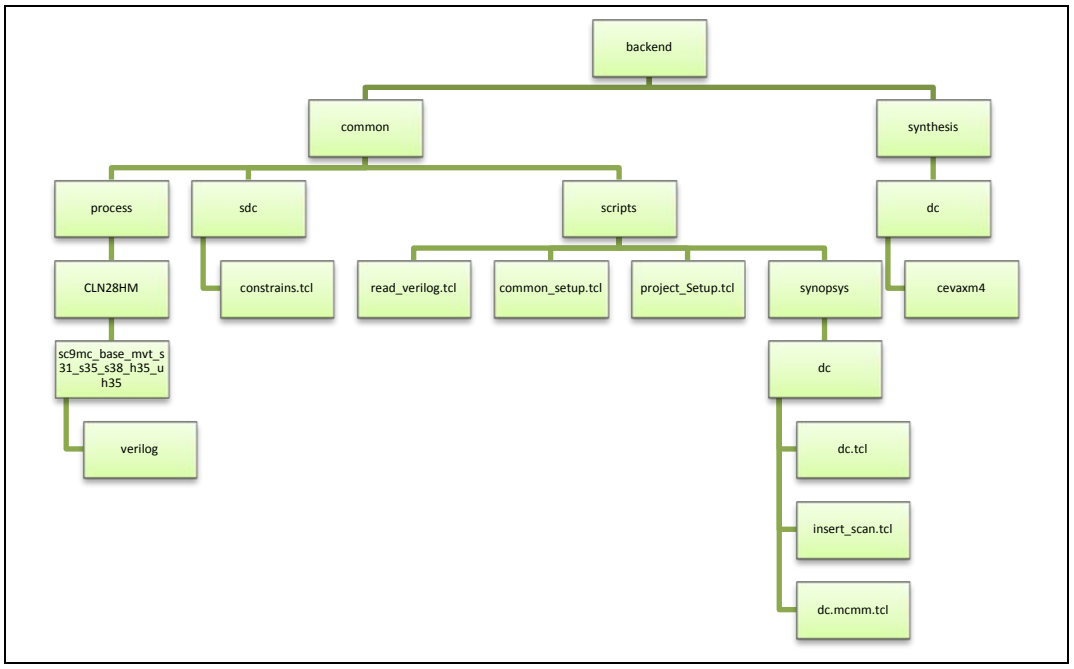

Figure 5-3: Synthesis Directory Tree

Table 5-1 describes the **backend/common/** and **synthesis/** database directories.

Table 5-1: CEVA-XM4 common/ and synthesis/ Directories

Directory Name	Description
common/process/<process_nam>/<library name>/verilog	Location of clock gater netlist with process-specific cell instantiations
common/sdc	Location of the timing constraints files
common/scripts	Common scripts that should be sourced by all tools. For more details, see Table 6-1.
common/scripts/synopsys/dc	Location of DC scripts
synthesis/dc/cevaxm4	Run directory (reports, results, and log directories are created during the synthesis under this working directory)

Note: *The timing constraints of all of the modules in the design are for reference only, and must be reviewed by the user.*

5.3.1 Synthesis Setup

The following sections describe what must be updated before CEVA-XM4 synthesis.

5.3.1.1 project_setup.tcl File

The **project_setup.tcl** file, which is located under the **backend/common/scripts** directory, includes settings that are relevant to (almost) all of the tools in the CEVA-XM4 flow. The user can modify these settings by either typing new values instead of them or modifying the **user settings** section at the end of the file.

Table 5-2 describes the settings that can be updated.

Table 5-2: Library project_setup.tcl Parameters

Parameter	Description
CycleTime	CEVA-XM4 clock period (default is **2.2 ns**).
Process	Process name (default is **CLN28HM**).
StdLibName	Standard cell library name (default is **sc9mc_base_mvt_s31_s35_s38_h35_uh35**).
MemVendor	The memory vendor (default is **arm**).
SetupClockUncertainty	Sets the clock uncertainty for setup.
HoldClockUncertainty	Sets the clock uncertainty for hold.
SynthWithFloorplan	When set to **TRUE**, DCT with a real floorplan is done; otherwise, DCT with floorplan parameters is done (for example, aspect ratio and utilization) but no real floorplan is used as an input to the synthesis run.
CompileWithSpg	When set to **TRUE**, DCG-SPG (the **-spg** flag in compile_ultra) is done. To avoid errors, **SynthWithFloorplan** must also be set and a proper floorplan must be provided.
DEF_FILE	Sets the path to the floorplan DEF file when running a DCT flow with a real floorplan
FpUtilization	Used only with DCT. Sets the required utilization when running a DCT flow without a real floorplan (that is, based on physical parameters)
FpAspectRation	Used only with DCT. Sets the required aspect ratio when running a DCT flow without a real floorplan (that is, based on physical parameters)
PropagationDelay(ceva_free_clk)	The propagation delay for ceva_free_clk (default is **0**).
UseScan	If insert scan is used, this is set to **TRUE**; otherwise, **FALSE**.

Parameter	Description
ScanChains	Number of scan chains. The **sci[*]** and **sco[*]** ports are created automatically if **UseScan** is set to **TRUE**.
OutputLoad	Output load on non-memory I/F outputs.
MaxTran	Sets the maximum transition.
MaxFan	Sets the maximum fanout.
MaxCap	Sets the maximum capacitance.
CriticalRange	Critical range value for path optimization (recommended value is **0.2 ns**).
PmemTcmLatency	PMSS TCM set latency for all flows, including CTS exceptions.
PmemCacheLatency	PMSS cache latency for all flows, including CTS exceptions.
DmemTcmLatency	DMSS TCM set latency for all flows, including CTS exceptions.
FLOORPLAN_INPUT	Determines how the floorplan is created: 1. CREATE (default) 2. DEF
ClockGateSplit	Uses the clock gater split command in CTS (default is **false**).
CoreWidth	Sets the width of the die size (default is **2000**).
CoreHeight	Sets the height of the die size (default is **2000**).
DCResultsDir	The location directory of the netlist for ICC, Formality and TetraMax (default is **../../../synthesis/dc/cevaxm4/**).
SVF_FILE	SVF filename and location (default is **${DCResultsDir}/${DesignName}.svf**).
VERILOG_NETLIST_FILE	Verilog netlist filename and location (default is **${DCResultsDir}/${DesignName}.scan.v**).
SCAN_DEF_FILE	Scandef filename and location (default is **${DCResultsDir}/${DesignName}.scandef**).
DontMergeCells	One of the following: • **ALL** = Sets "don't_merge" on all of the registers • **big** - Sets "don't_merge" over critical registers only (the default) • **False** or **0** = Merging allowed These registers are based on 28nm HPM runs and might differ depending on the process or run. The names of the registers are defined in the <install_dir>/backend/common/sdc/constraints.tcl file.
SpgFlow	Determines in the SPG flow is run (default is **false**).

Parameter	Description
CompileTimingOpt	Provides higher priority for timing performance during synthesis (default is **true**).
CompileAreaOpt	Provides higher priority for area reduction during synthesis (default is **false**).
CompileHierarcy	Determines if hierarchical synthesis is run (default is **false**).
CpuNum	Number of CPUs that are used during a DC/ICC run.
UseSubClocks	When set to **TRUE**, sets the generated clocks according to the PSU clock gaters. The PSU has unique clock gaters for each pipe-stage and unit within the processor. When this switch is enabled, the backend tool t works simultaneously on various path optimizations.
RunMemsUsufulSkew	When set to **TRUE**, the clock latency to the memories is automatically adjusted according to timing slack from/to the memories during DC/ICC.
ApplyAutoWeight	When set to **TRUE**, the group paths that exist in the design (except for INPUT/OUTPUT/FEEDTHROUGH) are dynamically adjusted according to the path group slack.
Directory name has pattern: pden<number>cull	When detected, the flow marks library cells that have pin cell density greater than <number> as don't_use.
Directory name has pattern: oversize<number>pct	When detected, stretches all standard library cells on their left and right sides, effectively increasing their area by <number>%.
PowerGridUtilization	When set with pairs of layers and percentages (for example, **M2 2% M3 2% M4 2% M5 3% M6 3%**), the tool reserves routing resources for the PG as stated.
CtsConcurrentClockData	When set to **TRUE**, the CTS step of the flow enables concurrent clock and data optimization (for example, in ICC this uses **clock_opt - concurrent_clock_and_data**). Disabled by default.
RouteFocalOpt	When set to **TRUE**, the route step of the flow included some post-route **focal_opt**-based optimizations. Disabled by default.
CloneRun	Used in **run.tcl**. When set, the tool creates a branch in the flow, and outputs all new databases with suffix **_$CloneRun**. This is currently only implemented in the CTS step of the flow.
CloneRunStartPoint	Used in run.tcl. When set, the tool selects a starting database with a prefix of **_$CloneRunStartPoint**. This is currently only implemented in the CTS step of the flow.

Parameter	Description
RunMCMM	Enables the Multi-Mode Multi corner flow. The following targets must also be set: • **set LvtPrcnt <integer>** • **set HVT_lib <lib_file_list>** • **set LVT_lib <lib_file_list>**
PlaceOptOptimizePower	Enables power optimization for the place stage. When running with **RunMCMM**, this parameter is set to **TRUE** (default is **false**).
CtsOptOptimizePower	Enables power optimization for the CTS stage. When running with **RunMCMM**, this parameter is set to **TRUE** (default is **false**).
RouteOptOptimizePower	Enable power optimization for Route stage. When running with **RunMCMM**, this parameter is set to **TRUE** (default is **false**).

5.3.1.2 Var_file.tcl Files

The backend environment **common/process** directory is built as **backend/common/process/<process_name>/<lib_name>**. The default scripts define the process as **CLN28HM** and the library as **sc9mc_base_mvt_s31_s35_s38_h35_uh35**.

The following **Var_file.tcl** files must be updated:

- **backend/common/process/<process_name>/Var_file.tcl**: This file includes all of the settings that are relevant to the process.

  If the same process as defined in this reference flow is used, then the **backend/common/process/CLN28HM/Var_file.tcl** file can be edited. Otherwise, a new directory named **backend/common/process** should be created and named by the process used, and then a new **Var_file.tcl** file created below it that can be edited (use this release process's **Var_file** as a reference).

- **backend/common/process/<process_name>/<Library_name>/Var_file.tcl**: This file contains all of the settings that are relevant to a specific process's library.

  If the same process and library as defined in this reference flow are used, then the delivered **backend/common/process/CLN28HM/sc9mc_base_mvt_s31_s35_s38_h35_uh35 /Var_file.tcl** file can be edited**.** Otherwise, the file should be created under the **backend/common/process/<process_name>/<Library_name>/** directory and can be edited (use this release library's **Var_file** as a reference).

Table 5-3 and Table 5-4 describe the parameters that should be updated in each of the **Var_file.tcl** files.

Table 5-3: Process Var_file.tcl Parameters

Parameter	Description
MEMORIES_WORST_DB_FILES	List of memory database (worst-case) files
MemoriesMwReferenceLibDirs	Milkyway memory libraries
MemoriesLefReferences	Memory LEF files
MEM_TETRAMAX_FILES	List of Verilog memory files for TetraMax
TLUPLUS_MAX_FILE	tluplus max full path
TLUPLUS_MIN_FILE	tluplus min full path
TECH_FILE	Technology file
MAP_FILE	Map file
MinRoutingLayer	Lowest routing layer
MaxRoutingLayer	Highest routing layer
PowerNet	Power net (default VDD)
PowerPort	Power port (default VDD)
GroundNet	Ground net (default VSS)
GroundPort	Ground port (default VSS)

Table 5-4: Library Var_file.tcl Parameters

Parameter	Description
WORST_LIB_FILES	Standard cell worst-case libraries for synthesis. If the user uses a wire-load model that is located in a different file from the standard cell library, then the full path of the wire-load model (.DB) must be added. Multi-VT libraries can be included for multi-VT support.
WORST_LIB_FILES_LVT	Standard cell libraries (worst for leakage)
TYPICAL_LIB_FILES	Standard cell typical-case libraries for synthesis
BEST_LIB_FILE	Standard cell best-case libraries for synthesis
LIB_VERILOG_PATH	Verilog path for TetraMax (not supported in this release)
DrivingCell	CEVA-XM4 input driving cell (for all non-clock ports)
InputClockDrivingCell	Input driving cell for the clock port

5.3.1.3 Constraints

The CEVA-XM4 top level constraints are located in the **<install_dir>/backend/common/sdc/constraints.tcl** file. This defines the clocks, boundary constraints, input/output delay, ideal nets for DC, case analysis, and false paths for the CEVA-XM4.

5.3.1.3.1 Group Paths

Group paths are used to guide the synthesis so more effort is put where it is needed. The following are the default group paths:

- OUTPUTS: Paths to all of the outputs
- INPUTS: Paths to all of the inputs
- FEEDTHROUGH: Paths from the inputs to the outputs
- FROM_MEM: Paths from all of the memories
- TO_MEM: Paths to all of the memories
- $ClockName: Paths of all of the sequential elements that get the main clock

To improve the timing result, group paths are created based on the following guidelines:

- For a specific pipe stage, a gaters group is defined (the RTL already includes the clock gater pipe stage as part of the clock gater name).
- A generated clock is introduced to each of the gater groups.
- After creating a generated clock, a group path is automatically created.
- Each group weight is defined according to microarchitecture timing analysis.

After the first optimization stage (before the scan insertion and the second optimization), the **proc_auto_weight** script should be used to set new weights for all of the groups based on their current worst negative slack.

5.3.1.4 Tool Setup

Before running synthesis, the user should ensure that the tool license file is correctly set.

5.3.1.5 Memory Instantiation

Before running top-level synthesis, the user should edit the relevant memory instantiation in the CEVA-XM4 module. In the reference flow, the memory files are located under the following:

- **<install_dir>/design/top/cevaxm4_dmem.v**
- **<install_dir>/design/top/cevaxm4_dcmem.v**
- **<install_dir>/design/top/cevaxm4_pmem.v**

Important: *These files are given as an example, with ARM 28HM memories used at the reference flow.*

5.3.1.6 Floorplan Examples

To run a DCT with a floorplan (or when running with physical guidance, in which case the user must follow with a floorplan), a DEF file of the floorplan must be provided.

Figure **5-4** is an example of a floorplan created with the following memory configuration:

- Program: TCM 32 KB, P-CACHE 64 KB
- Data: 4 Blocks 256 KB

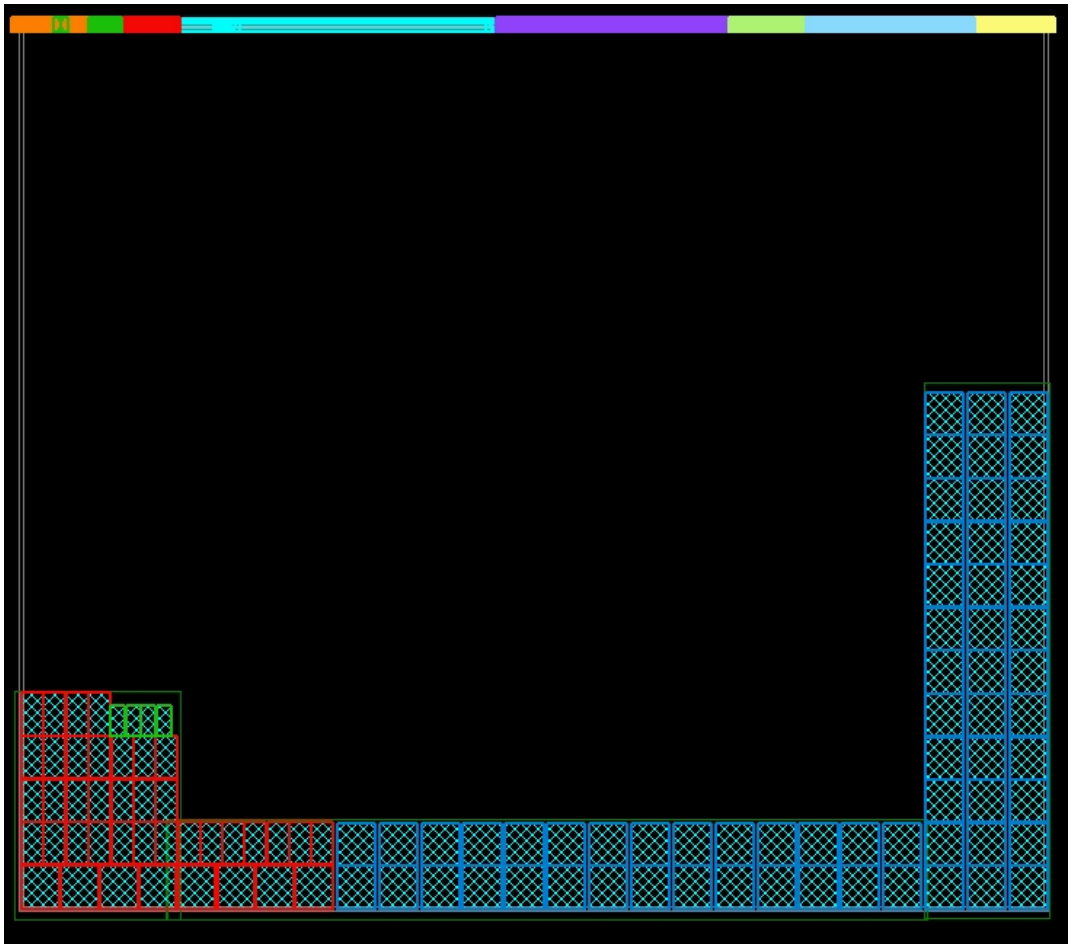

Figure 5-4: CEVA-XM4 Reference Floor-plan for default Configuration

In this floorplan, the program, program cache, and data memory are placed close together on the bottom side of the die. The data memory is on the right side (colored blue), and the program memory is in the left side (set memories are red and tag memories are green). The idea is that logic common to both data and program memory should be placed between them, and will be closed to both of the data and program memory cuts.

It is recommended to locate the ports that should interact with the DMSS logic (***edp*/*edap***) close to the data memory, and ports that should interact with PMSS logic (***epp***) close to the program memories.

In this floorplan, the port locations are set according to the following critical paths:

- **Orange**: epp
- **Green**: iop
- **Red**: sco/sci
- **Purple**: AXI Slave
- **Light green**: edap
- **Light blue**: AXI Master
- **Yellow**: edp

This floorplan corresponds to the reference configuration.

5.4 Running Synthesis

Before running synthesis, the general **setenv** file should be sourced by typing:

```
source <install dir> /scripts/setenv.csh
```

The synthesis of the CEVA-XM4 is performed in the **<install_dir>/backend/synthesis/dc/cevaxm4** directory.

5.4.1 DC Synthesis

There are several ways to run synthesis:

- **Option 1**

  Via DCT mode with a real floorplan as an input from the command line:

```
> dc_shell  -topographical -64 -x "set
SynthWithFloorplan  true" -f $BACKEND_ROOT_DIR
/common/scripts/synopsys/dc/dc.tcl | tee -i cevaxm4.log
```

  In this case, the user should provide a DEF file with the floorplan description created by the ICC, and set the **DEF_FILE** variable in the **project_setup.tcl** file.

- **Option 2**

  Via DCT with physical guidance (SPG mode) and a real floorplan as an input from the command line:

```
> dc_shell  -topographical -64 -x "set CompileWithSpg
true; set SynthWithFloorplan true" -f $BACKEND_ROOT_DIR
/common/scripts/synopsys/dc/dc.tcl | tee -i cevaxm4.log
```

  In this case, the user should provide a DEF file with the floorplan description created by the ICC, and set the **DEF_FILE** variable in the **project_setup.tcl** file.

Note: *The user should have a DC extension license to run in SPG mode.*

- **Option 3**

  Via DCT mode **without** a real floorplan from the command line:

```
> dc_shell  -topographical -64 -x "set
SynthWithFloorplan false" -f $BACKEND_ROOT_DIR
/common/scripts/synopsys/dc/dc.tcl   | tee -i
cevaxm4.log
```

In this case, the user should provide physical parameters for the floorplan description, and set the **FpUtilization** and **FpAspectRatio** parameters in the **project_setup.tcl** file. DCT parameters can be added by the user if required.

Note: *The recommended flow for synthesis is based on a two-stage process:*

1. *In the first stage, synthesis is done via DCT without a floorplan (**Option 2**).*
2. *A detailed floorplan is created by the ICC using this netlist, and then synthesis is done again via DCT based on this floorplan DEF file (Option 1).*

6. Equivalence Checking

Equivalence checking uses formal techniques to check the functional equivalence of two designs. The equivalence checking in this database is based on the Synopsys Formality tool, which practices static techniques and does not require simulation vectors. The Formality tool should have two inputs: a functionally correct, or "golden" design (called the reference design), and a modified version of the design (called the implementation).

Both the reference and the implementation can be either gate-level netlists or its RTL source. In this stage, the CEVA-XM4 netlist is checked against the CEVA-XM4 RTL.

6.1 Formality Directory Structure

Figure 6-1 shows the formality directory tree.

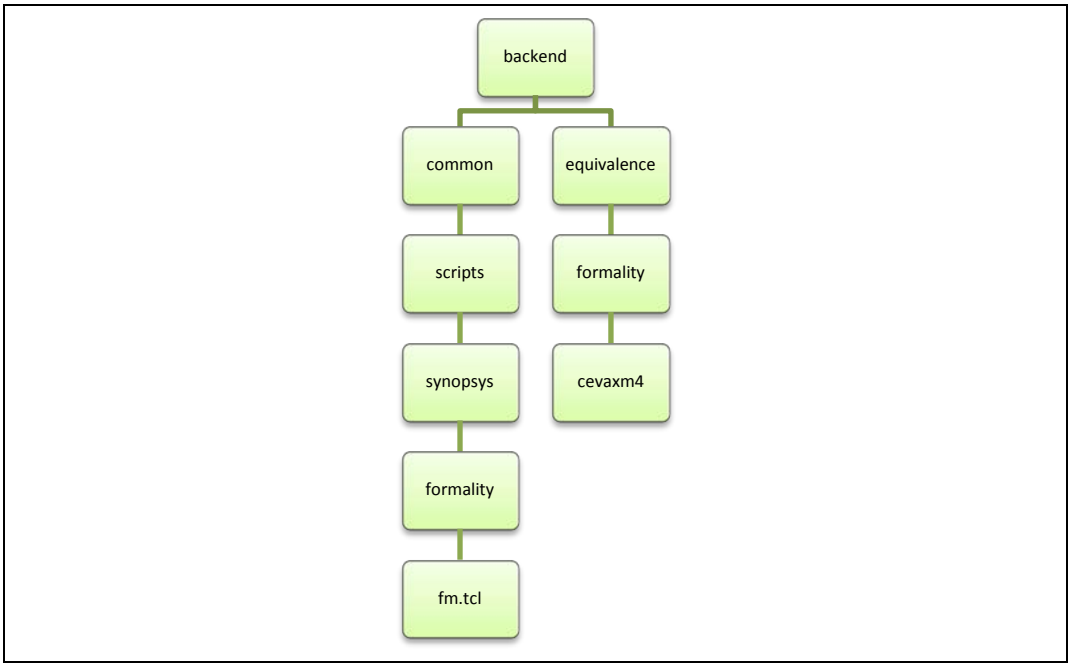

Figure 6-1: Formality Directory Tree

Table 6-1 describes the **common/** and **equivalence/** database directories.

Table 6-1: CEVA-XM4 common/ and equivalence/ Directories

Directory Name	Description
common/scripts/ synopsys/formality	Includes the script files for running Formality
equivalence/formality/ cevaxm4	Run directory for Formality. Contains the following subdirectories: • outputs: Files generated by Formality • reports: Match and verify reports generated by Formality • logs: Logs generated by Formality

6.1.1 Run Formality

The equivalence check should run after the synthesis is complete. Formality runs in the following stages:

1. The RTL files are compared to the pre-scan netlist.
2. The pre-scan netlist is compared to the post-scan netlist.

By default, the Formality script receives the netlist and SVF files from the **<install_dir>/backend/synthesis/dc/cevaxm4/outputs** synthesis run directory. If the user wants to take them from a different location, the following variables should be set in the **project_setup.tcl** file:

- **VERILOG_PRE_SCAN_NETLIST <netlist path>**
- **VERILOG_NETLIST_FILE <netlist path>**
- **PRE_SCAN_SVF_FILE <SVF file path>**
- **SVF_FILE <SVF file path>**
- **FM_LOCAL_SETTINGS_FILE** (exceptions for the Formality script such as **set_dont_verify_point** and so on)

Note: *The user can also update these variables by adding* ***-x "set VERILOG_NETLIST <netlist path>"*** *to the command line.*

Unless the file has already been run on the same terminal, the general **setenv** file should be sourced before running by typing:

```
source <install dir> /scripts/setenv.csh
```

As the flow progresses through the **place and route** stage, the user can check continuing formal equivalence by setting the **VERILOG_NETLIST_FILE** parameter to point to the latest netlist produced (for example, **cevaxm4_route.v**) and then re-running the equivalence check.

6.1.2 Invoke Formality

To invoke Formality, go to the **<install_dir>/backend/equivalence/formality /cevaxm4/** directory, and then type:

```
fm_shell -f
$BACKEND_ROOT_DIR/common/scripts/synopsys/formality/fm.t
cl| tee -i  cevaxm4_fm.log
```

6.1.3 Output Files

Formality generates the following output log files:

- **cevaxm4_fm.log**: This file is a replication of the standard output during the Formality execution.
- **formality.log**: This file contains verbose information not printed to the transcript.
- **fm_shell_command.log**: This file contains a history of the Formality shell commands run during the session.

In addition, Formality generates the following reports in the **reports** directory:

- **cevaxm4 _pre_scan_unmatched.rep**: A report of unmatched points
- **cevaxm4 _pre_scan_verify.rep**: The verify report
- **cevaxm4_unmatched.rep**: A report of unmatched points
- **cevaxm4_verify.rep**: The verify report

If Formality fails in the **match** stage, it does not continue to the **verify** stage, and exits the current session after saving. The saved session is located in the **backend/equivalence/formality/cevaxm4/logs/** directory.

If Formality fails in the **verify** stage, it exits the current session and saves it. The saved session is located in the **backend/equivalence/formality/logs/** directory.

Note: *The equivalence checking log reflects a reference design with a 100 MHz synthesis target frequency.*

6.1.4 Reference Log

The Formality reference log is located in the **<install_dir>/reference/backend/synopsys/28HM/logs/cevaxm4_ Formality.log** file.

7. Place and Route

The following sections describe the CEVA-XM4 Place and Route stages, based on the ICC.

This stage has the following input options:

- A Verilog netlist from the DC scandef file, and an optional DEF file
- A DDC file from the DC

The output is placed and routed to the Milkyway CEL and netlist.

The Place and Route stages have the following phases:

1. **Floorplan Creation**: Creating the MW library and floorplan cell

Note: *If the user uses the DC topographical flow with a floorplan, the created floorplan phase is based on the DEF used in the DC topographical stage.*

2. **Placement**: Placing the design and reordering the scan chains
3. **Clock Insertion**: Inserting the clock tree and routing the clock nets
4. **Route**: Routing the entire design

7.1 ICC Directory Structure

Figure 7-1 shows the structure of the ICC directories.

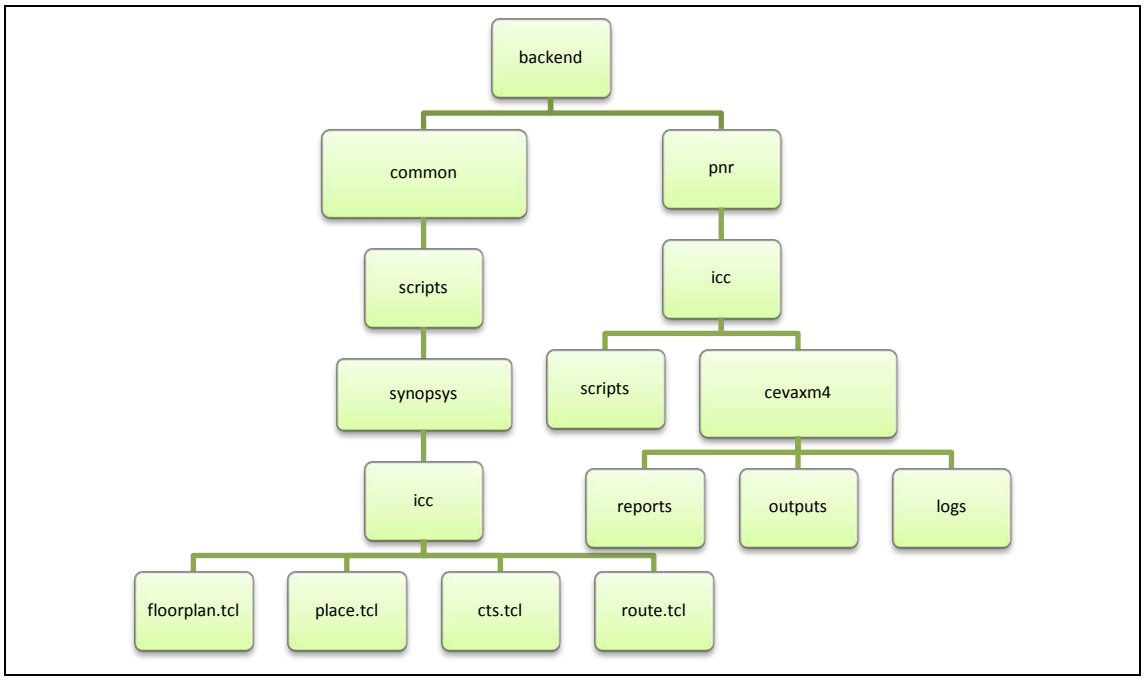

Figure 7-1: CEVA-XM4 ICC Directory Structure

Table 6-1 describes the **common/** and **pnr/** database directories.

Table 7-1: CEVA-XM4 common/ and pnr/ Directories

Directory Name	Description
common/scripts/synopsys/icc	The ICC general scripts
pnr/icc/scripts	The local ICC scripts (for example, blockages and macro locations)
pnr/icc/cevaxm4	The run directory. Contains the following subdirectories: • **reports**: Reports generated by the ICC • **outputs**: Verilog netlists generated by the ICC • **logs**: Log files generated by the ICC

7.1.1 Synopsys ICC Flow

Before running the flow, the general **setenv** file should be sourced (unless it has already been sourced on the same terminal) by typing:

```
source <install dir>/scripts/setenv.csh
```

The ICC should run after synthesis is complete and formal verification has been passed. The run directory for all above four phases is **<install_dir>/backend/pnr/icc/cevaxm4/**.

7.1.1.1 Floorplan Creation

7.1.1.1.1 ICC-SPG Flow

If the user runs an SPG flow, the DDC file should be read from the synthesis and not from the Verilog netlist. To do this, configure the **project_setup.tcl** file with the following:

- **set SpgFlow true**
- **set ICC_INIT_DESIGN_INPUT DDC**
- **set DEF_FILE <path to the same DEF file used during synthesis>**
- **set ICC_IN_DDC_FILE <path to the DDC file>**

7.1.1.1.2 Using a DEF File

By default, the floorplan script builds the floorplan using TCL scripts for placing macros, ports, blockages, and so on. To build a floorplan from a DEF file, the **set DEF_FILE <Path to DEF** file> TCL variable must be configured in the **project_setup.tcl** file.

To invoke the ICC, execute following command from the ICC run directory:

```
icc_shell –f
$BACKEND_ROOT_DIR/common/scripts/synopsys/icc/floorplan.
tcl | tee -i floorplan.log
```

7.1.1.1.3 Output Files

The script generates the following:

- A Milkyway (MW) library located in the **<install_dir>/backend/icc/cevaxm4/cevaxm4_lib** directory
- A floorplanned MW cell named **cevaxm4_floorplan**
- The following reports in the **reports** directory (which is created under the run directory):
  - **cevaxm4_floorplan.qor_snapshot.rpt**: create_qor_snapshot report
  - **cevaxm4_floorplan.sum**: report_design –physical report
  - **cevaxm4_floorplan.timing**: report_timing

7.1.1.2 Placement

To invoke the ICC, execute the following command from the ICC run directory:

```
icc_shell –f
$BACKEND_ROOT_DIR/common/scripts/synopsys/icc/place.tcl
| tee -i place.log
```

7.1.1.2.1 Output Files

The script generates the following:

- A placed MW cell named **cevaxm4_place**
- The following reports in the **reports** directory:
  - **cevaxm4_place.con**: Report constraints (all violations)
  - **cevaxm4_place.qor_snapshot.rpt**: qor reports
  - **cevaxm4_place.max.tim**: Timing report (maximum)
  - **cevaxm4_place.min.tim**: Timing report (minimum)
  - **cevaxm4_place.qor**: qor report

7.1.1.2.2 Placement Example

Figure 7-2 shows an example of placed cells.

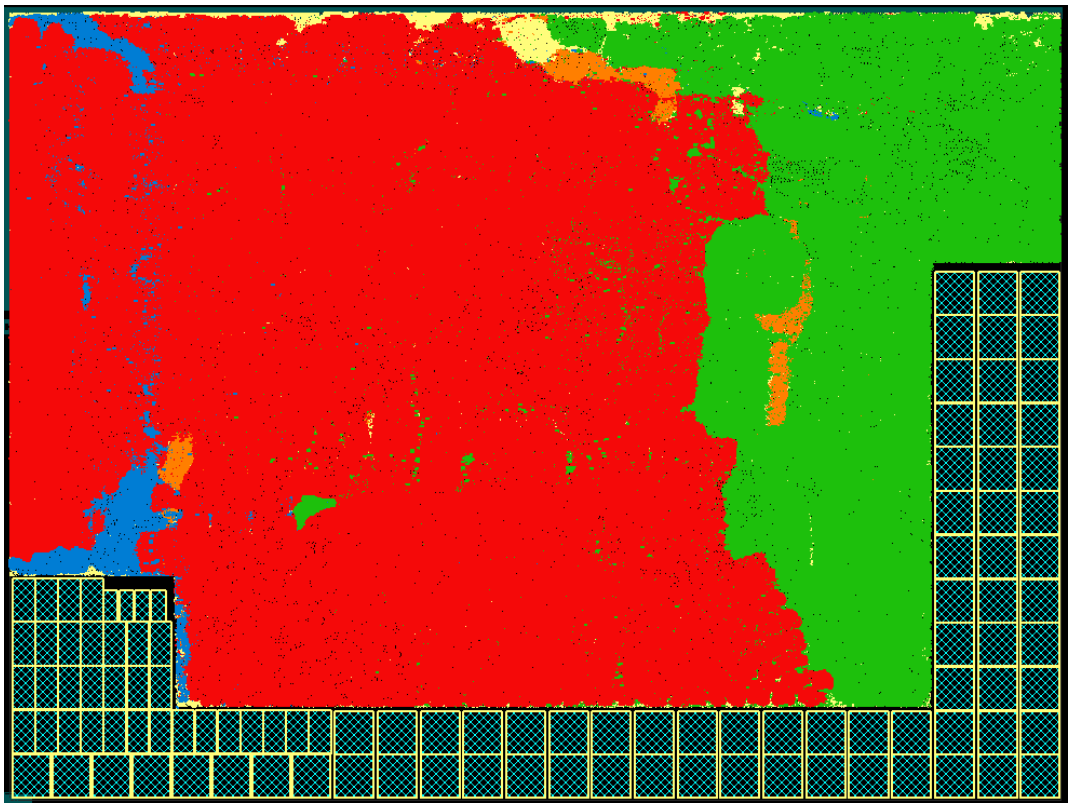

Figure 7-2: CEVA-XM4 Placement Example

The main highlighted groups are:

- **Yellow**: Top level
- **Red**: cevaxm4_core_top
- **Green**: cevaxm4_dmss
- **Blue**: cevaxm4_pmss
- **Orange**: cevaxm4_emulation

7.1.1.3 Clock Insertion

To invoke the ICC, execute the following command from the ICC run directory:

```
icc_shell –f
$BACKEND_ROOT_DIR/common/scripts/synopsys/icc/cts.tcl |
tee -i cts.log
```

7.1.1.3.1 Clock Gater Split

To split the clock gaters because of critical paths upon clock gater enable, execute the following command from the ICC run directory:

```
icc_shell -x "set ClockGateSplit true; set
ClockGateSlackMargin <slack margin>" -f
$BACKEND_ROOT_DIR /common/scripts/synopsys/icc/cts.tcl |
tee -i cts.log
```

The **ClockGateSplit** and **ClockGateSlackMargin** parameters can be set permanently in the **project_setup.tcl** file.

7.1.1.3.2 Output Files

The script generates the following:

- An MW cell named **cevaxm4_cts** with a synthesized and routed clock
- The following reports in the **reports** directory:
  - **cts_pre_rtd.***: Various reports about the synthesized clock before the clock routing stage
  - **cevaxm4_cts.clock_tree**: Clock tree global skew report
  - **cevaxm4_cts.clock_timing**: Clock tree local skew report
  - **cevaxm4_cts.max.tim**: Timing report (maximum)
  - **cevaxm4_cts.min.tim**: Timing report (minimum)
  - **cevaxm4_cts.qor**: qor report
  - **cevaxm4_cts.con**: Report constraints (all violations)
  - **cevaxm4_cts.qor_snapshot.rpt**: qor report

7.1.1.4 Route

To invoke the ICC, execute the following command from the ICC run directory:

```
icc_shell -f
$BACKEND_ROOT_DIR/common/scripts/synopsys/icc/route.tcl
| tee -i route.log
```

7.1.1.4.1 Output Files

The script generates the following:

- A routed MW cell named **cevaxm4_route**
- The following reports in the **reports** directory:
  - **cevaxm4_route.rep**: Report constraints (all violations)
  - **cevaxm4_route_opt.qor_snapshot.rpt**: qor reports
  - **cevaxm4_route_opt.max.tim**: Timing report (maximum)
  - **cevaxm4_route_opt.min.tim**: Timing report (minimum)
  - **cevaxm4_route_opt.clock_tree**: Clock tree report
  - **cevaxm4_route_opt.clock_timing**: Clock tree timing report

8. Extraction

This stage is used to extract the capacitance and resistance of the design. It is done using the Star-RCXT tool. Star-RCXT generates a SPEF file, which is a standard parasitic format for PrimeTime-SI.

8.1 star/ Directory Structure

Figure 8-1 shows the structure of the **star/** directories.

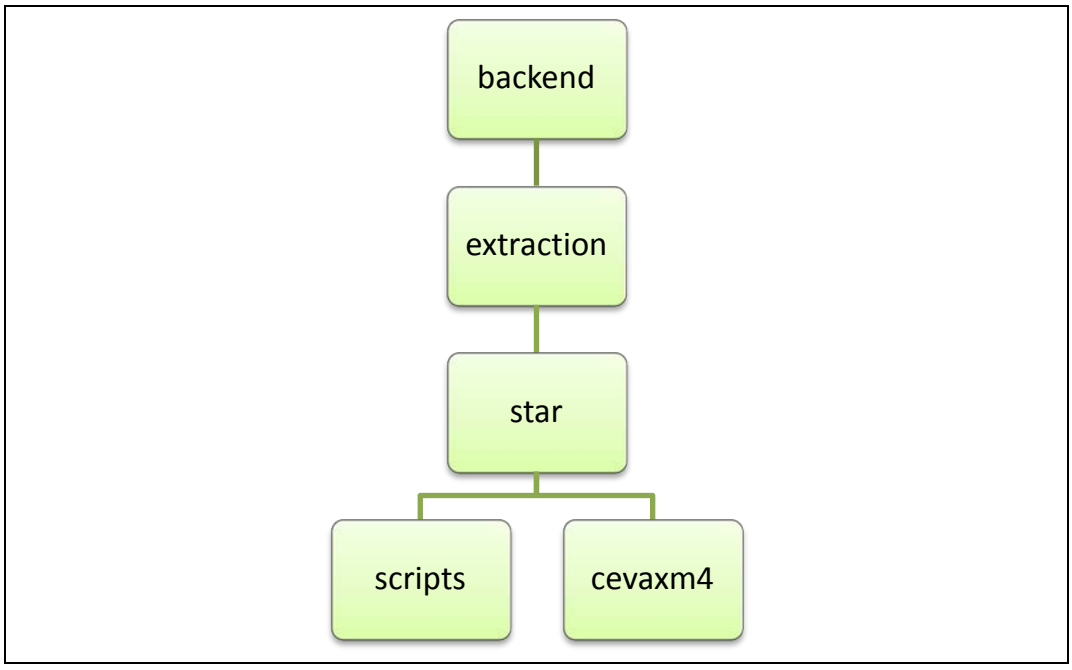

Figure 8-1: CEVA-XM4 star/ Directory Structure

Table 8-1 describes the **star/** directories.

Table 8-1: CEVA-XM4 star/ Directories

Directory Name	Description
extraction/star/scripts	The star scripts
extraction/star/cevaxm4	The work directory

8.2 Running Star-RCXT

Extraction is done from the **/backend/extraction/star/** directory.

8.2.1 Tool Setup

Before executing the script, the following parameters must be updated in the **cevaxm4_CWorst_28HM.cmd** script:

- **TCAD_GRD_FILE**: A nxtgrd file provided by the vendor
- **MAPPING_FILE**: A mapping file provided by the vendor
- **MILKYWAY_DATABASE**: The MW path to the route CEL

8.2.2 Invoke Star-RCXT

After the script is updated, invoke Star-RCXT by executing the following command from the **/backend/extraction** directory:

```
StarXtract -clean cevaXM4_CWorst_28HMULVT.cmd
```

Note: *This release supports only the worst-case operating condition.*

8.2.3 Output Files

The output is the **cevaxm4.spef** parasitic file.

9. Static Timing Analysis

Static Timing Analysis (STA) checks and verifies the timing of the CEVA-XM4. STA is based on the Synopsys PrimeTime-SI tool, and uses the post-layout netlist and SBPF/SPEF (Standard Parasitic Format for PrimeTime SI) as inputs. The STA flow outputs setup, hold, clock skew reports, and SDF.

9.1 PrimeTime Flow Description

PrimeTime requires a netlist, SBPF/SPEF, a vendor minimum and maximum library, and a constraints file, as shown in Figure 9-1.

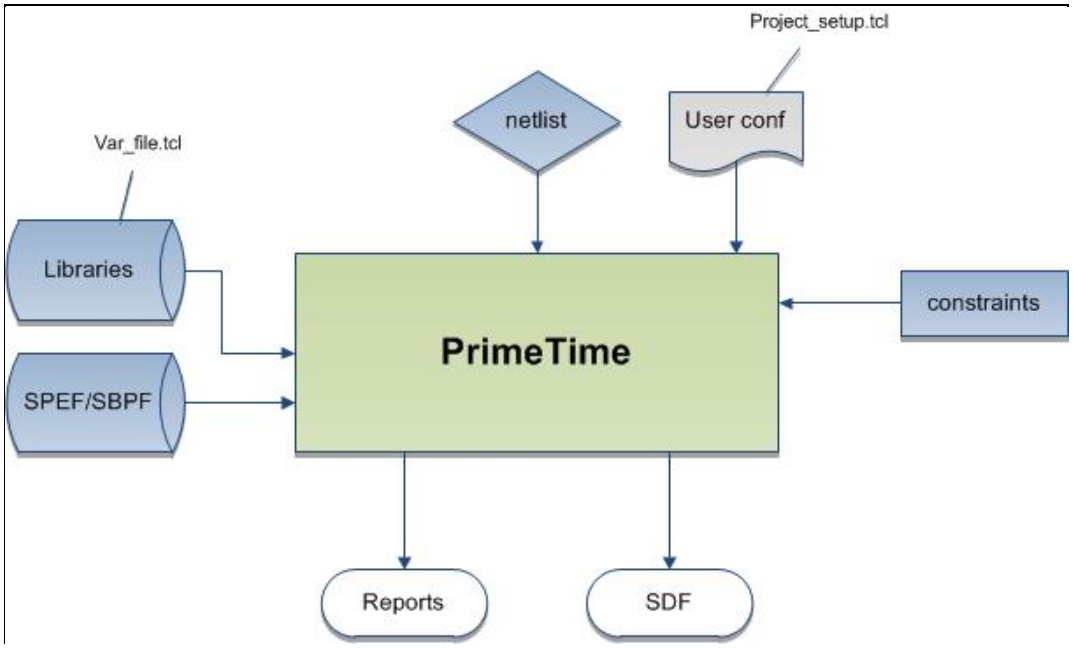

Figure 9-1: PrimeTime Flow Description

9.2 PrimeTime Directory Structure

Figure 9-2 shows the structure of the **sta/** and **common/** directories.

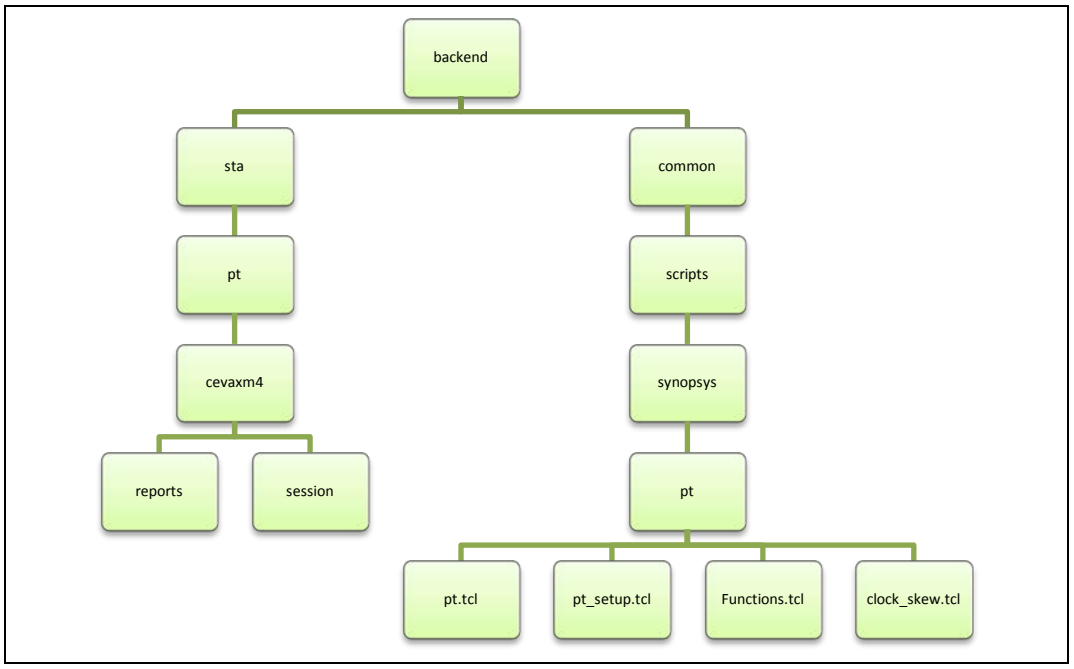

Figure 9-2: CEVA-XM4 sta/ and common/ Directory Structure

Table 9-1 describes the **sta/** and **common/** directories.

Table 9-1: CEVA-XM4 sta/ and common/ Directories

Directory Name	Description
sta/pt/cevaxm4	Work directory for timing analysis. Contains the following subdirectories: ● **reports**: PrimeTime reports directory ● **session**: Saved sessions
common/scripts/synopsys/pt	The scripts for PrimeTime run

9.3 Executing the STA Flow

9.3.1 PrimeTime Environment Setup

The timing analysis process uses scripts that are located in the **/backend/common/scripts/synopsys/pt/** directory. The following parameters should be updated in the **/backend/scripts/synopsys/pt/pt_setup.tcl** file:

- **Set RUN_MODE_TYPE func/test**: Determines whether to run in functional mode or test mode (default is **func**)
- **Set OPER_COND cworst/rcworst/cbest/rcbest/typical**: Determines which operation condition to use for the run (default is **cworst**)
- **Set SAVE_SESSION TRUE/FALSE**: Determines whether to save the session (default is **TRUE**)
- **Set WriteSdf TRUE/FALSE**: Determines whether to write SDT (default is **TRUE**)
- **Set EXTRACT_MODEL TRUE/FALSE**: Determines whether to create ETM (default is **FALSE**)
- **Set GROUP_PATH group_path_enabled/group_path_disabled**: Determines whether **group_path_enabled** is set; if so, the following groups are created (disabled by default) :
  - INPUT_2_REG
  - INPUT_2_OUTPUT
  - REG_2_OUTPUT
  - REG_2_REG
- **Set PRIMETIME_SI si_disabled/si_enabled**: Determines whether to run PrimeTime SI (default is **si_disabled**)
- **Set QUIT_SWITCH TRUE/FALSE**: Determines whether to quit PrimeTime when run is finished (default is **TRUE**)

Note: *The CEVA-XM4 constraints file should be reviewed as well; the existing files are for reference only.*

9.3.2 Invoke PrimeTime

Before running PrimeTime, the general **setenv** file should be sourced (unless it has already been run on the same terminal) by typing:

```
source <install_dir>/backend/scripts/setenv.csh
```

After you set up the environment (as described in Section 9.3.1), invoke PrimeTime by executing the following command from the **backend/sta/pt/cevaxm4/** directory:

```
pt_shell -f $BACKEND_ROOT_DIR/scripts/synopsys/pt/pt.tcl
| tee -i cevaxm4_pt.log
```

9.3.3 Output Files

The following reports are generated in the **reports** directory:

- **cevaxm4/reports/<operating condition>/<run number>/timing/report_timing.rep**
- **cevaxm4/reports/<operating condition>/<run number>/timing/report_constraint.rep**
- **cevaxm4/reports/<operating condition>/<run number>/timing/report_disable_timing.rep**
- **cevaxm4/reports/<operating condition>/<run number>/skew/ report_clock_skew.rep**
- **cevaxm4/reports/<operating condition>/<run number>/read_constraints.rep**
- **cevaxm4/reports/<operating condition>/<run number>/read_parasitics.rep**

If **set WRITE_SDF = TRUE**, then the **pt/sdf/<operating condition>/<tool_version>/cevaxm4.sdf** file is also generated.

Note: *Depending on the flags set by the user, other outputs might also be created.*

The following reference logs and reports are also created:

- **<install_dir>/reference/backend/synopsys/28HM/logs/ cevaxm4_ PT.log**
- **<install_dir>/reference/backend/synopsys/28HM/reports/ cevaxm4_ PT.rep**

10. TetraMax (ATPG)

TetraMax is an Automatic Test Pattern Generator (ATPG), and verifies the test coverage of the design. It is in the **/backend/tetramax/cevaxm4/** directory.

TetraMax runs after the design contains a clock tree and route (it can also run after the DC post-scan-chain stitching).

10.1 Non-Functional Flip Flops (NFFs)

To increase scanability over the memory logic, an additional MUX can be introduced into this logic. During test mode, when the **testmodep** indication is on, this MUX transfers data to/from the NFFs. For more details, see the *CEVA-XM4 Integration Reference Guide*.

10.2 Tool Setup

The following parameters must be defined in the **/backend/common/scripts/project_setup.tcl** file before running:

- **DCResultsDir**: The netlist location
- **VERILOG_NETLIST_FILE**: The netlist name

10.3 Invoke TetraMax

Before running TetraMax, the general **setenv** file should be sourced (unless it has already been run on the same terminal) by typing:

```
source <install_dir>/backend/scripts/setenv.csh
```

Invoke TetraMax by executing the following command:

```
tmax -tcl -shell
$BACKEND_ROOT_DIR/common/scripts/synopsys/tetramax/tm.tc
l | tee -i cevaxm4_tm.log
```

10.4 Output Files

The following files are generated:

- **cevaxm4_tm.log**: List of errors and warnings

Tip: *It is recommended that you open this file and ensure that there are no errors and warnings. You should also check the test coverage reached by the tool.*

- **cevaxm4_par.stil**: Test pattern generated by the tool

11. Backend Design Overview

CEVA-XM4 has a single-clock design. Two additional clocks are available when using the ETMR4. You should be familiar with the clock structure in the design, as well as understand the SCAN and the Reset methodologies.

11.1 Clock

The main clock of CEVA-XM4 is **ceva_free_clk**. In addition, the following are the other watchdog clocks:

- **ceva_sys_wdog_clk**
- **ceva_epp_wdog_clk**
- **ceva_edp_wdog_clk**
- **ceva_iop_wdog_clk**
- **ceva_amX_wdog_clk**

All of these clocks are fully synchronous.

There is no logic on the clocks, except for clock gaters. Up to two levels of clock gaters exist on **ceva_free_clk**.

In case of real trace configuration using internal ETMR4, there are two additional clocks:

- **etm_atb_clk:** Quarter frequency of **ceva_free_clk**
- **apb_clk_dbg** : Quarter frequency of **ceva_free_clk**

11.2 DFT

Scan chain stitching is done in the synthesis stage, and scan reordering is done in the Place stage in the ICC.

The user can use as many chains as needed, and should be aware of the following signals when running in scan mode:

- The **scanen** input is connected (post-scan chain stitching) to the scan enable pin in all of the FFs.
- The **tst_gatedclock** input is connected to the scan enable pin in all of the clock gaters **except** for the clock gaters that control the memory clock signals.

  This is done to ensure that, during scan testing, clock gaters can be controlled directly from the input pin. This makes ATPG much simpler and increases test coverage.

- The **tst_mem_gatedclock** input is connected to the scan enable pin in all of the clock gaters that control the memory clock signals.

  This is done to enable bist mode and ensure that the clocks are not stopped in this mode.
- If the design is in test mode, the **testmodep** input should be set to **1**.

  This signal ensures that the reset arrives directly from an input port and not from the reset synchronizers.

During the reference flow, these signals are considered to be **static** (with their value assigned to 1'b0, using **set_case_analysis**, based on its functional mode value).

11.3 Reset

Because the CEVA-XM4 receives asynchronous resets, the user must check the recovery/removal timing in PrimeTime, as well as make sure that the reset endpoints (that is, register reset pins) are not susceptible to glitches.

It is recommended to mark the reset nets as ideal nets during synthesis, and implement them as high fanout nets starting from the placement stage.

Because some of the registers generating the reset signals have high fanout, it might be challenging to meet the timing for reset de-assertion at high frequencies. Do the following to help reset the timing closure:

1. Ensure that the reset synchronizers are placed at the "center of gravity" of their fanout (this is usually about the center of the core).

   To look at the location of a reset register's fanout, type the following command in the Synopsys tools:

```
change_selection [all_fanout -flat -endpoints_only –from
<reset_register_out_pin> ]
```

2. Because the reset registers have a sample before them, a considerable early skew can be used on their clock. This will give the reset signal more time to propagate to the end registers.

11.4 JTAG

For details about the JTAG interface, see the *CEVA-XM4 Integration Reference Guide*.

12. Glossary

Table 12-1 defines the acronyms used in this document.

Table 12-1: Acronyms

Term	Definition
CTS	Clock Tree Synthesis
DACU	Data Access Control Unit
DCG	Design Compiler Graphical
DCT	Design Compiler Topographical
DFT	Design For Test
DMSS	Data Memory Sub System
DSP	Digital Signal Processor
ECC	Error Check and Correct
ICC	IC Compiler
MUX	Multiplexer
NFF	Non-Functional Flip Flops
OCEM	On-Chip Emulation Module
PMSS	Program Memory Sub System
PSU	Power Scaling Unit
QMAN	Queue Manager
RTT	Real-Time Trace
SBPF	Standard Parasitic Exchange Format
SDF	Standard Delay Format
SDT	Software Development Tools
SPEF	Standard Parasitic Exchange Format
SPG	Synopsys Physical Guidance
SPU	Scalar Processing Unit
TCM	Tightly Coupled Memory
VPU	Vector Processing Unit