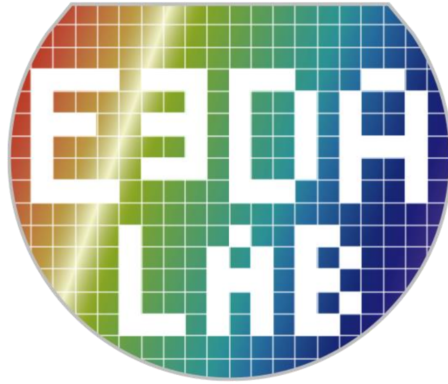

2.5D In-Context Flow Guide

Version 1.0

Energy-Efficient Electronics and Design Automation (E3DA) Lab

Department of Computer Science and Computer Engineering (CSCE)

University of Arkansas

Disclaimer of Warranty

The software is provided “AS IS”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

Table of Contents

Package Overview 4

Innovus Usage 4

Design Flow 5

  Overall Flow 5

  Top-Level Plan Generation 5

  Top-Level Implementation 8

    In-Context Partition for Core-Chiplet 8

    In-Context Partition for Memory-Chiplet 8

  Chiplet Physical Design 9

Scripts Usage 9

  gen_pins.py 9

  gen_rdl_plan.py 9

Related Publications 10

Authors 10

Package Overview

This package implements the flow used to design the experimental two-chiplet heterogeneous design presented in the paper [1]. The package contains a design setup for Cadence Innovus v20.14 and python scripts written in Python-2.7. Within the package, there are several sub-directories and files. Table 1 briefly describes the contents of the subdirectories.

Table 1: Package organization

Design-Flow	Contents
├── **implement**	Contains workspaces for the top-level package, in-context partitions, and chiplets sub-design workspaces
├── **lib**	Contains standard cells, RAM, and ROM macros, and technology files for two modified versions of Nangate45nm PDK.
├── **netlists**	Contains Verilog netlists of the system
└── **planning**	Contains the RDL-planner tool and configuration files for generating RDL plans. Chiplet macros are also available here.

Innovus Usage

This section describes the steps to start Innovus and load any design into it. The in-context flow described here deals with several Innovus workspaces, which requires loading designs using a pre-defined **`.globals`** file or using the **`source`** command. To load a design using a pre-defined **`.globals`** file follow the steps mentioned below.

1. Start Innovus in the **`innovus`** sub-directory within a workspace. Though the starting directory doesn't matter, this way the workspace remains clean.
2. Within the Innovus prompt, type "**`cd ..`**" to come back to the top-level directory and follow the steps shows in Fig. 1 to load the design settings using the **`cmsdk_mcu.globals`** file.

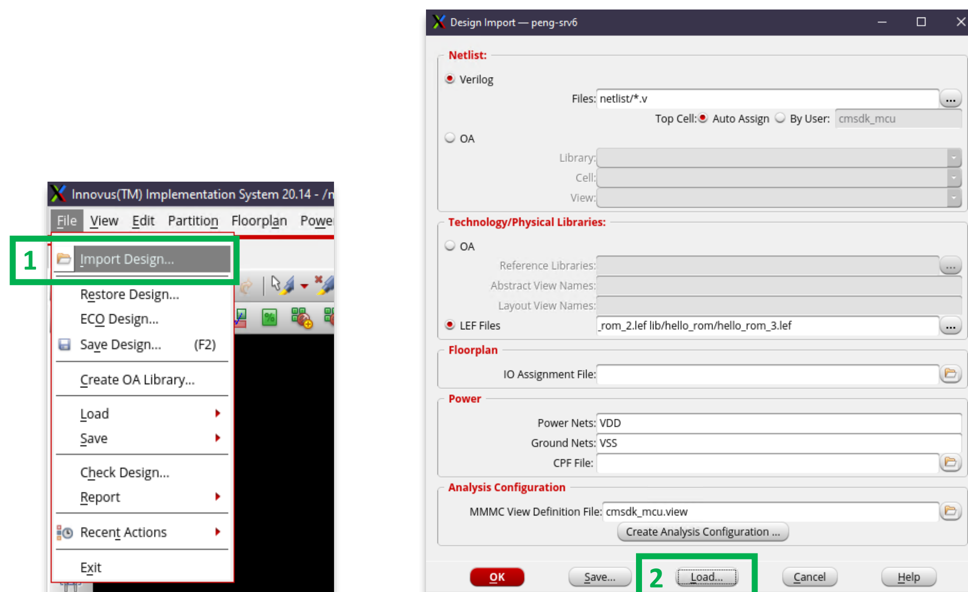

Fig. 1: Loading the design using the **`.globals`** file

3. After loading the settings, the design can be loaded hitting the *OK* button in the *Design Import* form.

In all Innovus workspaces, the design state after each step is saved in the **saved_steps_enc** directory by the script as *native Innovus DB format*. These saved designs can be loaded into Innovus using the **source** command as shown below. Note that if a design is already loaded, it must be unloaded using the **freeDesign** command.

```
innovus 2> source saved_steps_enc/pins_placed_D1.enc
```

Design Flow

Overall Flow

This in-context flow is composed of several steps, where the designer needs to switch between configuration files, scripts, and tools to design, analyze, optimize different parts of the system. The overall flow can be described in the following steps,

Top-level plan generation: This step generates the top-level plan of the system. It involves deciding the package floorplan, chiplet pin configurations, and package routing. Chiplet macros are also generated here to be used as a black-box macros in Innovus.

Top-level implementation: This step implements the top-level plan and creates sub-designs for chiplets and the package in Innovus. It uses the settings and scripts generated in the previous step to implement the package design. In-context partitions are created for chiplets in this step. These in-context partitions are used to perform in-context assembly and extraction.

Initial iteration: This step involves designing chiplets based on the initial estimations and budgets. It uses different settings compared to all later iterations. All chiplets are designed using Innovus in their own sub-directories, specifying the top-level budgets as design constraints. After the physical design in Innovus, all chiplets are assembled with their top-level design to generate files for in-context extraction. After extraction, parasitic netlists (SPEFs) are stitched within PrimeTime and STA is performed to generate timing contexts for the next iteration.

Subsequent iterations: The timing contexts generated by PrimeTime in the initial iteration are used to re-implement the chiplets. After re-implementation of chiplets, they are again assembled with their top-level designs for extraction. STA using PrimeTime is again performed after stitching the parasitics, as in the initial iteration. Several such iterations can be performed until there is no more performance improvement or the design goal is met.

The experimental design used here consists of two chiplets: core-chiplet and memory-chiplet. The core-chiplet contains 8KB of memory and all logic blocks. It is implemented in 7M3R version of the Nangate 45 nm technology. The memory-chiplet contains only 8KB of memory and is implemented using 6M3R version of Nangate 45 nm technology.

Top-Level Plan Generation

The top-level plan is generated in the **planning** sub-directory under the *top-level directory* (Design-Flow). The package floorplan and chiplet dimensions need to be determined first. These

parameters are determined based on design needs and the designer's experience. In this experimental design, following settings are used.

Table 1: Package and chiplet dimensions

Package dimensions (w × h)	1300 µm × 1150 µm
Core-chiplet dimensions (w × h)	390 µm × 590 µm
Memory-chiplet dimensions (w × h)	350 µm × 470 µm

Based on these settings, suitable floorplans for the package and chiplets are determined. Fig. 2 shows the top-level floorplan of the package and chiplets.

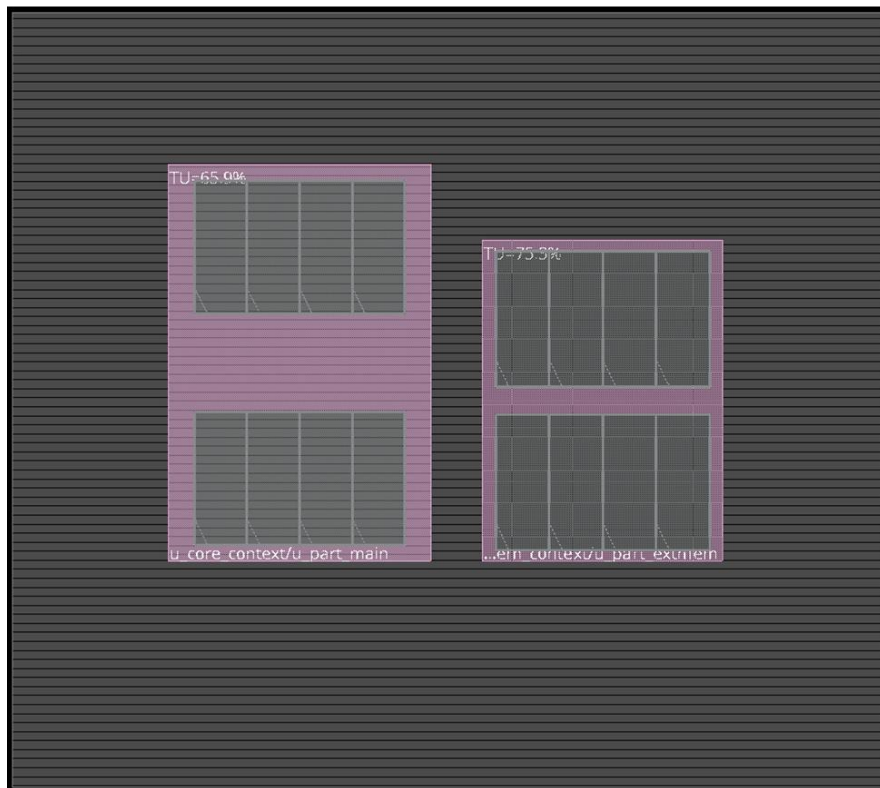

Fig. 2: Package and chiplet floorplans

The next step is to use the RDL-planning tool to generate chiplet pin assignments and package routes. The RDL-planning tool requires a configuration file to generate the scripts for top-level plan. File 1 on page 7 shows the settings used in this design. Lines 1-8 defines the routing pitch, location of chiplets, layer and via names used in Innovus layer map, size of chiplet pins, and the net-slack mapping file obtained from the gate-level netlist. Lines 10-14 define the planning mode, chiplet context names, and the cut-line location. The mode must be **`inContextHierPart`** for this flow.

Lines 17-34 define the pin configuration of the chiplets. These lines define the pin grid array, pin pitch as a multiplier of the **`pitch_unit`** defined in line 2, how many layers to use for fan-out, a

prefix used with the pin names in the net-slack map, and location to the CSV file specifying the pre-assigned signals. The required CSV files for this design can be found in the same directory as the File 1.

File 1: planning/rdl_plan/plan_D1/RDL_plan_settings.conf

```
1  # Layout configuration
2  pitch_unit = 20                          # micron
3  chiplet_origin = (663.065, 355.2)  # micron
4  chiplet_distance = 110.25          # micron
5  layer_names = ['rdl1u','rdl2u','rdl3u']
6  via_names = ['R2U_R1U_via', 'R3U_R2U_via']
7  pin_size = (10.0, 10.0)          # (width, height)
8  net_slack_csv = "pkg_wire_slacks.csv"
9  
10 # In-Context Specific settings
11 mode = 'inContextHierPart'
12 Lchip_context_name = 'CoreContext'
13 Rchip_context_name = 'MemContext'
14 cut_line = (670,None)   # cut-line is a vertical line passing through (670,0)
15                         # used in package routing script generation
16 
17 # Left chiplet configurations
18 Lchip_name = "PARTMAIN"
19 Lchip_row_cnt = 15
20 Lchip_col_cnt = 10
21 Lchip_pin_row_pitch = 2  # number of tracks
22 Lchip_pin_col_pitch = 2  # number of tracks
23 Lchip_layer_cnt = 3
24 Lchip_pin_prefix = 'u_core_context/' # prefix with pin name in net_slack_csv
25 
26 # Right chiplet configurations
27 Rchip_name = "PARTEXTMEM"
28 Rchip_row_cnt = 12
29 Rchip_col_cnt = 9
30 Rchip_pin_row_pitch = 2  # number of tracks
31 Rchip_pin_col_pitch = 2  # number of tracks
32 Rchip_layer_cnt = 3
33 Rchip_pin_prefix = 'u_mem_context/' # prefix with pin name in net_slack_csv
34 Rchip_pin_name_csv = 'pin_conf_PARTEXTMEM.csv'      # for pre-placed pins
35 
```

To generate the RDL-planning scripts and pin layouts of chiplets, run the following command while being in the directory **planning/rdl_plan/plan_D1**.

```
$ python gen_rdl_plan.py RDL_plan_settings.conf
```

The outputs are generated in the same directory. These outputs include CSV files for chiplet signal assignment, Tcl scripts for pin assignment of in-context partitions, and a Tcl script for package routing. The CSV file for the core-chiplet contains some empty cells marked as *None*. These cells need to be edited to specify the system external pins.

Another part of top-level plan generation step is to create partition-LEFs for chiplets to be used in the implementation step. These LEF macros can be directly written by the designer or generated using a custom script. The partition-LEF of chiplets are available in **`planning/chipletMacros`**.

Top-Level Implementation

The top-level implementation step is carried-out in the **`implement/topPackage`** directory. This step utilizes the scripts generated in the previous step to implement the top-level plan in Innovus. In this step an in-context partition is created for each chiplet for in-context assembly and extraction.

In-Context Partition for Core-Chiplet

To implement the in-context partition for the core-chiplet, enter the directory **`implement/topPackage/coreContext`**, start an Innovus session, and load the design using the **`cmsdk_mcu.globals`** file. The package-level design is implemented by sourcing the **`scripts/full_flow_D1.tcl`** script in Innovus. It is a good idea to keep a copy of the **`saved_steps_enc, PTN_D1`**, and **`outputs`** directory before running the script.

```
innovus 2> source scripts/full_flow_D1.tcl
```

The Tcl script has Innovus commands to perform the following steps.

1. Set the package dimensions.
2. Connect the *VDD* and *VSS* pins of the cells to power (*VDD*) and ground (*VSS*) nets.
3. Place the chiplet macros according to the floorplan.
4. Source the package pin-placement script.
5. Source the package routing script generated by the RDL-planner to implement the package routing.
6. Source the routing scripts written by the author for routing package external pins.
7. Define in-context partition blocks around the chiplet macros.
8. Source scripts to assign pins of the in-context partitions. This is necessary to push down the routes within the partitions.
9. Perform budget extraction and push down in-context partitions.

The design state after each step is saved in the **`saved_steps_enc`** directory. This script generates a partition directory named **`PTN_D1/CoreContext`**. The in-context partition can be loaded in Innovus by running the following command in Innovus prompt.

```
innovus 2> restoreDesign . CoreContext
```

In-Context Partition for Memory-Chiplet

Similarly, to implement the in-context partition for the memory-chiplet, enter the directory **`implement/topPackage/memContext`**, start an Innovus session, and load the design using the **`cmsdk_mcu.globals`** file. The package-level design is implemented by sourcing the **`scripts/full_flow_D1.tcl`** script in Innovus. It is a good idea to keep a copy of the **`saved_steps_enc, PTN_D1`**, and **`outputs`** directory before running the script.

```
innovus 2> source scripts/full_flow_D1.tcl
```

This script performs the same steps as in the core-chiplet. This generates a partition directory named **PTN_D1**. This directory contains the in-context partition for the memory-chiplet as well as the top-level package design.

Chiplet Physical Design

The physical design of chiplets in the initial as well as all later iterations are performed within the **implement** directory. There are three sub-directories under this directory. Among these three, following two directories contain the workspace for physical design of chiplet.

- **coreChiplet:** It contains the sub-design for the core-chiplet.
- **memChiplet:** It contains the sub-design for the memory-chiplet.

These directories are empty in this package because the chiplets contain proprietary contents protected by non-disclosure agreement (NDA).

Scripts Usage

This section briefly describes the usage of the scripts used in this flow written by the author. The python scripts are written for Python-2.7.

gen_pins.py

This script is located at **pin_confs/gen_pins.py**. This is the simplest script without any options and configuration file. This script generates a script for Innovus, which is used for package pin placement. The variables like *die_width*, *die_height*, etc. at the top of the script are set up based on the design included in the package. To use the script, just invoke it using the Python interpreter in your shell.

```
$ python gen_pins.py
```

gen_rdl_plan.py

This script is located at **planning/rdl_plan/plan_D1/gen_rdl_plan.py**. This script implements the RDL-planning algorithm. This script takes a single configuration file as the command line argument. This configuration file specifies other files containing design-related information like timing slacks and pre-placed pins. An example RDL-plan configuration file is located at **planning/rdl_plan/plan_D1/RDL_plan_settings.conf**. To use the script, invoke it using the Python interpreter in your shell specifying the configuration file as the argument.

```
$ python gen_rdl_plan.py <configuration.conf>
```

Related Publications

This section contains a list of all the current publications related to the 2.5D in-context design flow as of the date of this document.

[1] MD Arafat Kabir, Dusan Petranovic, and Yarui Peng, "Coupling Extraction and Optimization for Heterogeneous 2.5D Chiplet-Package Co-Design", in *Proc. International Conference on Computer-Aided Design*, pp. 1–8, Nov 2020.

[2] MD Arafat Kabir, Weishiun Hung, Tsung-Yi Ho, and Yarui Peng, "Holistic and In-Context Design Flow for 2.5D Chiplet-Package Interaction Co-Optimization", in *Proc. International Symposium on VLSI Design, Automation and Test*, pp. 1–4, May 2021.

[3] MD Arafat Kabir, Dusan Petranovic, and Yarui Peng, "A Scalable In-Context Design and Extraction Flow for Heterogeneous 2.5D Chiplet-Package Co-Optimization", in *Proc. IEEE Conference on Electrical Performance of Electronic Packaging and Systems*, pp. 1-3, Oct 2021.

Authors

This flow is still under development. A brief introduction to the authors is as follows.

- **MD Arafat Kabir**
  Ph.D. Student
  Computer Science and Computer Engineering Department
  University of Arkansas, Fayetteville, AR, USA
  Email: makabir@uark.edu

- **Dr. Yarui Peng**
  Assistant Professor
  Computer Science and Computer Engineering Department
  University of Arkansas, Fayetteville, AR, USA
  Phone: 479-575-6043
  Email: yrpeng@uark.edu