Cheetah: Innovation of Design System Infrastructure

Divya Ramarao
Design Infrastructure
Intel India Pvt. Limited
Bangalore, India
divya.ramarao@intel.com

Mahesh Deshpande
Design Infrastructure
Intel India Pvt. Limited
Bangalore, India
mahesh.deshpande@intel.com

Prateeksha Keshari
Design Infrastructure
Intel India Pvt. Limited
Bangalore, India
prateeksha.keshari@intel.com

Susmita Pal
Design Infrastructure
Intel India Pvt. Limited
Bangalore, India
susmita.pal@intel.com

***Abstract*— As market demands like any other industry, Intel also strive to bolster product execution models targeted at all segments of Intel to enable remarkable reduction in base and derivative product development time. The Cheetah Design System is one of the initiatives that aimed to improve the efficiency and productivity of the design execution model and time to market. This paper discusses an overview of the Cheetah Design System Infrastructure (aka Backbone Infrastructure) architecture and the design principles being used to make the Cheetah Design System as a bare-metal design system that are important innovations to align with the Intel goal. The paper also reviews the Backbone Infrastructure components developed and their seamless integration with different design workflows from system level to tape-in. The principles adapted in this design systems are Layered Approach, Flows interaction through central APIs, Configurability through metadata, one implementation per functionality, Design workflow and technology agnostic and Abstraction to allow multiple implementations**

Keywords*— *Bare-metal, Cheetah Design System Infrastructure, Backbone Infrastructure, Metadata, Artifacts, Workarea

I. INTRODUCTION

Intel is emerging into new market segments which are very dynamic and require quick development cycle. we need to be competitive enough in order to meet the product's timeline. This demands a design system which allows various modular design processes (e.g. System Level, Virtual Prototyping, Front End RTL Design, R2G, Analog & Mixed Signal, Chip Package Board Co-Design, Post Silicon and Firmware) to integrate to the system easily in order to support a variety of Intel products (e.g. IP development, SoC design, special applications, etc.). Besides, another success criteria of a design system is the scalability of infrastructure e.g. to satisfy high compute and data demands.

Considering the above facts Cheetah design system shall be lightweight or more known as "bare-metal" in use of vendor tools so that we can enjoy benefit directly from all types of capabilities and new features released with a new tool version. In addition to that, this design system shall utilize internal and external process technology where it will give more flexibility to a business unit to meet custom requirements. It also must be able to ramp design teams quickly whether it is internal or external designers on existing and new projects to adapt with rapid change business environment due to various reasons such as shifting project priorities, new customers etc. To enable re-iteration across design steps and traceability of data paths, a design system needs a cohesive design data management as well. Re-iteration across all or a subset of design steps allows an overlap of design work which will support a shift left of development while traceability is important to address the market needs.

To achieve all the above success criteria, it's important to have a stable and reusable base layer called backbone infrastructure which is then is common to all tool flows of Cheetah Design System. Backbone infrastructure has been designed to provide a flexible and configurable environment which can easily be adapted to accommodate different project needs. It also allows a project to construct their configuration to fit their business execution model, e.g. IP development might require a consumer oriented setup while SoC design is towards product centric.

Cheetah Backbone Infrastructure consists of following components;

1. An Enter Project Scope component to configure and setup a user environment in which a designer can execute the workflows and tools;
2. A Work Flow Script component to provide a consistent work flow anatomy such as populate, prepare, run, release and archive steps;
3. A Point Tool Executable component to setup a runtime environment for vendor tools;
4. A Job Management component to submit jobs into the compute farm;
5. A Project Configuration component to create metadata (XML based configuration files);
6. A Design Project Management component to support indicator reporting, milestone closure and signoff;
7. A Design Data Management component to deal with source management systems;

Please note that Design Data Management and Design Project Management components are out of scope for this paper.

II. CHEETAH DESIGN SYSTEM INFRASTRUCTURE

An overview of the Cheetah design system's infrastructure is provided below.

A. *Overview*

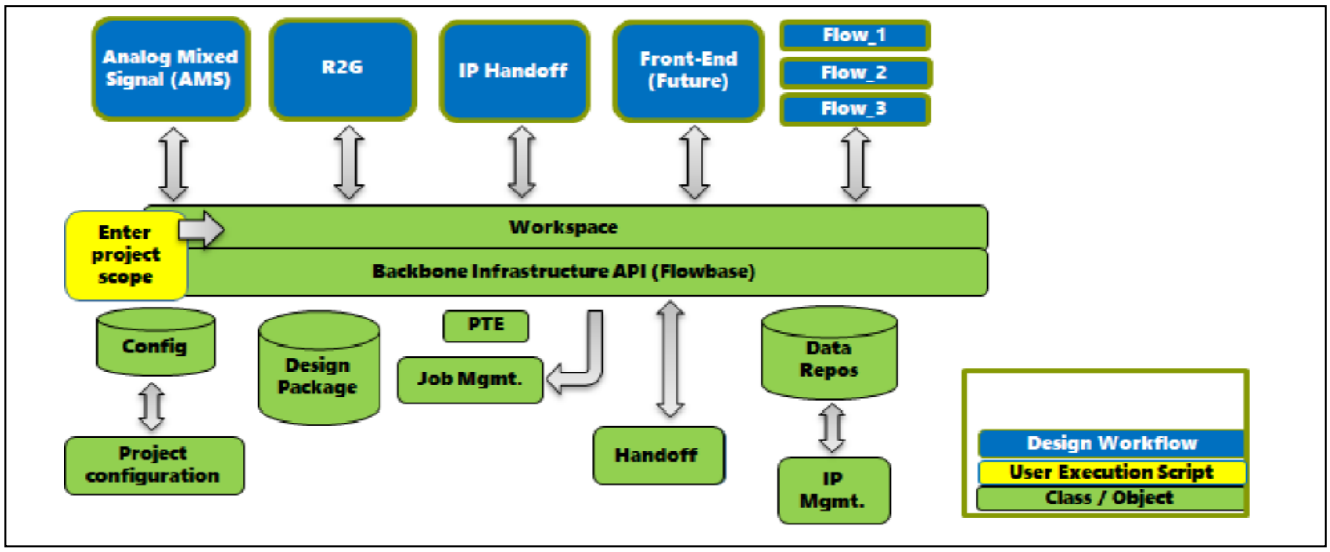

Figure 1.0: Cheetah Design System Infrastructure connects environment and flows

Current Intel design systems are highly optimized for specific products which ultimately increases the complexity to make changes to the design system. This can negatively impact the velocity of our execution. Leveraging past learnings, the Cheetah Design System Infrastructure is designed to be lightweight and scalable which connects the environment and flows to the holistic workspace. As a result, we have chosen a set of principles to follow during architecture design stage. As shown in Figure 1.0 above, the Cheetah Design System Infrastructure uses a layered approach with standard generic interfaces and APIs that are provided by Backbone Infrastructure software using an object-oriented methodology to allow for modularity and exchangeability of solutions. The set of APIs enables support design workflows to have a consistent access to the underlying IT infrastructure and design system components such as Design Package, Point-Tool-Executables (PTE), Configuration, Data Repos, etc.(in detail will be discussed in section II)

The Backbone Infrastructure is the entry point to a design project on a UNIX platform and handles all data and configuration management. The Backbone Infrastructure is independent of a functional design workflow or technology. By construction, this principle avoids conflicts among design workflows. A Design Engineer (DE) on-boards to a design project through "**Enter Project Scope**" (EPS) script where it sets up the workspace, starts a new shell and sets the minimum required environment based on the project configuration. After that the DE will be able to start execution of design workflows.

A **design workflow** (WFS) uses the backbone and its APIs to communicate with the infrastructure. It need not implement any infrastructure aspects such as access **configuration items** (configuration management) nor shall it handle command line arguments, access source control systems, **dispatch compute jobs** (Job Management), accessing of **technology libraries or packages** (Design Package), and execution of **3rd party tools or in-house tools** (Point Tool Execution) or data handoff between flows and other consumers etc. This principle allows us to address common issues in a consistent way as only one functionality exists for one implementation. The API abstraction layer also enables plug-in and plug-out of various infrastructure components, e.g., introducing a new source control methodology (Design Data management) or operating system such as SUSE LINUX 12. Cheetah uses metadata stored in an XML format to describe all artifacts commonly used or shared across design workflows— flows, collaterals, exchanged files, archives, etc. This metadata is stored together with the corresponding artifacts. The Backbone Infrastructure is built using a generic interface through the "Configuration management" component to allow a workflow to deal with all kind of artifacts. Lastly, the Cheetah flow execution path has no dynamic dependencies except for a few well-defined exceptions. This principle provides robustness in execution of a workflow and reproducibility of its results.

B. *Enter Project Scope (EPS)*

EPS is a process of onboarding a user to the Cheetah Design System workarea (landing into specified disk space) with the defined contour of data and tool configurations. EPS can be operated in three modes; interactive, command line or batch. Interactive mode prompts the user to select only the hierarchical project configuration, and workarea configuration definition (repositories to populate). It then auto-selects the rest of the required data selection from the configuration if present. If the data is not present, it will prompt the user to enter the required data. Command line mode expects all the selections to be present in the command line and enters to the new shell otherwise it will error out. Batch mode is similar to command line mode except it executes the command in the project shell and exits the shell once completed.

Once all the required data is obtained EPS starts the process of onboarding to the workarea. It does this by washing the required UNIX groups into the new shell, merging all the

hierarchical configurations files, creating a workarea on a predefined disk if it does not already exist. It then sets up minimum environment variables, loads the infrastructure related tools and environment, sets the initial license(s), sets the aliases and finally enters to the workarea for the Work Flow Script (WFS) execution.

Example structure of the project configuration:

<cheetah-cfg-root>/Domain/BU/cthconfig.xml
/cthconfig.iind.xml
/Project/EG1/cthconfig.xml
/Stepping/A0/cthconfig.xml
/Project/EG2/cthconfig.xml
/Stepping/A0/cthconfig.xml

The schema also allows for a 'table of contents' section which houses all classifications used in the XML manifest. This feature enables a Work Flow Script (WFS) developer to lookup valid classifications and construct the correct Design Package query that fetches either a design view of a library, or a technology node data from the PDK.

As the schema is designed to be modular, generating a new XML instance is greatly simplified. A Perl function is provided to associate a classification category to a file or directory metadata. This process repeats to create other set of metadata that makes a complete PDK or Library component. At this juncture, a process developer may decide to use the function provided to convert this component into physical

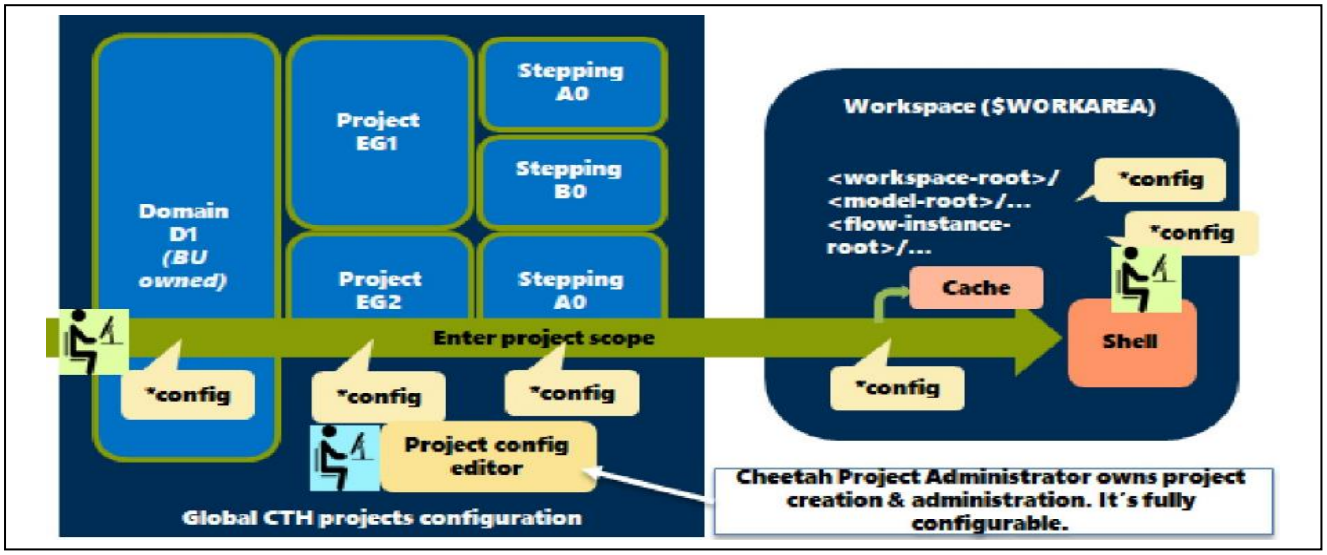

Figure 2.0: EPS on-boarding to user work area

C. *Design Package*

The Design Package consists of two major components, technology libraries and the Product Design Kit (PDK). These components use XML to describe characteristics and metadata of its own in order to ensure simplicity and flexibility. An example of this schema is shown in Figure 3.0. The major feature of this schema is to strictly enforce usage of a single XML element category, having only two attributes— name and value, to describe a classification it represents. The simplicity of this element allows the same category structure to nest within itself as much as needed, creating a complete set of classifications associated to the metadata file or directory.

```
<xs:element name="category">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded" minOccurs="0">
      <xs:element name="file" type="xs:string"/>
      <xs:element name="dir" type="xs:string"/>
      <xs:element ref="category"/>
    </xs:choice>
    <xs:attribute type="xs:string" name="name"
use="optional"/>
    <xs:attribute type="xs:string" name="value"
use="optional"/>
  </xs:complexType>
</xs:element>
```

Figure 3.0: Category Schema

XML file for tool release. While the generated file is likely to be correct-by-construction, a utility script is supplied to validate the XML file.

For WFS developers, Perl API's are provided to query for metadata of this instance. And as the XML instance is built on identical schema, identical API's can be used across components of Design Package.

D. *Work Flow Script (WFS)*

WFS is a PERL script used to prepare an environment for a work flow based on a Populate-Prepare-Run-Archive (PPRA) paradigm. A WFS is typically instantiated on a design block(s) and a corresponding Flow Instance Directory (FID) is created inside workarea.

In the PPRA paradigm, the "Populate" stage will pull required data from the source repository into a user workarea and create the corresponding FID. "Prepare" will create a Flow Environment Snapshot (FES) where all required metadata will be gathered such as design package information or it will write necessary files into the FID that will be needed later by the "Run" stage. In the "Run" stage, it will gather required Point Tool Executables (PTE) and set up the PTE from the corresponding stage and dispatch tasks for execution. Finally, once WFS has generated its desired result, all design data will be archived in "Archive" stage.

A WFS configuration consists of stages where each stage will have a set of PTEs. The purpose of each stage is to help WFS to handle situations when different PTEs (or settings) are needed in different scenarios within a single WFS script. For example, PTE 'VendorTool' version 1.0 is needed when processing Task A but PTE 'VendorTool' v 2.0 is needed to process Task B. In this case, a WFS configuration can create two stages: stage A which has PTE 'VendorTool' v 1.0 while stage B has PTE 'VendorTool' v 2.0. So when a WFS is processing Task A it will load PTE 'VendorTool' in stage A and load PTE 'VendorTool' v 2.0 in stage B when processing Task B. This provides enough flexibility to WFS to handle different scenarios that are needed for different settings.

In Cheetah Backbone Infrastructure, the FlowBase component is an interface module which provides APIs for WFS to retrieve the necessary information from components in the backbone such as getting design package content from DesignPackage module. The purpose of creating a FlowBase component is to provide a ready utility to a WFS without spending effort to understand how the Cheetah Backbone Infrastructure works behind the scene. Figure 4.0 illustrates the example of retrieving information of PDK with or without FlowBase component. This not only saves a developer's ramp up time but also, using the API provided by FlowBase, the WFS will not get impacted by any code refactoring happening behind Cheetah Backbone components as long as the output of the API remains the same.

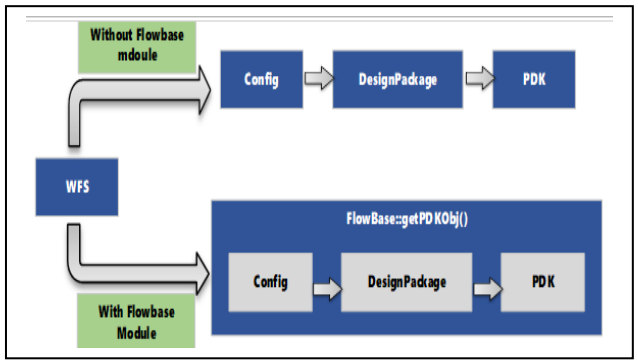

Figure 4.0: Example how WFS retrieves PDK object with/without going through FlowBase

E. Point-Tool-Executable (PTE)

The PTE component is aimed to create a runtime environment for the execution of tools. A PTE can be a 3rd party vendor tool or internal software packages which can be called directly by users or from a WFS. A PTE configuration resides in stages listed in a WFS. There are two ways to load a PTE: call a loadPte() API when running a WFS; or through a utility script named pte_setup.

loadPTE() is an API which will retrieve a particular PTE object and load content such as its environment variable setup, alias setup and license setup accordingly.

pte_setup is a utility script provided by the Backbone Infrastructure and exists for three purposes. First, it provides callback mechanism from a vendor tool to initialize other vendor tools. For example, some vendor tools use Tcl language to build a complex workflow and a utility callback script is necessary in order to invoke the Cheetah Backbone Infrastructure content which is coded in Perl. Second, to provide debugging functionality of the particular WFS. WFS developers need a direct way to debug their WFS configuration instead of debugging from their WFS script, which will help WFS developers to ensure their PTE is being configured correctly in every stage. Third, it allows for standalone PTE setup outside of a WFS run. This is especially useful when a WFS is unable to load a particular PTE. Instead of debugging a WFS script, which may take extensive debugging or compile time, pte_setup can be used as a first hand debug to ensure a WFS/PTE configuration has been configured correctly.

F. Job Management

Intel has its own **native** Load Balancing System, Cheetah has accommodated the same. Load balancing helps optimize the usage of CPU and memory consumptions; user machines can submit the jobs to the execution server through compute cluster. Although we only supported native load balancing now, the object oriented and layered approach that implemented in Cheetah Design System infrastructure is allowed us to plug-in other job management system such as LSF (Load Sharing Facility) quickly.

Four modes of job submission are supported: batch, interactive, shell and local. In **batch** mode a job is submitted to the batch pool and the command returns immediately. In **interactive** mode a job is submitted to interactive pool and the executing shell is blocking until the command returns. In **shell** mode a job is submitted to interactive pool and command gets executed in a terminal. In **local** mode job is submitted to local host and command gets executes locally.

1) Job Submit

There are two ways to dispatch the jobs: call the **createJob()** API when running a WFS; or through utility script named **cth_submit**.

createJob() is an API which takes compute requirements such as submission mode, required CPU, RAM, execution host, logfile path and etc. as input data and dispatches the task/job to native job submission tool. In verbose mode, a job submission command is printed out which is useful for debugging.

cth_submit is an utility script provided by the Cheetah Infrastructure to dispatch the jobs to native job submission tool or local execution. First, it provides for script callback to submit jobs from other vendor tools. For example, some vendor tools use Tcl language to build a complex workflow and a utility callback script is necessary in order to invoke the Cheetah Backbone Infrastructure content which is coded in Perl. Second, to provide debugging functionality of the particular WFS.

2) Job Operations

Job module provides a generic interface and methods to control job execution e.g.: status, kill, suspend, resume of a submitted job. These operations can be performed either through APIs or using an interface script named

cth_jobOperations. Additionally, job querying provides the status of the submitted jobs through native job submission tool.

G. Configuration Management (cth_config)

The Cheetah Design System allows projects to define a project configuration structure (XML based) that suits their project-specific needs for the SOC or IP development. It is very important that project administrator are able to create an accurate and error-less project configuration that is processed seamlessly by Cheetah Enter Project Scope (cth_eps) script. *cth_config* is a Configuration File Editor which is created to ensure that the project configuration is constructed correctly before released to the project design environment. It supports command line and Graphical User Interface (GUI) capability.

H. Design project management (cth_Indicator)

The Cheetah's backbone infrastructure has come up with generic solution for Indicator catering to all BUs requirements by providing many configurability options including hooks for different parsing mechanism, database options etc. With Cheetah being design system basic configurability like overriding on project level, user level are by product for Indicator solution. Currently 3 flows have integrated with Indicator hooks and are able to capture the design data in dashboard.

III. RESULT

Since Cheetah Infrastructure Backbone developed by taking requirements from different available design systems, most of basic requirements are provided in first production release. In a span of few months 18 flows (WFS) are developed in Cheetah design system. All these flows are using Backbone infrastructure APIs and added only tool specific code in their flows. I.e. Common backbone infrastructure enabled lean WFS and shorter development time. This can be demonstrated by looking into file size, overall backbone infra code size is ~80k and every WFS code size in the range of 0.5k to 10k as per complexity. WFS development was not at all complex, Infrastructure team provided consultation on how to use the Backbone Infrastructure APIs in their WFS to align with Cheetah's "bare-metal" concept and that was sufficient for WFS developers to follow. As of today, we successfully enabled **RTL2GDS, AMS** and **IP Handoff** flow as a pioneer WFS release and in same foot print continued for Front End flows. Initial development was much focused on feature and then we improved usability. Once we achieved stable baseline of infrastructure we refactored the code and improved speed by 60%-90% depends on the design complexity. The benefit we are achieving here in Backbone leads to overall improvement in Cheetah design system.

In the Design Package domain, the modular schema first adopted in PDK and Library components has been proposed to Design Data Management charter to be used to express a design block's manifest file. This manifest file contains design bundles such as RTL, OpenAccess schematic or layout, Register files, and secondary views such as GDS and Spice netlists. The same concept can be applied where each bundle has its associated classifications. Therefore, identical XML capabilities can be used across PDK, Process Libraries and Bundle.

Configuring Cheetah project with Configuration File Editor is possible only in an hour, once the physical setup exists. During new cheetah project configuration, this utility was quite handy and helped to configure seamlessly.

At present, Cheetah has been deployed to 5 projects in three business units spread across five geographical locations.

In the latest Cheetah infra backbone release we achieved the performance efficiency by 90% and improved the strict XML schema validation. Below table illustrates one of the example to showcase the efficiency:

Cheetah releases	Time taken by Enter Project Scope (EPS)	Conclusion
Old	100 seconds	Taking long time in merging all the XML's
New	9 seconds	90% efficiency achieved after the parser refactored code

Table 1.0: Time taken by EPS versus performance efficiency

Paper on Cheetah infra backbone topic was submitted to **Intel's premiere technical conference** (DTTC) and got selected for **presentation**.

IV. CONCLUSION

In summary, the Cheetah Design System is a "bare-metal" design system which allows the modular design workflows to be built on top of a scalable infrastructure whereas the Backbone Infrastructure is software used to fulfill the scalable infrastructure criteria. In order to align with set goals, a set of design principles being used to design Cheetah Design System Infrastructure (aka Backbone Infrastructure) to ensure we could move to more native point-tool-executable usage and shift to vendor supported design workflows. Backbone Infrastructure comprises backbone capabilities such as Enter Project Scope, PTE, Design Package, Job Management and Flowbase which were discussed in the details of this paper. All these Backbone Infrastructure capabilities are needed for different design workflow execution and are accessible via APIs that provide consistent access to the underlying IT infrastructure and design system components. Backbone Infrastructure needs to continue to align with the bare-metal concept by strongly adhering to design principles. It needs to extend its capabilities and integrate all design workflows from system level until tape-in to Cheetah in order to help Intel win new market segments. As for the first step to win, the Cheetah Design System Infrastructure has already been successfully enabled for new technologies. The next step is to await feedback from projects for further improvements in full execution mode.

V. ACKNOWLEDGMENT

The authors would like to take this opportunity to acknowledge and thank the Cheetah Infrastructure architects Marek Rouchal, Olson Christopher and Jarod Eells for their guidance and technical direction in order to make Cheetah Backbone Infrastructure a success. Special thanks to Backbone Infrastructure development team for the efforts; Sheau Lan Lee, Satiaseelan Selvan, Puneeth H K, Chong Yuong Chee, Pavan Ghooli, Bharathi Chandra Shekara and Chaitanya Pande.

Many others contributed to the development of Cheetah Backbone Infrastructure; some of these are, in no particular order, Sankarshanan Pillaipakkam, Martin Maurer, Michael Penner, Brian Lee and Dhananjay Haridas.

REFERENCES

- Cheetah Design System Infrastructure is an internal tool developed in Intel and uses the learnings from previous Design Systems

- Hui Chyn Ng, Michael Wirth, Divya Ramarao, Michelle Mian Yiung Phoon, Shih Peng Tan, “Cheetah: innovation of design system infrastructure”, in press.

- Chee Chong Yuong, Susmita Pal, Prateeksha Keshari, Hui Chyn Ng, “Configuration file editor for effective cheetah configuration management”, unpublished.

AUTHORS BIO

Divya Ramarao is with PESG-Design Infrastructure team in Intel India. She is currently working as a tech lead/scrum master in the Cheetah Design System Infrastructure space. She is also one of the key developer on Cheetah Infrastructure and also driving the topics for future Infrastructure releases. Since 2006 she has been also working on various in-house tools and methodologies for the success of mobile products.

Mahesh Deshpande is with PESG-Design Infrastructure team in Intel India. He is currently working as a program manager and immediate manager for the cheetah infrastructure.

Susmita Pal is a Computer Aided Design Engineer in PESG-Design Infrastructure team at Intel India. She is one of the key developer and played significant role in developing various component in Cheetah Infrastructure.

Prateeksha Keshari is a Computer Aided Design Engineer in PESG-Design Infrastructure team at Intel India. She is one of the developer for the Cheetah Infrastructure and has contributed significantly towards several components.