Process Design Kits with Open Standards

Dzmitry Pustakhod[1], Ronald Broeke[2], Rui Santos[1], Sylwester Latkowski[1], Kevin Williams[1], and Xaveer Leijtens[1]

[1]Eindhoven University of Technology, Electrical Engineering, 5612 AZ, Eindhoven, the Netherlands
[2]Bright Photonics, Horsten 1 – Multimedia paviljoen, 5612 AX Eindhoven, the Netherlands

In this paper, we present a tool-independent workflow for creating Process Design Kits (PDKs) for Photonic Integrated Circuit design. Historically, fabs had to work separately with each Electronic-Photonic Design Automation (EPDA) tool vendor to create a PDK for the corresponding tool. We have developed and tested an alternative PDK creation process based on a standardized PDK data representation. An important advantage of the new approach is that it lowers the load on the Foundries – PDK owners. Additionally, a Foundry PDK is validated automatically and independently before it is converted to the PDK instance and delivered to the PIC designers. Finally, presence of the open standards for Foundry PDK significantly lowers the entry barrier for new EPDA tools.

1. Introduction

In both integrated electronics and photonics, Process Design Kits (PDKs) play a central role: they enable design of (photonic) integrated circuits, PICs [1, 2]. A PDK describes a specific fabrication process from a specific fab, and contains all information required to create a PIC design: design manual, information about technology, a set of building blocks, design rules. Usually, a PDK is implemented in a package or a library to be used with a specific Electronic-Photonic Design Automation (EPDA) tool.

Currently, a foundry owing a process has to work separately with each EPDA tool to create PDKs for this process (Fig. 1a). And although PDKs for the same technology are based on the same raw data, they are created and compiled in a different way for each design software.

Taking into account design tool diversity, it becomes a difficult task for a foundry to create and support a separate PDK for each tool its customers (designers) use. Besides

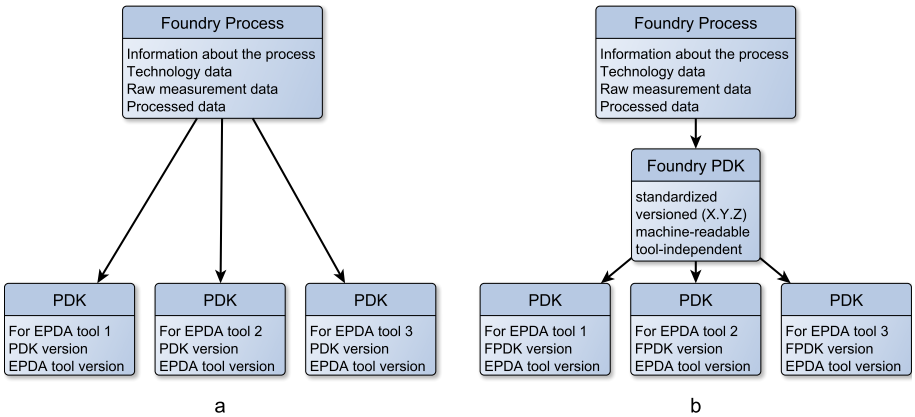

Fig. 1. PDK creation. a – Traditional workflow with direct implementation of PDKs for various tools. b – Proposed workflow with a standardized Foundry PDK as an intermediate representation of the data.

adding extra load and overhead on the foundries, the existing approach has a number of intrinsic problems directly affecting designers and foundry customers. First, this approach is error-prone, since each PDK implementation has to be validated separately. Releasing regular updates to the PDKs is another issue, as it requires time and effort to propagate the changes to all the PDKs. In case of an EPDA tool update, all PDKs for this tool have to updated too. In addition to the mentioned problems with PDKs from the foundry, extending PDKs with IP blocks from design houses is challenging because of the same issues. From design tool perspective, each fab which provides a PDK for this tool creates it in a different way, thus lacking the possibility to automate the PDK creation and validation process.

In this paper we propose a solution to these problems which features a standardized tool-independent representation of the foundry offer, *Foundry PDK* (FPDK) (Fig. 1b). It is used as an intermediate step between the raw foundry data and the PDK implementations for the design tools (referred to as *PDK instances* further in the paper). We demonstrate that such standardization allows for independent validation of the created PDKs. It simplifies creation of the PDK releases on both the foundry side – single release per technology is needed – and the design tool side – it becomes possible to automate PDK instance creation and reuse this automation for PDKs from several fabs.

We present a process of PDK creation which makes use of the FPDK standardization. The process was developed and implemented within the joint European platform for photonic integrated components and circuits (JePPIX) [3] and is currently used to create PDKs for two InP foundries (Fraunhofer Heinrich Hertz institute (HHI) and SMART Photonics) and four EPDA tools (Nazca Design, Photon Design, Synopsys, and VPI Photonics). In addition, an extension for the SMART Photonics PDK by the Eindhoven University of Technology (TU/e) was created using the same workflow.

2. Foundry PDK

Structure The proposed standardized Foundry PDK consists of the following parts:

1. *Technology platform description*, description of the fabrication platform. It includes a list of mask layers, materials, and their properties; layer stack description; geometry of cross-sections and their mapping to the mask layers, etc.
2. *Foundry building blocks (BB, Pcells)*, representation of black-box layout information of the standard BBs provided by the foundry [2]. It includes BB dimensions, connections (optical ports and electrical pins), and optional parametrization.
3. *BB performance data – numeric and compact models*. It is provided in either numeric (tabular) form, or as analytic expressions. Data may include mean values and variations.
4. *Design rules*, a list of rules representing platform limitations.
5. *Design manual*, a human-readable representation of the FPDK. It is typically a PDF document with detailed explanations of the process, building blocks, and data. The design manual is usually made available directly to the designer.

The FPDK release is a zip package with a folder structure which reflects the above list. We currently adopted a 3-number versioning scheme for the PDK releases (X.Y.Z).

Open standards for Foundry PDK representation The data in the FPDK is very diverse, and it can be grouped into three classes: tabular numeric data, analytic data, and complex structured data. To enable to enable unambiguous import and validation, we have to standardize the way in which this data is represented.

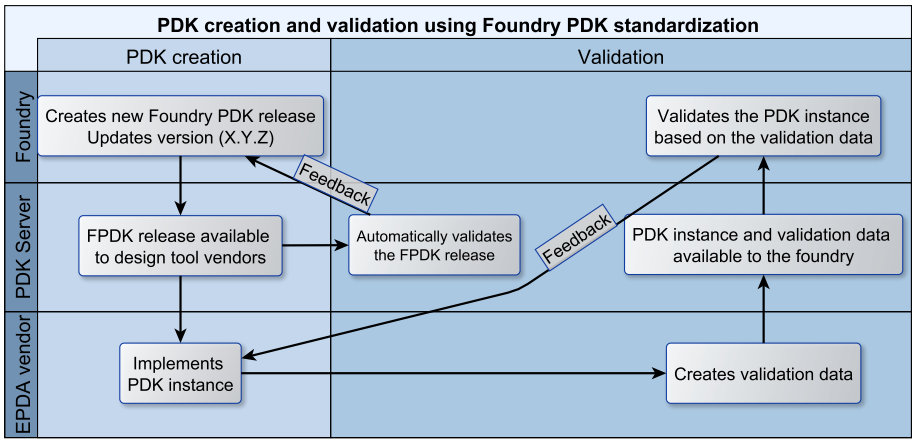

Fig. 2. Implemented workflow for PDK creation and validation

A data format includes two independent parts: data serialization and data schema. To represent the data in the PDK, we have adopted existing open formats.

Serialization formats. Serialization of the data is the way how it is represented as stream of characters. We employ two serialization formats: YAML [4] and openEPDA data format [5–7]. YAML is used for complex structured data because it offers means to represent such data in both machine- and human-readable way. The openEPDA data format is used to represent numeric and analytic data, and it combines metadata and scalar data with tabular numeric data in a single file.

Data schemas. A data schema defines the data structure and value types. We use JSON Schemas [8] to describe the data. An example of the data schema used for representation of the foundry building blocks is the uPDK format [9]. Schemas for the numeric and analytic data are generated dynamically from a set of standard attributes.

3. PDK creation and validation

The new PDK creation process consists of two steps: FPDK release creation, PDK instance creation. Additionally there is data validation step. We have setup an internal JePPIX PDK server which enables all the workflow and is currently in use by the foundries and software vendors. The steps are described below in more detail.

FPDK release creation A foundry creates an FPDK release. Contents and form of a FPDK release is defined in section 2. The specific FPDK release creation process may differ for different foundries, but ideally the data comes from a single source with version control and the creation process is automated using the continuous integration / continuous delivery (CI/CD) practices. After creation of the FPDK release, a foundry makes it available to the SW vendors via the JePPIX PDK server.

PDK instance creation By PDK instance we mean an implementation of a PDK for a specific EPDA tool. It may be an installable package, a plugin, or some other kind of extension of the design software.

To create a PDK instance, the SW vendors obtains the FPDK release by downloading it from the JePPIX PDK server. Next, they import the data from the FPDK release. The definitions in section 2 make this process unified for FPDKs of different foundries. Then, the data is used to create a library or a package for the design tool.

After creating a PDK instance, a software vendor generates validation data using the EPDA tool – of a specific version – and the PDK instance. The validation data may include

a GDSII layout file for a standardized PIC layout or a simulated performance of predefined circuits. Finally, the PDK instance and validation data are provided to the Foundry for validation via the JePPIX PDK server.

Independent PDK validation The standardization of the various elements enables independent PDK validation on several levels.

Validation of the Foundry PDK release. This validation takes place automatically upon upload An example of such a validator – implemented as an online service – is the uPDK file validator [10]. An uploaded uPDK is automatically checked against the uPDK schema. In addition, a GDSII file with all the building blocks described in the uPDK file is generated. It can be used by the foundry for automated cross-checking their data.

Validation of the PDK instances. The validation data generated by the EPDA tool vendors (e.g. a PIC layout which uses black boxes of the BBs) can be automatically checked against the corresponding FPDK. This is one of the steps of confirmation of the PDK instance correctness. Validated PDK instances become available to the customers.

4. Conclusions

We have presented a software-independent process for creating PDKs for PIC design. The process decouples foundry PDK releases from creation of PDK instances for specific EPDA tools. The intermediate foundry PDK representation uses open and free standards.

The approach offers several advantages from both technical and business perspective. Technically, a higher quality PDK is achieved due to independent validation steps throughout the process. From business point of view, PDK standardization reduces the effort needed to create and support PDKs for several EPDA tools and also lowers barriers for new EPDA tool developers.

Acknowledgements The authors would like to acknowledge support by European Union's Horizon 2020 program no. 824980 – InPulse. The authors would like to thank the InPulse consortium partners for collaboration on this topic.

References

1. W. Bogaerts and L. Chrostowski, "Silicon Photonics Circuit Design: Methods, Tools and Challenges," *Laser & Photonics Reviews*, vol. 12, no. 4, p. 1700237, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/lpor.201700237
2. L. M. Augustin *et al.*, "InP-Based Generic Foundry Platform for Photonic Integrated Circuits," *IEEE J. Sel. Topics in Quantum Electron.*, vol. 24, no. 1, pp. 1–10, Jan 2018.
3. "Joint European platform for InP-based photonic integrated components and circuits," http://www.jeppix.eu.
4. "YAML Ain't Markup Language (YAML™) Version 1.2." [Online]. Available: http://yaml.org/spec/1.2/spec.html
5. S. Latkowski *et al.*, "Open standards for automation of testing of photonic integrated circuits," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 25, no. 5, pp. 1–8, Sep. 2019.
6. "openEPDA Overview — openEPDA™ documentation," https://openepda.org/.
7. "CSVY yaml frontmatter for csv file format," https://csvy.org/.
8. "JSON Schema — the home of JSON Schema," http://json-schema.org/.
9. "openEPDA uPDK™Blocks — openEPDA™documentation," https://openepda.org/updk/pdk_components.html.
10. "Validate uPDK file," https://validate.openepda.org/v/updk.