Automated test framework for photonic integrated circuits

D. Pustakhod, S. Latkowski, M. Chatzimichailidis, W. Yao,
K. Williams, and X. Leijtens

Eindhoven University of Technology, Dept. of Electrical Engineering
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

With the development of fab-less approach to integrated photonics, there is an increasing demand on infrastructure for testing and characterization of fabricated devices. In this paper we describe a test software framework and propose standards for chip and measurement descriptions, and for the file format to be used for storage and exchange of measurement data.

Introduction

A generic approach to photonic integration has led to an increased interest to this technology in many application areas [1], [2]. Performance of a fabrication process has to be validated for every run, so as the performance of every fabricated device. Therefore testing of the photonic circuits plays an important role in the production chain, and its contribution to the total product cost can reach up to 30% [3]. Testing performed at different stages of the fabrication pursues different objectives. On-wafer testing during and after the fabrication is done for process qualification. Bar and chip testing is first run by the foundry for building block qualification, and later can be done for circuit-level performance characterization. Finally, module-level testing qualifies functional and system performance of the packaged devices.

In order to cope with increasing volume of fabrication, test automation is crucial. To satisfy the increasing demand in data amount, quality, and traceability we are developing a framework for measurement automation and data handling. The proposed approach is to interface the photonic integrated circuit (PIC) designers who define the test protocol on the design stage with the test facilities. We have developed a standardized modular approach to the test infrastructure, which enables automation of the process and at the same time allows user to define measurement procedures and their parameters. The modular approach is presented in the first part of this paper. In the second part, we present an open data format, which is used to store heterogeneous measurement data together with experiment metadata.

Framework Description

The idea behind the framework is to decouple the control software run by the test facility from the test settings provided by a PIC designer.

Software modules

The software consists of several control modules, each responsible for a specific task. The hardware control module is used for communication with the equipment. The positioning and alignment control modules perform chip placement and optical alignment respectively. Measurement control module initializes and runs the measurement modules according to user defined script, and the data storage controller is responsible for saving the measurement data. In Fig. 1, software part is shown in green.

The framework has modular structure, where different parts are coupled with other parts through a well-defined open application program interface (API), and can easily be replaced or extended. For example, the test actions are described through the measurement modules. These measurement modules can be standard (provided by the test house, e. g. current–voltage curve measurement) and user-specific (provided by the user to characterize a specific circuit). The measurement modules define the tests in an instrument-agnostic way, and can be run with any other instruments of the same class.

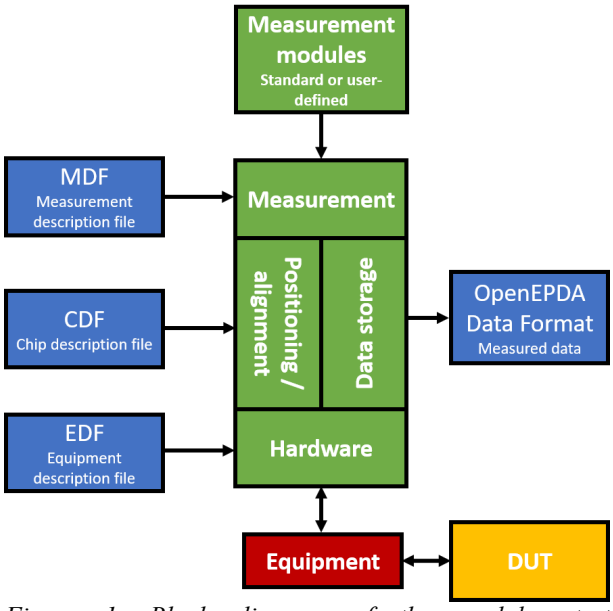

Figure 1. Block diagram of the modular test infrastructure

Configuration files

Various settings required for the measurements are contained in the configurations files, which are the equipment description file (EDF), the chip description file (CDF), the measurement description file (MDF) (Fig. 1, 2). The configuration files are written in the YAML format [4], which is a human readable and editable text format with a rich support in many programming languages.

```yaml
setup:  # setup name
    OLA TS1 Setup

# electronic / optical measurement equipment
devices:
  ktl1:  # device id to be used in MDF
    device_cls: Keithley2450
    device_params: {port: 'GPIB::23'}
    ch_ids: [a]
  ktl2:
    device_cls: Keithley2450
    device_params: {port: 'GPIB::24'}
    ch_ids: [a]
  ilx:
    device_cls: ILXLDC3900
    device_params: {port: 'GPIB::1'}
    ch_ids: [2, 3]
  ando:
    device_cls: AndoAQ6315A
    device_params: {port: 'GPIB::16'}
    ch_ids: ['A']
  pm:
    device_cls: AgilentTLS
    device_params: {port: 'GPIB::20'}
    ch_ids: [1]

# dictionary of stage controllers
stages:
  mg_left:
        stage_cls:  MG17PCW011
        stage_params:  {port: "GPIB::12"}
        ch_ids: [1, 2, 3]

stage_axis_mapping:
    left:
        fine:
            focus: [mg_left, 1]
            hor: [mg_left, 2]
            vert: [mg_left, 3]
```

a

```yaml
Cell: SP19-3-4
Unit: um

Fiducial:
  target:
    - fidt0: [2250, 257.5]
    - fidt1: [2250, 3642.5]
  cornerUL:
    - cul0: [1737.5, 257.5]
    - cul1: [1737.5, 3642.5]

Port:
  io_optical:
    - ioL001: [-50, 25]
    - ioL003: [-50, 50]
    - ioL005: [-50, 75]
    - ioL007: [-50, 100]
    - ioL009: [-50, 125]
    - ioL028: [-50, 362.5]
    - ioL030: [-50, 387.5]
    - ioL032: [-50, 412.5]
    - ioL034: [-50, 437.5]
    - ioL041: [-50, 525]
    - ioL043: [-50, 550]
    - ioL045: [-50, 575]
    - ioL047: [-50, 600]
    - ioL049: [-50, 625]
    - ioL053: [-50, 675]
    - ioL055: [-50, 700]
    - ioL057: [-50, 725]
    - ioL059: [-50, 750]
    - ioL061: [-50, 775]
    - ioL071: [-50, 900]
    - ioL073: [-50, 925]
    - ioL075: [-50, 950]
    - ioL077: [-50, 975]
    - ioL079: [-50, 1000]
    - ioL083: [-50, 1050]
    - ioL085: [-50, 1075]
```

b

```yaml
# Cell ID links to CDF
Cell: SP19-3-4
# Chip ID number from chip
Chip: 35F1
# Wafer ID is needed, because chips from different
# wafers may have the same chip ID
Wafer: SPL135-2

Reference:
# Straight waveguide #1 at bottom
  - ref0: [ioL001, ioR001]
# Straight waveguide #103 at top
  - ref1: [ioL309, ioR309]

# Only straight facet measurements
Measure:
  Meas00:
# Straight shallow
    - {opt0: ioL001, opt1: ioR001, measure: fp_loss}
    - {opt0: ioL003, opt1: ioR003, measure: fp_loss}
    - {opt0: ioL005, opt1: ioR005, measure: fp_loss}
    - {opt0: ioL007, opt1: ioR007, measure: fp_loss}
    - {opt0: ioL009, opt1: ioR009, measure: fp_loss}

  Meas01:
# Straight deep
    - {opt0: ioL041, opt1: ioR041, measure: fp_loss}
    - {opt0: ioL043, opt1: ioR043, measure: fp_loss}
    - {opt0: ioL045, opt1: ioR045, measure: fp_loss}
    - {opt0: ioL047, opt1: ioR047, measure: fp_loss}
    - {opt0: ioL049, opt1: ioR049, measure: fp_loss}

  Meas02:
# Serpentine deep
    - {opt0: ioL053, opt1: ioR053, measure: fp_loss}
    - {opt0: ioL055, opt1: ioR055, measure: fp_loss}
    - {opt0: ioL057, opt1: ioR057, measure: fp_loss}
    - {opt0: ioL059, opt1: ioR059, measure: fp_loss}
    - {opt0: ioL061, opt1: ioR061, measure: fp_loss}
```

c

Figure 2. Examples of EDF (a), CDF (b), and MDF (c)

The setup performing the test is described in the EDF. It contains the list of available equipment, their settings and calibration data. The EDF is provided by the test house, and is permanent for a given measurement setup.
Measurement tasks from the users are provided in the form of CDF and MDF. The CDF contains information needed for execution and automation of the measurement processes. It includes names and coordinates of optical input/output ports, electrical pads, and any fiducials present on the chip. Information about coordinates is required for automated chip positioning and accessing required inputs and outputs with electrical and optical probes. The MDF contains description of measurements to be run in a given test session. Here, specific measurement settings are stored together with the names of the test actions to be performed. Both CDF and MDF can be generated automatically during the chip design stage, which is a step towards implementation of design-for-test methodology.

OpenEPDA data file format

Data types

The data generated during the measurements is used for different purposes: pass-fail procedure, process control, device models development and calibration. This involves various parties, such as foundry, measurement labs, designers, and software companies which use different tools to generate and process the test data. Therefore, the generated data comes from different sources and can be heterogeneous. The experimental data is obtained from the measurement equipment directly when the observation is performed. It is usually numeric in form of scalars or arrays. The identifiers of the wafer, die and circuit under test represent metadata for the given observation. Metadata may also include the information regarding equipment used, particular settings, date of calibration, ambient conditions, etc. The metadata can be of various types for example simple numeric or textual and structured as arrays or maps. An overview of the datatypes is presented in the Table 1.

Table 1. Examples of the data types

Data type	Description	Examples	Remarks
Number	Any numeric value	2.3 .inf 1.9e-3	Representation is same as defined in section 10.2.1.4. of [1]
String	A list of characters	'spectrum analyzer' 'SPM18-3'	
Array	A sorted list of numeric of string values	[1, 2, 3, 4] ['west', 'east', 'north']	Values may have mixed types, which is discouraged
Map	Mapping of a set of values to another set of values in the `key: value` form.	{'wafer': 'SPM18-3', 'die': '38X23', 'design': 'SP00-38'}	Also called a named array, a look-up table, or a dictionary

Standardized file format

To facilitate exchange of the data between different parties and software tools, we have developed a standard file data format, which can store the measurement data and metadata in a human-readable way. The format is sufficiently flexible to include any arbitrary structured data, including arrays and maps. The generated files are straightforward to be imported by any software or analysis tool, for example MATLAB, python, and MS Excel. The proposed data exchange file format has the following structure. The first line is a format identifier, "`# OpenEPDA DATA FORMAT vX.Y`". After this, the data part follows, which contains two sections. The first section adheres to the YAML format [4],

and contains all scalar (numeric and textual) data in the `name: value` form. This format is applicable for storing all previously mentioned types of the data. The second section is a CSV-formatted [5] part to store the tabular measurement data.
The proposed format does not define any required keys in the metadata or data parts, it only defines how to write the information into a file. However, in the future particular reserved keys may be defined, which, for example can start with an underscore character. An example of a measured optical spectrum is shown in Fig. 3. Line 1 specifies the file format and its version. Lines 2 to 14 contain metadata in YAML-format. Here, only string and float data types are used, however arrays and maps can also be included. Line 14 contains a standard document-end marker. Lines 15 to the end of the document contain the CSV-formatted tabular data with a single header line.

```
Line #    File contents
 1         # OpenEPDA DATA FORMAT v.0.1
 2         _timestamp: '2018-09-12T09:59:19.310182'
 3         prj: OpenPICs
 4         setup: RF setup
 5         operator: Xaveer
 6         wafer: 36386X
 7         sample: 13L8
 8         cell: SP35-1-3
 9         ckt: MSSOA1-6
 10        J_kA_per_cm2: 1
 11        pol: NO
 12        port: east
 13        t_chip: 18
 14        ...
 15        "Wavelength, nm","Transmitted power, dBm"
 16        0.00000000000000000e+00,0.00000000000000000e+00
 17        1.00000000000000000e+00,1.00100100100100162e-01
```

Figure 3. Example of the OpenEPDA data file. The file contains the format identifier with the version, YAML-formatted section, and CSV-formatted section.

Conclusions

In this paper, we have presented a modular test infrastructure for PIC test and characterization. This approach defines a test platform with a high throughput at a lower cost. The presented OpenEPDA measurement data format facilitates efficient measurement and simulation data exchange and traceability for further statistical process control.

Acknowledgements

The authors would like to acknowledge support of the Stimulus OPZuid program through the Open Innovation Photonic ICs project (PROJ-00315) and European Union's Horizon 2020 programme (n° 431954 – PIXAPP).

References

[1] M. Smit *et al.*, "An introduction to InP-based generic integration technology," *Semicond. Sci. Technol.*, vol. 29, no. 8, p. 083001, Jun. 2014.
[2] "JePPIX roadmap 2018," *JePPIX*. [Online]. Available: http://www.jeppix.eu/vision/.
[3] E. R. H. Fuchs *et al.*, "Process-based cost modeling of photonics manufacture: the cost competitiveness of monolithic integration of a 1550-nm DFB laser and an electroabsorptive modulator on an InP platform," *J. Light. Technol.*, vol. 24, no. 8, pp. 3175–3186, Aug. 2006.
[4] "YAML Ain't Markup Language (YAML™) Version 1.2," 1992. [Online]. Available: yaml.org/spec/1.2/spec.html.
[5] RFC 4180, "Common Format and MIME Type for Comma-Separated Values (CSV) Files," 2005. [Online]. Available: tools.ietf.org/html/rfc4180.