PDAFlow Foundation
The emerging standard for electronics, photonics and fluidics

Arjen Bakker

Plat4M and SPTAB meeting
Brussel
STANDARDIZATION

OBJECTIVES

- Large variety of software
- Results in
  - high cost
  - incompatible libraries
  - data transfer errors
  - Nomenclature inconsistencies

- Standardize design flow
- Interoperate software
- Maintain advantages of each method / software
- Simplify training
Fabless designers using standard commercial tools

From
- C++ definitions
- TDK4PE XML
- OpenPDK
- SPT input
- Private content
- Plain text
- home grown

To
- PDAFlow C++ API
- Bridging the gaps
- Spice5
- Design rule check
- GDS
- Frequency domain
- Time domain
- High-speed inkjet printers
- OpenAccess & SPTAB

Non traditional
- CMOS
- Manhattan

Add
- Functionality
- Variability
- Multi-domain physics

External foundry services or in-house libraries / fab (PDK, IPB, ...)

Arjen Bakker (Phoenix BV)
OBJECTIVES & DEVELOPMENT

- Allow ( photonics ) software co-operation
- Single “foundry” definition in PDK
- Support IP-blocks
- “Plug-in” libraries
- Extend software capabilities using eg. co-simulation, mask layout post processing
- API defined in (“basic”) C++
- Joint development via SubVersion (svn)

STICHTING PDAFLOW FOUNDATION
Focus on design

- Simulation
- Mask layout
- Documentation (including legal aspects)

Using

- Integration via C++ API
- Building block
- Netlist
- Foundry
  - Process flow
  - Crosssection
Main focus is ‘functionality’

- Building block and technology definition (PDK)
  Input: OptoDesigner spt, TDK4PE xml, C++, ...
- Schematic capture & optical circuit simulation (Aspic, ComponentMaker, Interconnect, PicWave)
- Functional & geometric design rule check (OptoDesigner)
- Physical layer solvers like optical mode solvers, propagation; RF (FIMMPROP, OptoDesigner)
- Mask layout generation (CleWin, OptoDesigner)

≈ Design file persistence in binary and ascii format

➢ Link with OpenAccess in Plat4M
KEY CLASSES

- Using the “Model View Controller” concept
- Content = “attributes” (data) + “View”
- API defines COMPONENT’s
  - BB models, design rules, layout
  - NETLIST BB’s + connections + attributes
  - FOUNDARY mask layers, technology
- PROCESSFLOW “parametric” crosssection
- CROSSSECTION technology (substrate) model
**API: Data & View**

**Data** Describes the component “fully”
- length, width, technology for a straight application independent
- can include valid ranges and distribution

**View** Use the component to do something; eg. “simulate” or “layout”
- View’s can have data too
  - grid settings a modelling view
- View’s can access the ‘owner’ (bb, netlist, . . .)
  - Use length from the BB
  - Use modesolver view on the technology

Arjen Bakker (Phoenix BV)
PDAFlow Overview

PDAFlow API as Center Point

User Input

BBparamVals and circuit connections

Graphical Output

simulation result graphs

PicWave & Aspic

BBavailList from all dlls
For each BB: PortList, BBresponse

BBparamVals

Netlist

BBresponse BBlayout

MyFoundary
my BBavailList
Phoenix “Spt” data for each BB (parameterised)

BBavailList from other dlls
BBparamVals
(In principle all BB data from all other dlls)

BBresponse BBlayout

pdaFlow Library

Netlist, BBparamVals, BBlayout for each BB

PdaDll.dll

“Pxnetlist” - assembles set of BBlayouts into single device layout

GDSII MaskFile

Arjen Bakker (Phoenix BV)
```cpp
class myStraight : public BB {
public:
  pda::String getName() const { return "myStraight"; }

public:
  /// Construct
  myStraight(const Foundry* f)
  : BB(f) {
    // Ports
    addPort("in0", pda::EnumList::Optics);
    addPort("out0", pda::EnumList::Optics);
    // Variables
    addAttribute(new Value<pda::Double>("Width", 1.5));
    addAttribute(new Value<pda::Double>("Length", 100.));
  }
};
```
```cpp
int main() {
    pdaInit_setVersion("10");
pdaInit_pdaview();
    // Register local BB
    BB::Register<myStraight>();

    // Create the DemoFab.
    if (Foundry* f = Foundry::create("DEMOFAB")) {
        if (BB* bb = BB::create(f, "myStraight")) {
            bb->setDouble("Width", 200.);
            pdaCout() << "Works – BB definition is ready!\n";
        }
        else pdaCout() << "--myStraight not available\n";
    }
    else pdaCout() << "--Demofab not available\n";
    pdaShowCoutMsg();
    return 0;
}
```
class myStraight : public BB, public BBDocView {

public:

  pda::String getName() const { return "myStraight"; }

  //-- BBDocView interface --

  pda::String docAuthor() const {
    return "Me!";
  }

  pda::String docVersion() const {
    return "Alpha.";
  }

public:

  /// Construct

  myStraight(const Foundry* f)
  : BB(f), BBDocView(this) {
  }

Arjen Bakker (Phoenix BV)
class myStraight : public BB, public phxBBSpiceView {

public:

    pda::String getName() const { return "myStraight"; }

    //-- phxBBSpiceView interface --

    void spice_model(std::ostream& os, std::set<pda::String>* bbDone) {
        phxBBSpiceView::spice_model(os, bbDone);
        os << ".subckt peSourceAC"
        spice_model_args(os);
        os << "   V1 out0 gnd AC {Length} SIN(0 {Length} { Width})
            .ends
        "
                 ".ends\n\\n";
    }

public:

    /// Construct

    myStraight(const Foundry* f) : BB(f), phxBBSpiceView(this) { }
if (Foundry* f = Foundry::create("DEMOFAB")) {
    if (BB* bb = BB::create(f, "myStraight")) {
        bb->setDouble("Width", 200.);
        pdaCout() << "Works - BB definition is ready!\n";
        if (phxBBSpiceView* vw = bb->getView<
            phxBBSpiceView>()) {
            pdaCout() << "\n\n----\nSpice view: \n";
            vw->spice_model(pdaCout(), NULL);
        }
    }
} else pdaCout() << "--myStraight not available\n";
DemoFab concept
Where will it produce...

- Used for development within PhoeniX Software and by partners
- Demonstrates main concepts using non NDA / project information; thus fully open
- As extensive example
  - Designkit, file structure, ...
  - Mask layout BB’s
  - IPBlocks
  - Packaging templates
  - Using a wide variety of PDAFlow Views
    - Crosssection simulation
    - Process flow simulation
    - Topview / propagation simulation
    - Link with PhoeniX Living Database
- Or as base for a designkit (by removing a lot)
- Source available for users on svn server