Возможности использования открытых инструментов для автоматизации адаптации PDK к OpenLane

Открытая конференция ИСП РАН 2022

Уманский М.В., Лукьянченко Г.А., Монахов А.М., Черных Е.А.
ООО «Мальт систем»

MALT system

Содержание

Введение

Введение

- Открытые инструменты достигли достаточного уровня для разработки реальных промышленных микросхем;
- Самым полным из открытых маршрутов RTL-to-GDS является OpenLane;
- Остро стоит вопрос адаптации коммерческих PDK;
- Проприетарные инструменты имеют высокий порог входа и ориентированы на кремниевых разработчиков;
- Открытые инструменты предоставляют богатый API в привычном программисту виде;
- Открытость лицензии позволяет аккумулировать наработки разных коллективов;
- Мы расскажем о пройденном нами маршруте адаптации 130нм PDK, а также разработанном нами PCell efuse на GF180 в открытых инструментах.

Из чего состоит PDK?

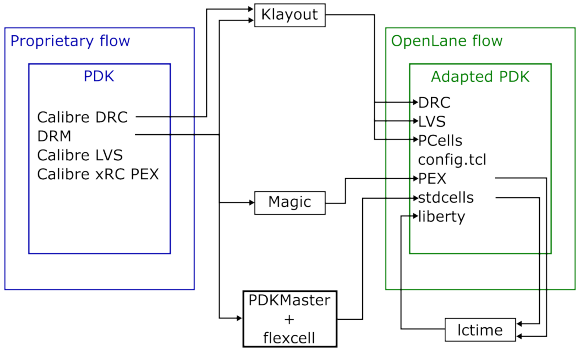
- DRM - Design Rule Manual;
- Скрипты для DRC (Design Rule Checking);
- Скрипты для LVS (Layout Versus Schematic);
- Скрипты для паразитной экстракции (PEX);
- Модели примитивов (devices);
- Библиотеки ячеек.

Открытые и закрытые форматы

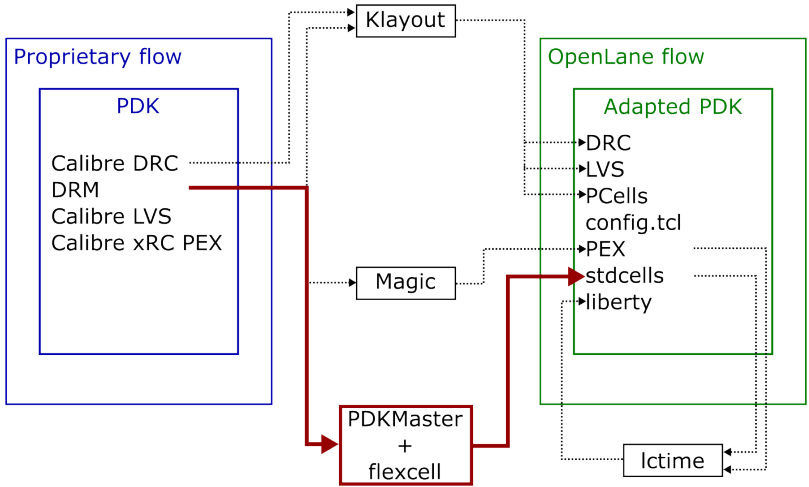
Элементы PDK	Форматы	Проприетарность
DRC-скрипты	SVRF	Да
LVS-скрипты	SVRF	Да
PEX-скрипты	Calibre xRC	Да
Модели для аналоговой симуляции	SPICE	Нет
Layout ячеек	GDS II	Нет
Нетлисты ячеек	SPICE	Нет
Результаты характеризации	Liberty	Нет
Топология	LEF/DEF	Нет

В чем состоит адаптация PDK к маршруту

- Получение LVS и DRC скриптов, исполняемых открытыми инструментами;
- Написание технологических файлов для требующих этого инструментов;
- Генерация библиотеки ячеек и технологического LEF.



PDKMaster

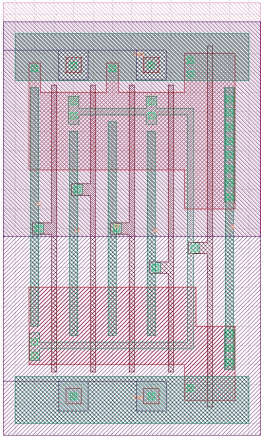

Klayout
Proprietary flow
PDK
Calibre DRC
DRM
Calibre LVS
Calibre xRC PEX
OpenLane flow
Adapted PDK
DRC
LVS
PCells
config.tcl
PEX
stdcells
liberty
Magic
PDKMaster
+
flexcell
lctime

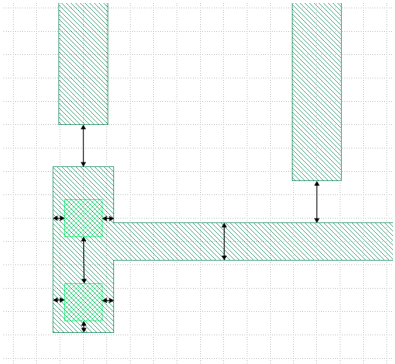
Что такое PDKMaster

- Фреймворк для работы с PDK;
- Написан на Python;
- Технология описывается также с помощью Python;
- Работает с KLayout;
- С его помощью нам удалось сгенерировать библиотеку ячеек на 130нм PDK.

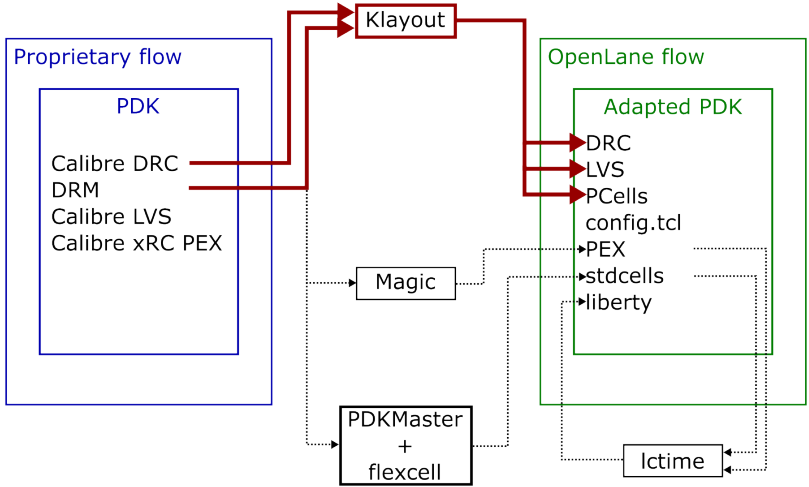
```
poly = prm.GateWire(name="poly",
    pin=pin_prims["poly"], blockage=block_prims["poly"],
    min_width=0.150, # poly.1a
    min_space=0.210, # poly.2
)
# transistors
mosgate = prm.MOSFETGate(name="mosgate",
    active=difftap, poly=poly,
    min_w=0.420, # difftap.2
    min_sd_width=0.250, # poly.7
    min_polyactive_extension=0.130, # poly.8
    contact=vias["licon"], min_contactgate_space=0.055, # licon.11
)
```

Параметризованные ячейки - утилита flexcell

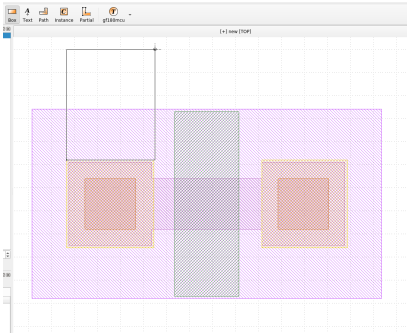




KLayout

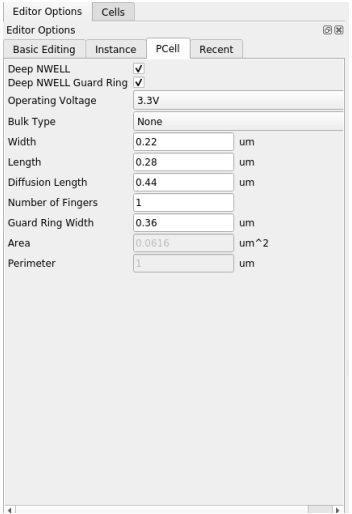

Klayout
Proprietary flow
PDK
Calibre DRC
DRM
Calibre LVS
Calibre xRC PEX
OpenLane flow
Adapted PDK
DRC
LVS
PCells
config.tcl
PEX
stdcells
liberty
Magic
PDKMaster
+
flexcell
lctime

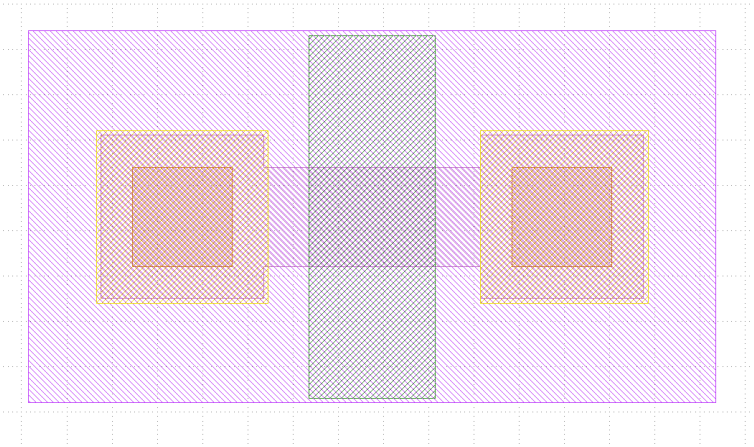
KLayout GUI & API



```
# Inserting a contact cell
cont_cell_index = layout.add_cell("contact")
cont_cell       = layout.cell(cont_cell_index)

# Inserting shapes now into the *contact* cell
cont_cell.shapes(contact).insert(pya.Box.new(0, 0, cont_size, cont_size))

# Contact array count and postions
nx = int((ld - (cont_size+cmp2cont+cont2ply))/(cont2cont+cont_size)) + 1
ny = int((w - (cont_size+2*cmp2cont))/(cont2cont+cont_size)) + 1
dx = (ld - nx * cont_size - (nx - 1) * cont2cont)*cmp2cont/cont_size
dy = (w - ny * cont_size - (ny - 1) * cont2cont)/2

# adding contact array and metals
# Left contacts
if not (w_changed == True and nf > 1) and (ld >= 440):
    cell.insert(pya.CellInstArray.new(cont_cell_index,
                pya.Trans.new(pya.Point.new(dx, dy)),
                pya.Point.new((cont2cont+cont_size), 0),
                pya.Point.new(0, (cont2cont+cont_size)), nx, ny))
    # Left metal
    cell.shapes(metal1).insert(pya.Box(-metal_violat, -metal_violat,
                               ld + metal_violat - (cont_size-2*cmp2cont),
                               w + metal_violat))
```

Параметризованные ячейки KLayout PCell

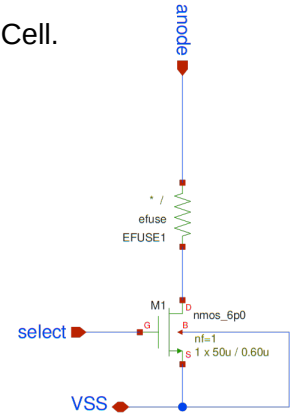




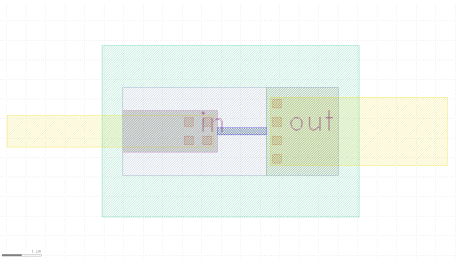
eFuse PCell

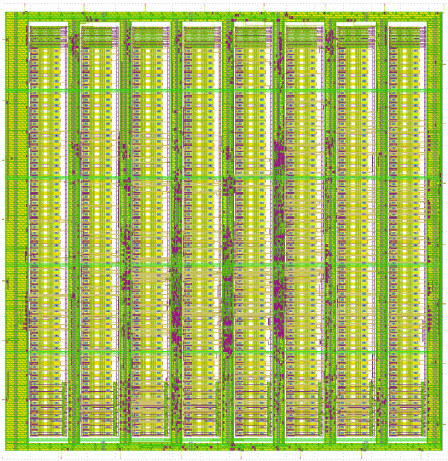
- Технология - GF180MCU;
- Площадь - 0.63 мм^2/килобит;
- Массив собран из параметризованных ячеек KLayout PCell.

```
for i in range(n_bitcells):
    # connect PMOS drain and bulk to VDD
    current_PMOS_displ = pya.Vector(PMOS_start_x+PMOS_step*i, 0)
    current_bulk_pin = pya.Point(PMOS_bulk_pin_x, PMOS_bulk_drain_pin_y)
        + current_PMOS_displ
    current_drain_pin = pya.Point(PMOS_drain_pin_x, PMOS_bulk_drain_pin_y)
        + current_PMOS_displ
    vdd_wire = pya.Box(current_bulk_pin.x-wire_width/2, vdd_rail.bottom,
                       current_drain_pin.x+wire_width/2, vss_rail.bottom-2000)
    efuse_array_cell.shapes(metal4).insert(vdd_wire)
```



eFuse PCell



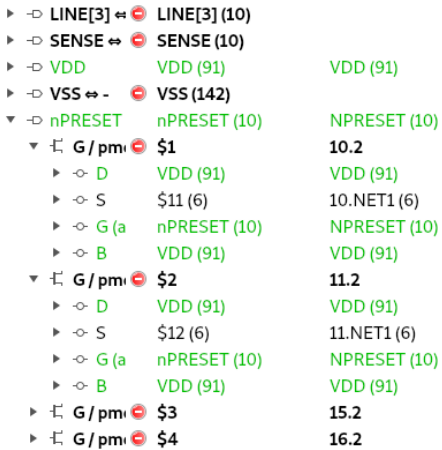


SVRF vs KLayout DRC

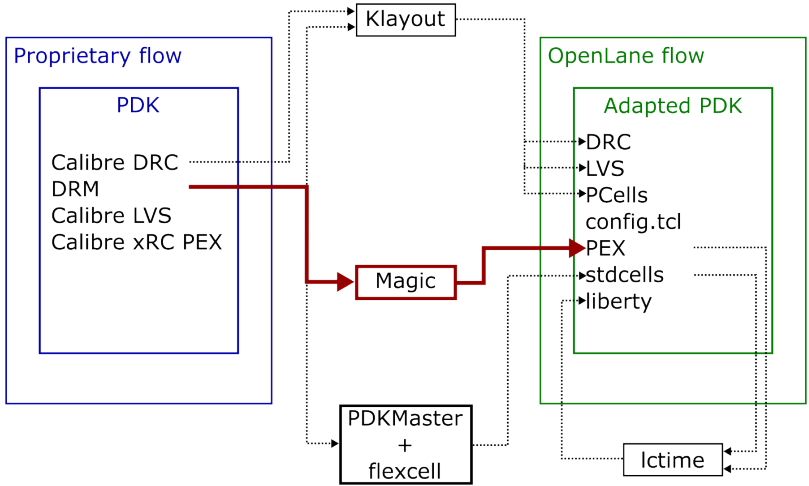
```
LAYER MAP 1 DATATYPE 0 1
LAYER L1 1
LAYER MAP 2 DATATYPE 0 2
LAYER MAP 2 DATATYPE 1 3
LAYER L2 2 3

TEST.1 {@ Min width of L1 should be 0.15
    INT L1 <0.15 ABUT<90 SINGULAR REGION
}

TEST.2 {@ Min spacing of L1 and L2 should be 0.20
    EXT L1 L2 <0.20 ABUT<90 SINGULAR REGION
}

TEST.3 {@ Min enclosure of L1 by L2 should be 0.30
    ENC L1 L2 <0.30 ABUT<90 SINGULAR REGION
}
```

```
L1 = input(1, 0)
L2 = input(2, 0) + input(2, 1)

# TEST.1
L1.width(0.15).output("TEST.1",
      "Min width of L1 should be 0.15")

# TEST.2
L1.separation(L2, 0.2).output("TEST.2",
      "Min spacing of L1 and L2 should be 0.20")

# TEST.3
L2.enclosing(L1, 0.3).output("TEST.3",
      "Min enclosure of L1 by L2 should be 0.30")
```

LVS


LINE[3] ⇔
LINE[3] (10)
SENSE ⇔
SENSE (10)
VDD
VDD (91)
VDD (91)
VSS ⇔ -
VSS (142)
nPRESET
nPRESET (10)
NPRESET (10)
G / pm
$1
10.2
D
VDD (91)
VDD (91)
S
$11 (6)
10.NET1 (6)
G (a
nPRESET (10)
NPRESET (10)
B
VDD (91)
VDD (91)
G / pm
$2
11.2
D
VDD (91)
VDD (91)
S
$12 (6)
11.NET1 (6)
G (a
nPRESET (10)
NPRESET (10)
B
VDD (91)
VDD (91)
G / pm
$3
15.2
G / pm
$4
16.2

Паразитная экстракция

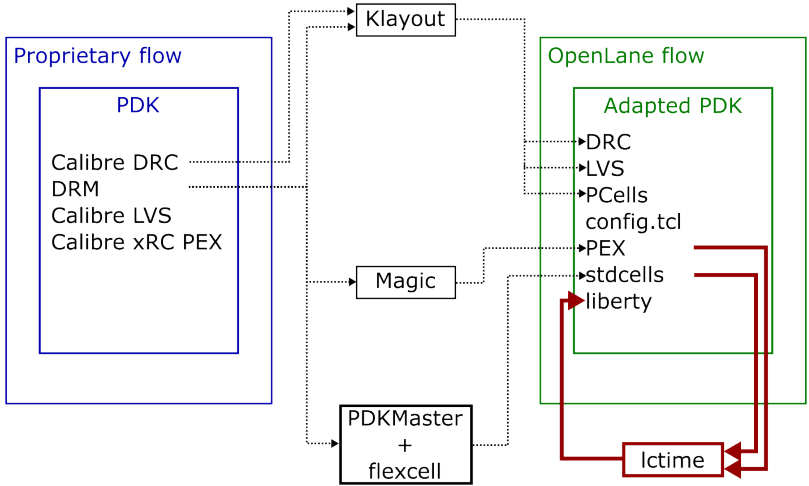


Экстракция с помощью Magic

- Magic - открытый САПР с широким набором возможностей;
- Требуется написать достаточно объемный технологический файл;
- Не требует специальных скриптов для PEX; даваемая на вход информация - сопротивления и емкости для различных сценариев;
- Для сгенерированной нами библиотеки ячеек результаты экстракции совпадают с результатами Calibre.

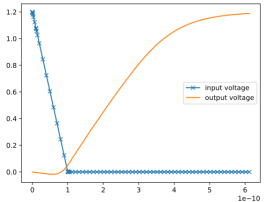
```
#metal1
defaultsidewall    allm1 metal1 36.9
defaultareacap     allm1 metal1 25.78
defaultperimeter   allm1 metal1 40.57
defaultoverlap     allm1 metal1 nwell,pwell well  25.78
defaultsideoverlap allm1 metal1 nwell,pwell well  40.57

#metal1->diff
defaultoverlap     allm1 metal1 allactivenonfet active 33.6
defaultsideoverlap allm1 metal1 allactivenonfet active 43.10
```

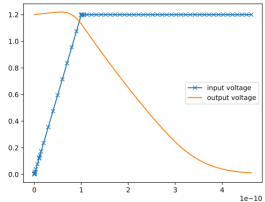
Характеризация



lctime

- Открытая утилита на Python;
- Умеет определять тип ячейки по нетлисту;
- Генерирует набор SPICE-тестов, после чего симулирует их с помощью ngspice и обрабатывает результаты;
- Позволяет рассчитывать емкости входов, времена перехода, характеристики триггеров;
- С ее помощью мы смогли отхарактеризовать большую часть ячеек из сгенерированной нами библиотеки.





Выводы

Выводы

- Открытые САПР предоставляют развитый API, позволяющий автоматизировать портирование коммерческих PDK;
- Степень абстракции достаточна для программиста на Python, поверхностно знакомого с проектированием микросхем;
- На примере нескольких PDK нами отрабатывается механизм адаптации всех их компонентов (DRC, LVS, PCell и т.д.) к открытому маршруту OpenLane;
- Нет принципиальных препятствий в адаптации любого зрелого PDK;
- Open source радикально снижает порог входа для людей из других областей разработки ПО, например ML.

Дальнейшие планы

- Тесты на более тонких процессах;
- Сгенерировать полностью открытую библиотеку ячеек на коммерческий процесс;
- Совершенствование механизмов автоматической адаптации PDK;
- Совершенствование процесса характеризации;
- Более тщательное сравнение Magic с Calibre в контексте паразитной экстракции.