

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники».

Институт микроприборов и систем управления имени Л.Н. Преснухина

Лабораторная работа №8

«Генерация топологии с помощью языка AEL»

**По курсу
«Моделирование антенно-фидерных устройств в среде Keysight Advanced Design System»**

Москва, Зеленоград

2023

Оглавление

Оглавление	2
Введение	3
Методика выполнения работы	3
Особенности языка AEL (Application Extension Language)	3
AEL-функции создания геометрии	5
Базовые функции	5
Определение контекста.....	5
Определение слоя.....	5
Определение и настройки единиц	6
Преобразования единиц углов	7
Настройки аппроксимации дуг и эллипсов	7
Функции создания примитивных объектов.....	8
Создание прямоугольника (Rectangle)	8
Создание круга (Circle).....	8
Создание эллипса (Ellipse)	9
Создание линии (Path)	9
Создание полилиний (Polyline).....	11
Последовательность создания полилиний.....	11
Группа создания дуг (отдельных полилиний).....	12
Объединение полилиний	12
Работа с полигонами.....	13
Создание полигона (Polygon)	13
Задание узлов, сглаженный окружностью.....	13
Срез узла в полигоне.....	14
Сглаживание узла в полигоне	14
Добавление узла	14
Преобразование фигур в полигон.....	14
Команды выбора и поиска фигур	15
Команды манипуляции геометрией	16
Копирование и вставка (Copy/Paste)	16
Вставка повторением (Step and Repeat)	17
Объединение (Union)	17
Вычитание (Difference)	18
Пересечение (Interception)	18
Сдвиг (Move)	18
Поворот (Rotate)	18
Команды отражения (Mirror)	19
Вырезка (Chop)	19
Обрезка (Crop)	19
Разрез (Split).....	19
Создание пинов	19
Установка компонента	20
Разбор компонента	22
Стратегия по разбивке одной функции на несколько	22
Пример универсальной функции генерации обрезанного сектора	23
Вывод информации об ошибках.....	26
Пример генерации топологии компонента с помощью языка AEL.....	28
Использование редактора VSCode.....	37
Задание на выполнение	39
Требования к отчёту	49

Задание на самостоятельную работу	50
Контрольные вопросы.....	51
Литература.....	52

Введение

Цель работы: получить навыки работы со средой моделирования Advanced Design System (ADS), изучить подходы при генерации топологии с помощью языка расширения AEL.

Используемое оборудование или ПО: материал подготовлен на основании версии Keysight Advanced Design System 2021

Продолжительность работы: 4 часа.

Методика выполнения работы

Особенности языка AEL (Application Extension Language)

Язык AEL это C-подобный язык, с некоторыми особенностями.

В AEL нет препроцессора, т.е. директивы типа `#include`, `#if`, `#ifdef` и пр. не поддерживаются. Однако есть функция `load()`, позволяющая загрузить в текущее пространство имен файл с нужными функциями;

Переменным при определении не задается тип, он выводится из присвоения.

Поддерживаются стандартные типы данных `long`, `double`, `string` и `boolean`.

Поддерживаются комплексные числа (`decl x = 1+3i;`).

Поддерживаются списки (`list`). Индексация в списках идет с 0. Обращение к элементам списка в прямоугольных скобках `[]`.

Структуры не поддерживаются.

Для возвращения результата из функции используется ключевое слово `return` Если функция ничего не возвращает, то `return` можно не ставить.

Переменные могут быть определены глобально, локально для функции или локально для блока. Локальные переменные объявляются с ключевым словом `decl`

Доступны циклы `while` ___ и условные выражения `if` ___ `elseif` ___ `else` ___ и `switch` ___ `case` ___.

АЕЛ язык со сборщиком мусора, управление памятью не доступно.

Комментарии объявляются либо как `//` для строки или `/* */` для блока строк.

В *.ael файле может быть много АЕЛ скриптов (функций), каждый должен быть объявлен с ключевым словом `defun`

Внутренность блоков кода (функция, цикл, логическое ветвление) должна быть обрамлена фигурными скобками `{ }`

Каждая строка должна заканчиваться точкой с запятой ;

Присутствует несколько predefined констант:

Таблица 1. Предопределённые константы

Константа	Значение
<code>TRUE</code>	boolean true
<code>FALSE</code>	boolean false
<code>stdin</code>	standard in
<code>stdout</code>	standard out
<code>stderr</code>	standard error
<code>hugeReal</code>	3.4e +38
<code>tinyReal</code>	2.2e -308
<code>e</code>	2.718281828
<code>ln10</code>	ln(10) 2.302585093
<code>c0</code>	speed of light 2.997 924 58 e+08 m/s
<code>e0</code>	vacuum permittivity 8.8541878176204e-12 F/m
<code>u0</code>	vacuum permeability 1.2566370614359e-06 H/m
<code>boltzmann</code>	Boltzmann's constant 1.380658e-23 J/K
<code>qelectron</code>	charge of an electron 1.60217733e-19 C
<code>planck</code>	Planck's constant 6.6260755e-34 J-sec
<code>PI</code>	= PI = 3.1415926535898
<code>on_PC</code>	TRUE if running on PC, else FALSE

АЕL-функции создания геометрии

В данном разделе перечислены некоторые встроенные функции, которые можно применять при генерации топологии с помощью языка АЕL. Данный список не в коем случае не претендует на всеобщий охват, но позволяет вычлениить из всего огромного списка доступных функций некоторый исходный базис.

Все доступные АЕL-функции обобщенно можно условно поделить на два класса:

- Работающие в пределах активного контекста (находящейся в фокусе открытой схемы, топологии и пр.). Фактически имитирующие нажатия мыши и выполнения команд – как правило начинаются с префикса «**de_**».

- Проводящие манипуляции в пределах указанного контекста, необязательно находящегося в фокусе. Как правило начинаются с префикса «**db_**».

Часть функций имеет двойников в другом классе, часть нет.

Нужно следить за тем, что возвращает функция. Например, функция создания прямоугольника `de_add_rectangle()` возвращает ссылку на созданную фигуру, а функция `de_union()` – возвращает `TRUE` или `FALSE` в зависимости, удалось ли объединить объекты.

Базовые функции

Определение контекста

Для получения текущего контекста используется команда

```
decl context = de_get_current_design_context();
```

Определение слоя

Часть функций требует указания слоя, в котором идет манипуляция.

Получить ссылку на желаемый слой из контекста можно по команде

```
decl layerid = db_get_layerid(context, layerName [, purposeName]);
```

`context` – нужный контекст

`layerName` – имя слоя

`purposeName` - назначение слоя, опционально (по умолчанию принимается назначение "drawing" или -1).

```
decl cond_layerID = db_get_layerid(context, "cond", "drawing");
```

Для функций, которые не имеют среди аргументов номер слоя, нужно сначала перейти на желаемый слой по команде

```
db_set_entry_layerid(context, layerId);
```

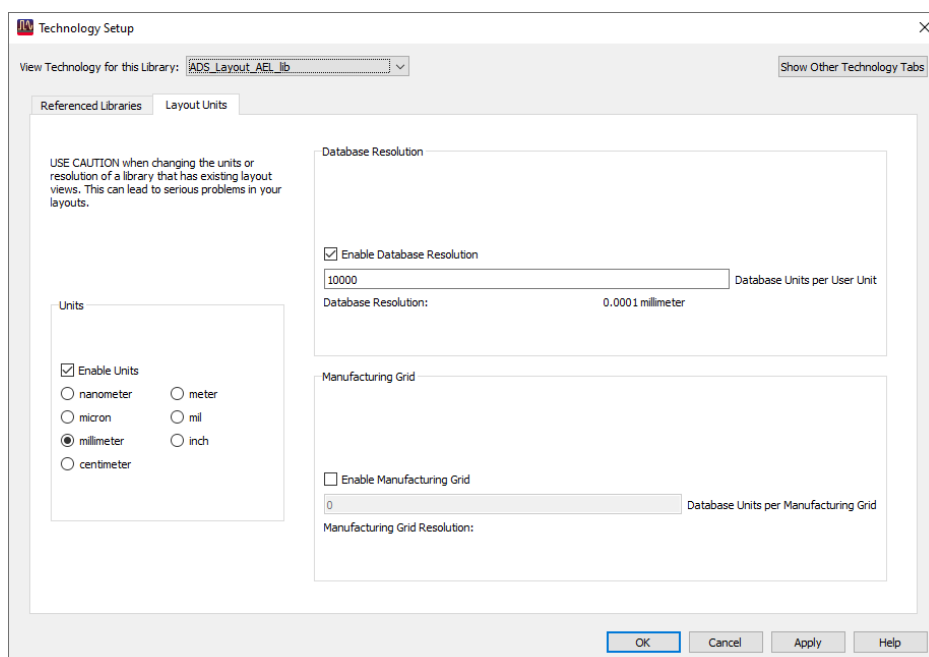
где `layerId` – ссылка на слой

Определение и настройки единиц

Для единиц AEL-функции могут работать с тремя определениями:

- mks – система СИ, для длин это всегда метры;
- uu – user units, пользовательские единицы технологии библиотеки проекта;
- dbu – database unit, единицы разрешения технологии библиотеки проекта.

Единицы uu и dbu задаются при создании технологии



Если в параметрах ячейки указывать размерам длин их тип Length, то в аргументы AEL-функции они передаются в системе msk (СИ). Большая часть функций (если в справке не указано иное) принимает размеры в единицах технологии (uu).

Можно промасштабировать длины из mks в uu с помощью множителя `mks2uu`. Получить этот множитель из текущего контекста можно с помощью команды

```
decl mks2uu = db_get_mks_to_uu_factor(context);
```

Затем аргументы длин надо умножить на этот множитель.

```
rect_width = rect_width * mks2uu;
```

Также доступны функции получения и остальных множителей

```
decl uu2mks = db_get_mks_to_uu_factor(context);
```

```
decl dbu2uu = db_get_dbu_to_uu_factor(context);
```

```
decl uu2dbu = db_get_uu_to_dbu_factor(context);
```

Преобразования единиц углов

Значения углов (тип Angle) передаются в градусах. Часть функций принимает углы в радианах, их можно преобразовать с помощью функции

```
decl alpha = rad(alpha);
```

Также есть и обратное преобразование `deg()`;

Настройки аппроксимации дуг и эллипсов

При преобразовании дуг и эллипсов в полигон (в том, числе при повороте на угол, некратный 90°, или операциях объединения и пр.), дуги и эллипсы аппроксимируются прямыми с заданным угловым шагом. Глобально для текущего контекста можно выставить данное значение по команде

```
de_set_resolution_for_arc(degree);
```

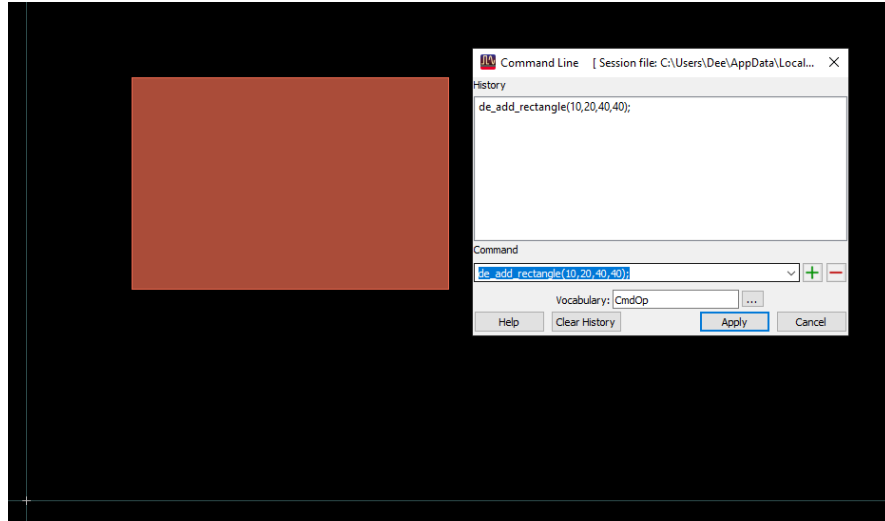
`degree` – угловое разрешение, в градусах.

Функции создания примитивных объектов

Создание прямоугольника (*Rectangle*)

```
decl shRectangle = de_add_rectangle(x1,y1, x2,y2);
```

x_1, y_1 и x_2, y_2 – координаты противоположных углов прямоугольника



Есть аналог с указанием контекста

```
decl shRectangle = db_add_rectangle(context, layerid, x1,y1, x2,y2);
```

Создание круга (*Circle*)

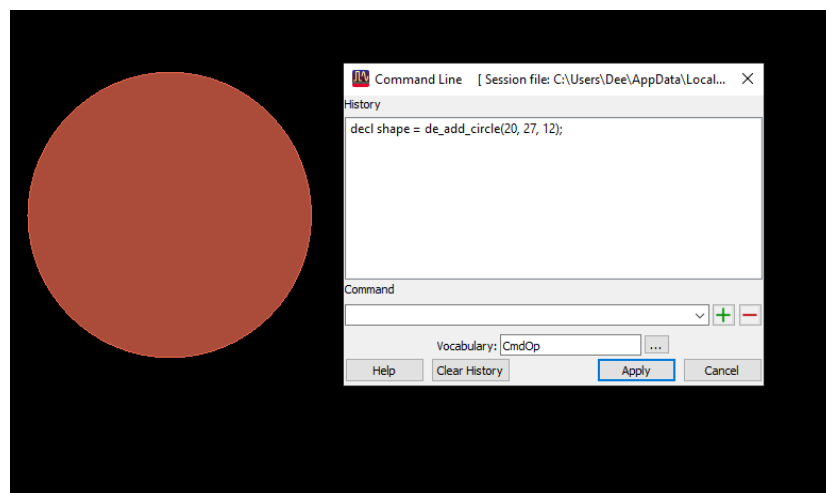
```
decl shCircle = de_add_circle(x,y, radius);
```

x, y – центр круга

radius - радиус

Есть аналог с указанием контекста

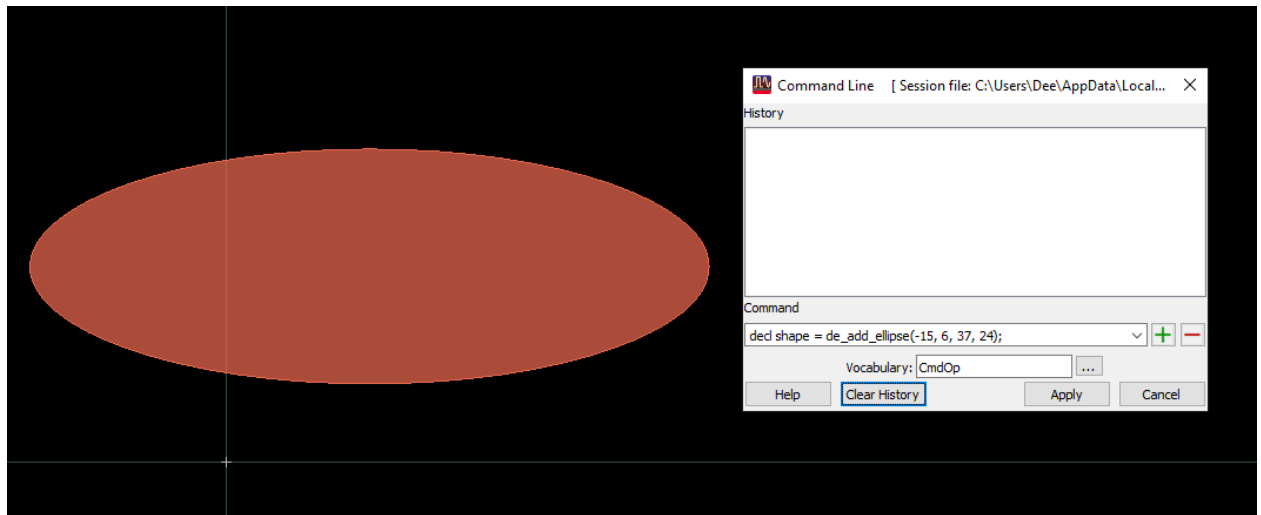
```
decl dbShape = db_add_circle(context, layerid, x1, y1, radius);
```



Создание эллипса (Ellipse)

```
decl shEllipse = de_add_ellipse(x1, y1, x2, y2);
```

`x1, y1` и `x2, y2` – координаты противоположных углов прямоугольника в единицах технологии, описанного вокруг эллипса. Эллипс может быть расположен только вдоль оси X или Y, при повороте на угол, отличный от 90°, превращается в полигон, аппроксимированный прямыми.



Создание линии (Path)

Path – это линия, имеющая заданную ширину, стили углов и концов. Задается по точкам. Для создания линии нужно выполнить несколько команд:

Задать ширину линии

```
db_set_path_width(context, width);
```

Определить стиль углов

```
db_set_path_corner(context, cornerType);
```

где `cornerType` может принимать три значения:

`DB_MITERED_CORNER` – срезанный угол

`DB_SQUARE_CORNER` – прямой угол

`DB_CURVED_CORNER` – сглаженный дугой угол

Для режима `DB_MITERED_CORNER` отдельно можно задать процент среза угла

```
db_set_miter_cutoff(context, miterRatio);
```

`miterRatio` – процент среза угла

Для режима `DB_CURVED_CORNER` отдельно можно задать радиус сглаживания

```
db_set_curve_radius(context, radius);
```

`radius` – радиус сглаживания

Стиль концов линии задается по команде

```
db_set_path_endcap(context, endCapType);
```

где `endCapType` может принимать три значения:

`DB_SQUARE_CAP` – обрезается ровно по точке

`DB_ROUND_CAP` – к концу добавляется полуокружность

`DB_SQUARE_CAP_THAT_EXTENDS_HALF_WIDTH` – к концу добавляется

половина ширины линии.

После задания этих настроек надо запустить создание линии по команде

```
db_add_path(context);
```

Далее нужно задать точки линии по нескольким вызовам команды

```
decl bOk = db_add_point(context, x, y);
```

Окончательно в конце по команде заканчиваем создание линии. Также нужно указать в какой слой пойдет линия и можно получить ссылку на нее.

```
decl dbShape = db_end(context, layerId);
```

Пример

```
db_set_path_width(context, 10);
```

```
db_set_path_corner(context, DB_CURVED_CORNER);
```

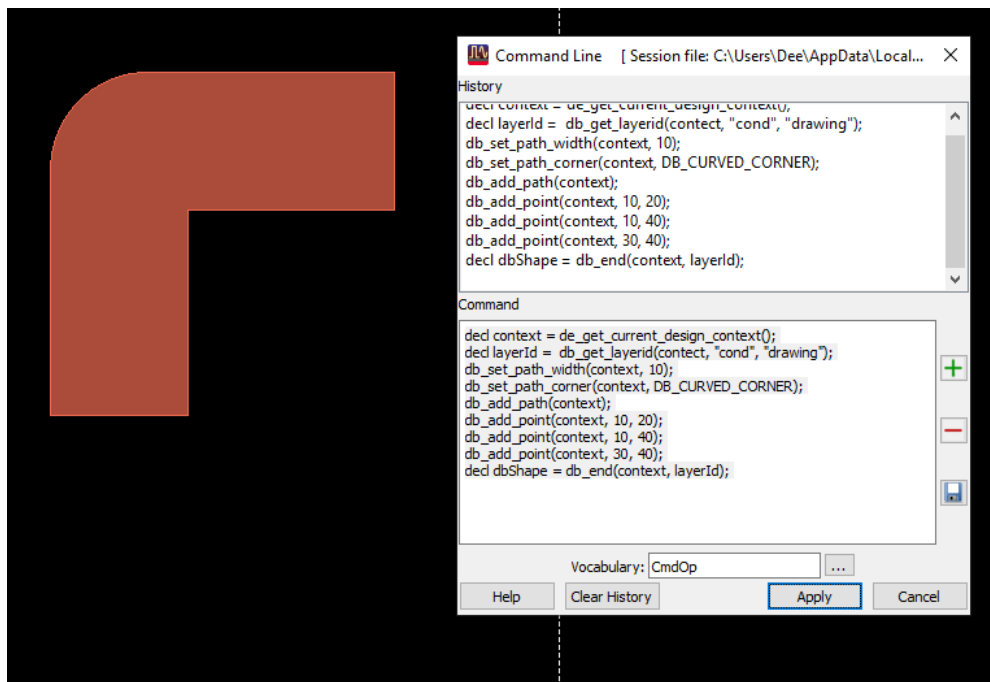
```
db_add_path(context);
```

```
db_add_point(context, 10, 20);
```

```
db_add_point(context, 10, 40);
```

```
db_add_point(context, 30, 40);
```

```
decl dbShape = db_end(context, layerId);
```



Создание линии также возможно по команде без указания активного контекста `de_add_path()`

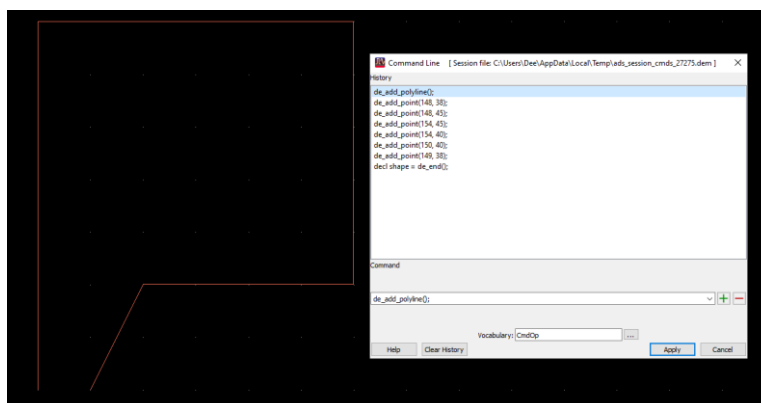
Создание полилиний (Polyline)

Последовательность создания полилиний

Полилинии создаются аналогичного Path.

Сначала запускается процедура создания полилинии, затем по точкам рисуется полилиния и окончательно завершается ее создание

```
db_add_polyline(context);
db_add_point(context,0,0);
db_add_point(context,10,20);
decl dbShape = db_end(context,layerId);
```



Группа создания дуг (отдельных полилиний)

Присутствует группа команд, которая позволяет создавать отдельные дуги.

`de_add_arc1(x1, y1, x2, y2, x3, y3);`

Режим начальная точка ($x1, y1$), касательная точка ($x2, y2$) и конечная точка ($x3, y3$) дуги.

`de_add_arc2(x1, y1, x2, y2, x3, y3, direction);`

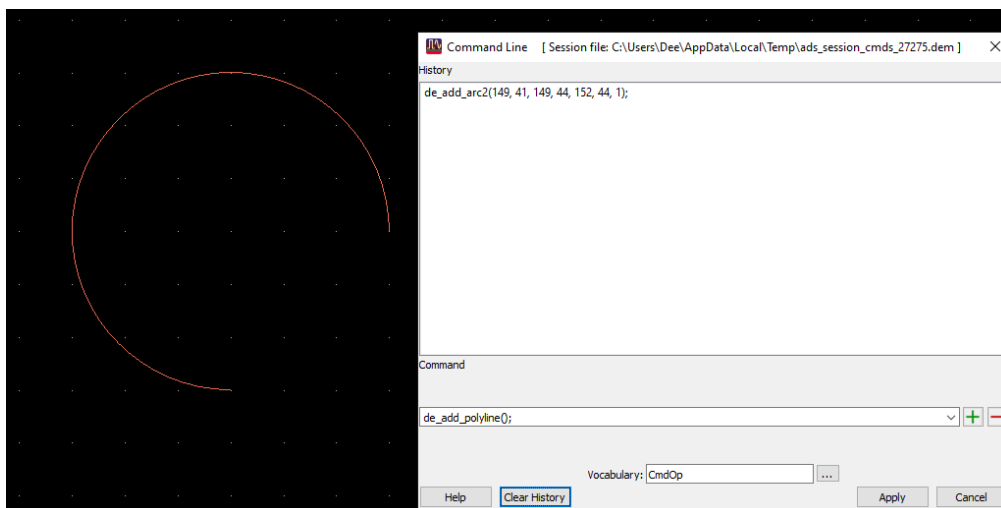
Режим начальная точка ($x1, y1$), центр дуги ($x2, y2$), конечная точка ($x3, y3$) дуги и направление `direction` (1 – по часовой стрелке, 0 - против).

`de_add_arc3(x1, y1, x2, y2, sweepAngle, direction);`

Режим начальная точка ($x1, y1$), центр дуги ($x2, y2$), угловой размер дуги `sweepAngle` (в град) и направление `direction` (1 – по часовой стрелке, 0 – против).

`de_add_arc4(x1, y1, x2, y2, chordLength, direction);`

Режим начальная точка ($x1, y1$), центр дуги ($x2, y2$), длина дуги `chordLength` и направление `direction` (1 – по часовой стрелке, 0 – против).



Объединение полилиний

Объединение выполняется по команде. Если полилиния формирует замкнутый контур, то образуется полигон.

```
de_modify_join([x,y]);
```

[x, y] – опционально, точка в пределах объединяемых полилиний.

Работа с полигонами

Создание полигона (Polygon)

По команде инициализируется создание полигона

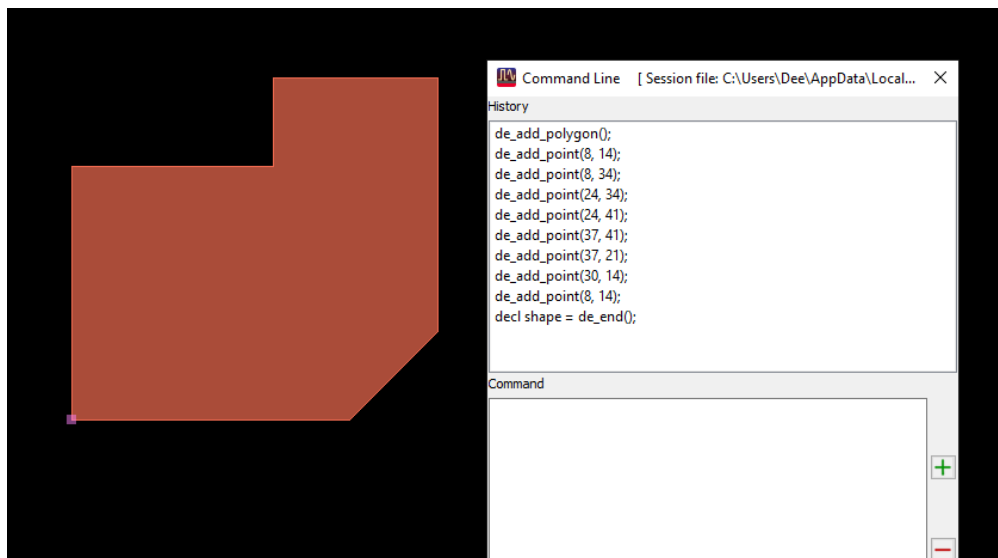
```
de_add_polygon();
```

Затем с помощью многократного вызова команды задаются узлы полигона

```
de_add_point(x, y);
```

Окончательно в конце по команде заканчиваем создание полигона и получаем ссылку на созданный полигон.

```
decl shPolygon = de_end();
```



Задание узлов, сглаженный окружностью

В полигоне при его создании можно определять узлы, сглаженные окружностью. Задается во время определения точек полигона следующей командой

```
de_add_arc(x,y, angle);
```

Задается центр дуги (x, y) и угловой размер дуги angle (в град, положительное против часовой стрелки, отрицательное – по часовой стрелке).

Радиус определяется автоматически относительно предыдущей точки в полигоне.

Срез узла в полигоне

У созданного ранее полигона можно срезать узел по команде

```
de_miter_vertex([x,y]);
```

[x, y] – опциональны, можно заранее выбрать точку.

Значение среза (в ui) надо предварительно определить по команде

```
de_set_miter_length(length);
```

Сглаживание узла в полигоне

Сглаживание узла по команде

```
de_vertex_to_arc([x,y]);
```

[x, y] – опциональны, можно заранее выбрать точку.

Значение дуги (в ui) надо предварительно определить по команде

```
de_set_arc_radius(radius);
```

Добавление узла

В созданный полигон можно добавить узел по команде

```
de_add_vertex(x, y, nx, ny);
```

x, y - точка на ребре, куда будет добавлен новый узел

nx, ny – координаты расположения нового узла

Преобразование фигур в полигон

Любую созданную фигуру можно превратить в полигон по команде

```
de_convert_to_polygon([x,y]);
```

[x,y] – точка в пределах нужной фигуры, опциональна, можно заранее выбрать фигуру.

Также автоматически преобразуются в полигон все фигуры, которые нельзя представить в исходных определениях, в том числе результаты булевых операций, повернутые на углы, не кратные 90° и пр.

Команды выбора и поиска фигур

Часть команд принимают не указатели на фигуры, а работают с заранее выбранными объектами.

Для выбора объекта по указателю используется функция

```
db_select(object);
```

Для выбора всех объектов в окне используется команда

```
de_select_all();
```

или

```
db_select_all(context);
```

При этом учитываются настроенные фильтры выбора.

Также можно выбирать все объекты на слое по командам

```
de_select_all_on_layer(layerNameOrNum);
```

`layerNameOrNum` – номер или имя слоя

или

```
db_select_all_on_layerid(context, layerId);
```

`layerId` – ID слоя

Часть команд после своей работы оставляют выбранными обработанные объекты. Для сброса выбора можно пользоваться командами

```
db_deselect_all(context);
```

или

```
de_deselect_all();
```

Если из множества выбранных объектов нужно сбросить выделение только одного, то это можно выполнить по команде

```
db_select(object, FALSE);
```

В случае, если после каких-то манипуляций потерян указатель на нужную фигуру, но известно, что по какой-то координате одна есть, то можно получить указатель на фигуру по команде ниже. Данная команда фактически имитирует клик мышкой. Если в этой координате несколько объектов, то они будут перебираться последовательно при каждом вызове команды.

```
decl shapehandle=db_pick_shape_at(context, x, y, tolerance);
```

x, y – координаты

context – текущий контекст

tolerance – точность выбора. Обычно точность привязана к текущему окну, получить ее можно парой вызовов

```
decl winInst = api_get_current_window();
```

```
decl tolerance = db_get_selection_tolerance(winInst);
```

Если результат после какой-то операции, не вернувшей новый указатель, оставившей объект выбранным, нужен как указатель, то можно этот указатель получить через итератор с ограничением на выбранные объекты.

```
decl iter = db_create_inst_iter(context);
```

```
iter = db_inst_iter_limit_selected(iter);
```

Команды манипуляции геометрии

Все они работают над предварительно выбранными фигурами.

Копирование и вставка (Copy/Paste)

```
de_copy(dx,dy);
```

Копирует выделенные объекты и располагает на смещении *dx, dy* от оригинала.

Также возможно сначала скопировать выделенное в буфер, а затем вставить.

```
de_copy_to_buffer([x,y]);
```


[x, y] – опционально, референсная точка, относительно которой будет сохранено в буфер. Если не задано, то сохраняется относительно первого неподсоединённого пина или нижнего левого угла окружающего прямоугольника.

Вставка по команде

```
de_paste_from_buffer(x, y[, angle, mirror]);
```

x, y – точка для вставки

angle – угол поворота перед вставкой

mirror – режим отзеркаливания перед вставкой, может быть MIRROR_X или MIRROR_Y

Порядок действий – сначала отзеркаливание, затем поворот, затем вставка.

Вставка повторением (Step and Repeat)

Выполняет по команде

```
de_step_and_repeat(x,y);
```

x, y – опорная точка для вставки

перед выполнением надо настроить по команде

```
de_set_step_and_repeat(xSpacing, ySpacing, numRows, numCols[, connectFlag]);
```

xSpacing, ySpacing – шаг между границами (окружающими прямоугольниками) копий

numRows, numCols – число строк и колонок

connectFlag – накладываемые объекты будут соединены в одинаковые цепи

Объединение (Union)

Выбранные объекты можно объединить. Результатом является полигон. Перед вызовом объекты надо выбрать. Команда объединения

```
de_union();
```

Возвращает TRUE или FALSE в зависимости от успеха операции.
Созданный объект остается выбранным.

Вычитание (Difference)

Создает несколько один или несколько полигонов, равных разности исходных фигур. Команда разности

`de_difference();`

Возвращает TRUE или FALSE в зависимости от успеха операции.
Созданные объекты остаются выбранными.

Пересечение (Interception)

Создает несколько один или несколько полигонов, равных пересечению исходных фигур. Команда пересечения

`de_intersection();`

Возвращает TRUE или FALSE в зависимости от успеха операции.
Созданные объекты остаются выбранными.

Сдвиг (Move)

`de_move(dx, dy);`

Сдвигает заранее выбранное на значение `dx, dy` в `u`

Объекты остаются выбранными

Поворот (Rotate)

`de_rotate(x,y, angle[, keepConnected]);`

Поворачивает на угол `angle` вокруг точки `x, y`

`keepConnected` – TRUE или FALSE, опционально, сохраняет присоединенные цепи.

Команды отражения (Mirror)

`de_mirror_x(x,y);`

Отражает относительно горизонтальной оси, проходящей через точку `x,y`

`de_mirror_y(x,y);`

Отражает относительно вертикальной оси, проходящей через точку `x,y`

Вырезка (Chop)

Вырезка прямоугольного участка из выбранных объектов.

`de_chop(x1, y1, x2, y2);`

`x1, y1` и `x2, y2` – координаты противоположных углов вырезающего прямоугольника. Остатки остаются выбранными.

Обрезка (Crop)

Обрезка прямоугольной границей выбранных объектов

`de_crop(x1, y1, x2, y2);`

`x1, y1` и `x2, y2` – координаты противоположных углов обрезающего прямоугольника. Остатки остаются выбранными

Разрез (Split)

Разрез прямоугольной границей выбранных объектов

`de_split(x1, y1, x2, y2);`

`x1, y1` и `x2, y2` – координаты противоположных углов разрезающего прямоугольника. Остатки остаются выбранными

Создание пинов

Делается по команде. Возвращает ссылку на созданный пин.

`decl newPin = db_create_pin(context, x, y [, angle, layerId, termNum, termName, termType]);`

`context` – контекст, в который нужно добавить пин

`x, y` – координаты пина

`angle` – направление, куда смотрит пин. Если параметр не задан, направление рассчитывается автоматически;

`layerId` – слой, в который добавляется пин. Если параметр не задан, пин добавляется в текущий слой контекста

`termNum` – номер соединения. Если не задан, то присваивается с инкрементом к предыдущему.

`termName` – имя соединения. Если не задано, то присваивается автоматически.

`termType` – тип пина. Может быть

`INPUT_PIN = 0` – входной пин

`OUTPUT_PIN = 1` – выходной пин

`IN_OUT_PIN = 2` – двунаправленный пин

По умолчанию присваивается тип `IN_OUT_PIN`.

Установка компонента

Делается в несколько этапов:

1. Инициализация компонента и поиск по библиотекам.

```
decl itemInfoToPlace = de_init_item(itemName);
```

`itemName` – строка с обозначением компонента. Может быть краткой `"C_Pad1"`, может быть полной `"ads_rflib:C_Pad1:layout"`

2. Задание параметров

```
de_set_item_parameters(itemInfoToPlace, listOfParameters);
```

`itemInfoToPlace` – ссылка на устанавливаемый компонент

`listOfParameters` – полный список параметров. Задавать нужно все параметры, даже те, которые не планируется изменять. Формируется с помощью сборки параметров в список `list()`. Отдельные параметры создаются с помощью функций `prm()` или `prm_ex()`, которые используют подтип `StdForm`.

3. Установка компонента в топологию.

```
de_place_item(itemInfoToPlace, xLoc, yLoc);
```

`itemInfoToPlace` – ссылка на устанавливаемый компонент

`xLoc`, `yLoc` – координаты референсной точки компонента. У большинства стандартных компонентов референсной точкой является первый пин.

Перед установкой можно повернуть шаблон компонента по команде `de_rotate_inc()`;

Поворот идет по часовой стрелке с инкрементом угла, определенным для текущего контекста, который можно отдельно установить по команде `de_set_rotation_increment(angle)`;

На данном этапе можно установить много однотипных компонентов.

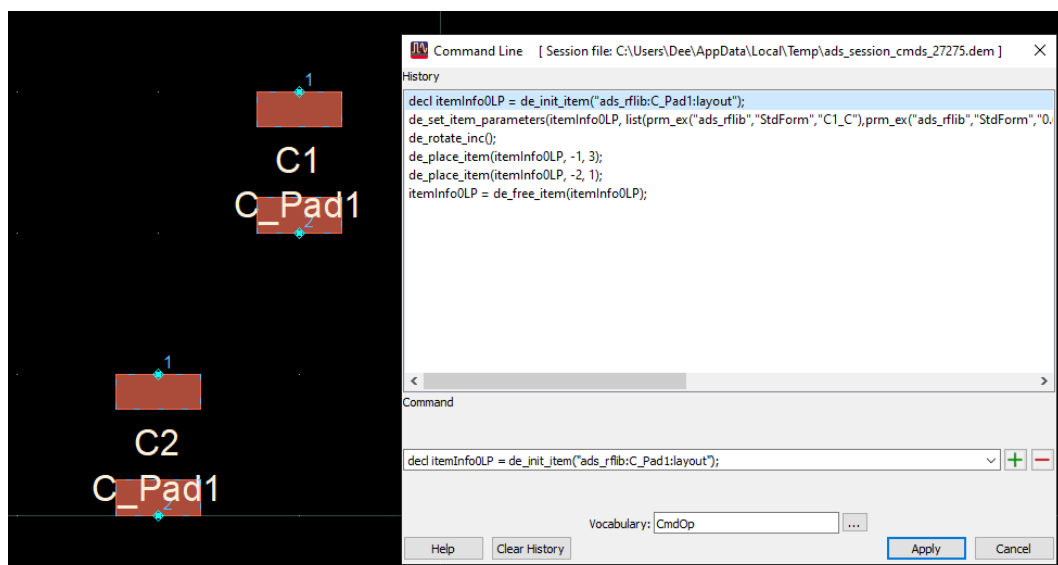
4. Окончательно выйти из режима установки компонентов

```
itemInfoToPlace = de_free_item(itemInfoToPlace);
```

`de_free_item()` возвращает `NULL`, есть рекомендация присваивать использованному указателю `itemInfoToPlace` `NULL` после окончания.

Пример:

```
decl itemInfo1LP = de_init_item("ads_rflib:C_Pad1:layout");
de_set_item_parameters(itemInfo1LP,
list(prm_ex("ads_rflib","StdForm","C1_C"),prm_ex("ads_rflib","StdForm","0.6
mm"),prm_ex("ads_rflib","StdForm","0.25 mm"),prm_ex("ads_rflib","StdForm","1
mm"),prm_ex("ads_rflib","layer_text_form","\cond:drawing\"))) );
de_rotate_inc();
de_place_item(itemInfo1LP, -1, 3);
de_place_item(itemInfo1LP, -2, 1);
itemInfo1LP = de_free_item(itemInfo1LP);
```



Разбор компонента

Установленный и выбранный компонент можно разобрать на примитивы по команде

```
de_flatten();
```

Стратегия по разбивке одной функции на несколько

Если при разработке AEL-функции по созданию топологии она получается слишком большой, и при этом присутствуют повторяющиеся участки геометрии, то более эффективной стратегией становится вынесение повторяющихся частей в отдельную AEL-функцию. При этом как правило желательно, чтобы эту функцию тоже можно было свободно привязывать к какой-либо другой ячейке.

Для сохранения такой переносимости все внутренние AEL-функции должны принимать аргументы длин в mks. Здесь в зависимости от структуры родительской AEL-функции верхнего уровня возможно предложить несколько вариантов:

- Пусть все внутренние вызовы пользовательских AEL-функций проходят в начале родительской AEL-функции, последующие манипуляции проходят потом. В этом случае нормировку к пользовательским единицам ($mks \rightarrow uu$) лучше проводить после всех внутренних вызовов.

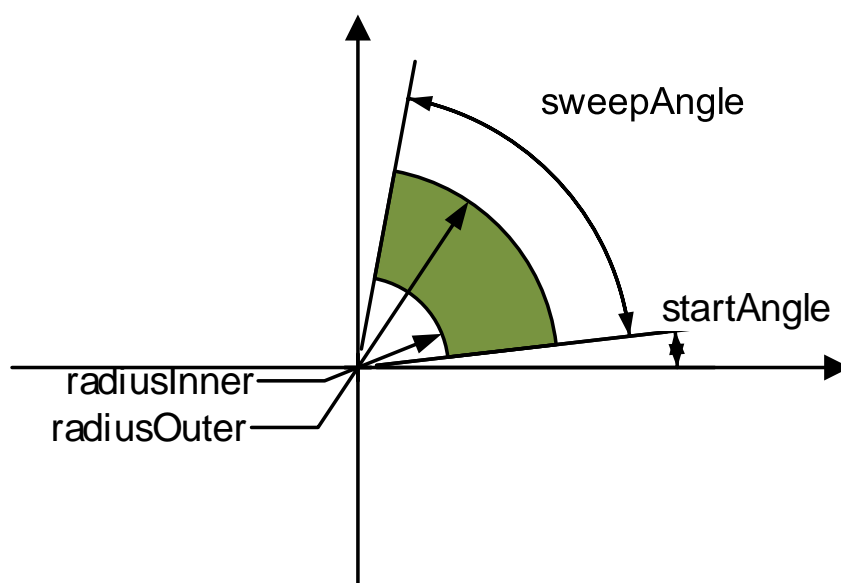
- В родительской функции сначала проводятся некоторые манипуляции, а потом вызываются все внутренние пользовательских AEL-функции. В этом случае, в начале родительской функции нужно провести нормировку $mks \rightarrow uu$, провести нужные манипуляции, а перед вызовом внутренних пользовательских AEL-функций провести обратную перенормировку $uu \rightarrow mks$.

- Если порядок вызова смешанный, то лучше всего в самом начале создать копию параметров в пользовательских единицах (и добавить им

суффикс `_ui`; переменным в формате `mks` суффиксы добавлять не стоит, т.к. эти имена этих переменных видны при подключении `ael`-функции к ячейке). При манипуляциях в пределах родительской `AEL`-функции использовать переменные с суффиксом `_ui`, а внутренние пользовательских `AEL`-функции вызывать относительно переменных в формате `mks`. Основная сложность такого подхода – надо следить, чтобы переменные в формате `mks` и `ui` были синхронны по значению.

Пример универсальной функции генерации обрезанного сектора

Например, для формирования сложных топологий полезно иметь функцию создания вырезанного сектора (ее нет в стандартных) с возвратом ссылки на созданную фигуру. Но при этом еще хочется иметь возможность привязывать эту функцию к ячейке.



Назовем данную функцию `prim_pie()`. Определим для нее следующие аргументы:

`centerX`, `center Y` – координаты центра сектора

`radiusInner` – радиус выреза

`radiusOuter` – внешний радиус

`startAngle` – угловое положение начала сектора, отсчитывается относительно положительного направления оси X против часовой стрелки

`sweepAngle` – угловой размер сектора. Если больше нуля, то сектор строится против часовой стрелки; если меньше нуля – то по часовой стрелке.

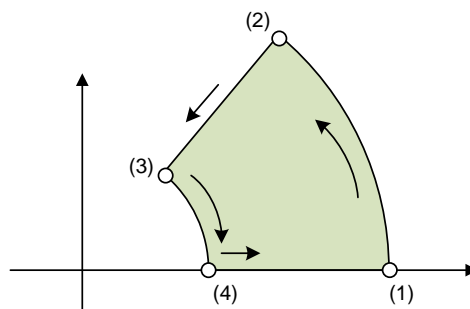
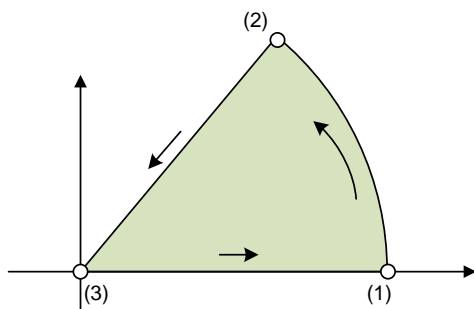
Т.к. мы хотим, чтобы эта функция была универсальна, то размеры она принимает в mks. Перенормируем размеры длин в uu

```
defun prim_pie(centerX, centerY, radiusInner, radiusOuter, startAngle,
sweepAngle)
{
  decl designContext = de_get_current_design_context();
  decl cond_layerID = db_get_layerid(designContext, "cond", "drawing");
  decl mks2uu = db_get_mks_to_uu_factor(designContext);
  centerX = centerX * mks2uu;
  centerY = centerY * mks2uu;
  radiusInner = radiusInner * mks2uu;
  radiusOuter = radiusOuter * mks2uu;
```

Если внутренний радиус больше внешнего, то геометрия будет строиться, но с некорректным поведением, поэтому создадим предупреждение. Также внешний радиус должен быть больше 0, а внутренний больше или равен 0

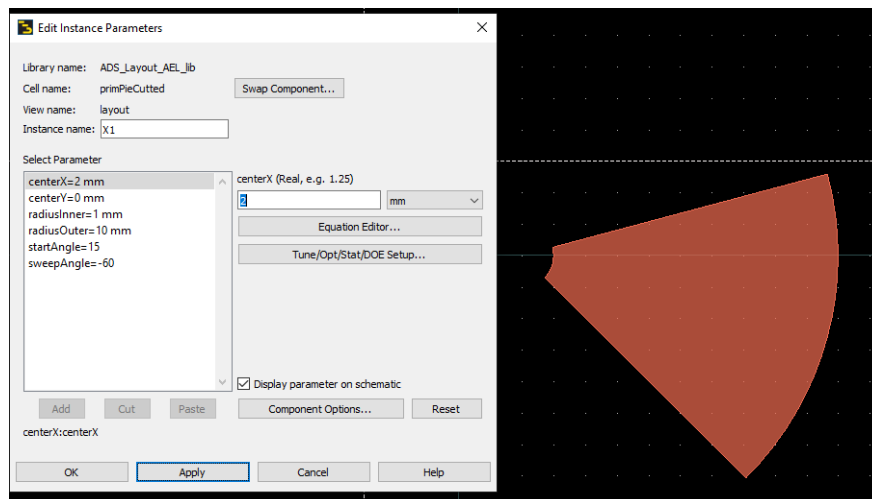
```
  if (radiusInner > radiusOuter)
    warning(stderr, 1, "radiusInner > radiusOuter", "Parameters Error");
  if (radiusInner < 0)
    warning(stderr, 2, "radiusInner < 0", "Parameters Error");
  if (radiusOuter <= 0)
    warning(stderr, 3, "radiusOuter <= 0", "Parameters Error");
```

Создадим сектор как полигон. Если `radiusInner` больше 0, то создадим еще внутренний вырез. Пусть полигон создается по следующим точками



```
//create Pie
de_add_polygon();
de_add_point(radiusOuter, 0);
de_add_arc(0, 0, sweepAngle);
// if radiusInner > 0 cut inner pie
if (radiusInner > 0)
{
    de_add_point(radiusInner * cos(rad(sweepAngle)), radiusInner *
sin(rad(sweepAngle)));
    de_add_arc(0, 0, -sweepAngle);
}
else
    de_add_point(0, 0);
decl shPie = de_end();
Окончательно повернём на нужный угол и перенесем в нужную точку.
de_select_all();
de_rotate(0, 0, startAngle);
de_move(centerX, centerY);
return shPie;
}
```

Теперь эту функцию можно привязывать к ячейке, для генерации вырезанного сектора.



Также эту функцию можно вызывать из других функций. Нужно помнить два ограничения:

- эта функция должна быть определена (или загружена) в ael-файле до вызывающей функции;
- размеры длин в нее должны передаваться в mks (метрах).

Вывод информации об ошибках

В случае, если генерируемая топология довольно сложна и есть взаимосвязанные переменные, то перед началом построений можно провести проверки.

Для вывода информации об ошибках и предупреждениях служат пару функций

`error(errorFileName, errorNum, defaultErrString, infoString);`

и

`warning(errorFileName, errorNum, defaultErrString, infoString);`

`errorFileName` — файл, в который отправится информация. Для вывода сообщений об ошибках в окне Messages ошибок используется `stderr`

`errorNum` — номер строки, в которую пишется инфо об ошибках, позволяет перезаписать однотипную ошибку.

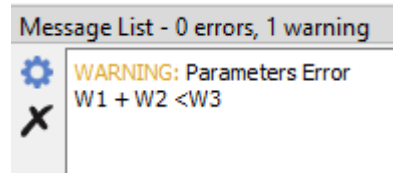
`defaultErrString` — полная строка с информацией об ошибке. Можно собрать сложную строку с помощью функций работы со строками `fmt_tokens();` и `strcat();`.

`infoString` – заголовок об ошибке.

Пример:

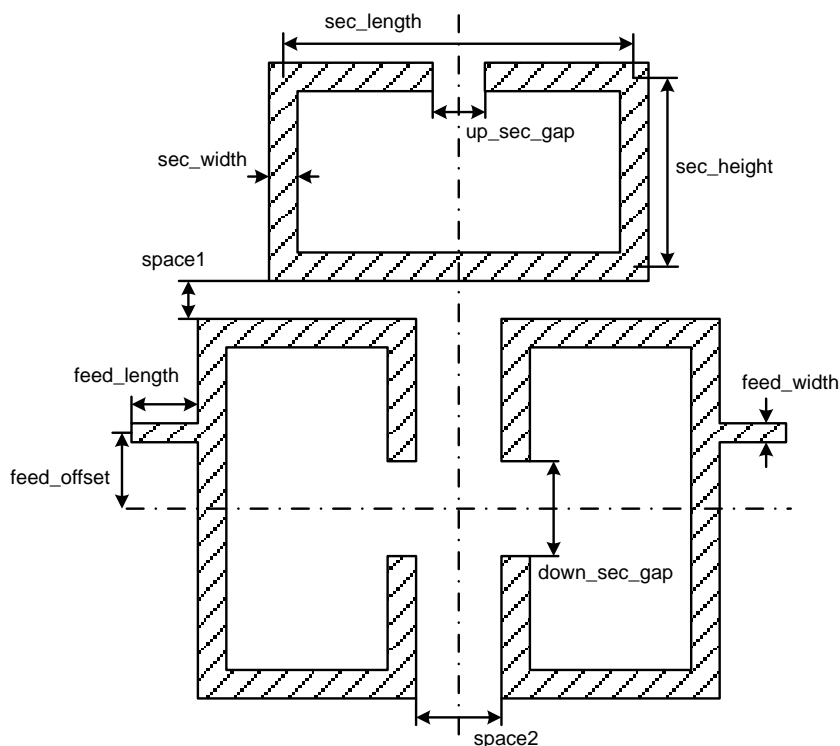
```
if (W1 + W2 < W3)
```

```
    warning(stderr, 20, fmt_tokens(list(errList, "W1 + W2 <W3")), "Parameters  
Error");
```



Пример генерации топологии компонента с помощью языка AEL

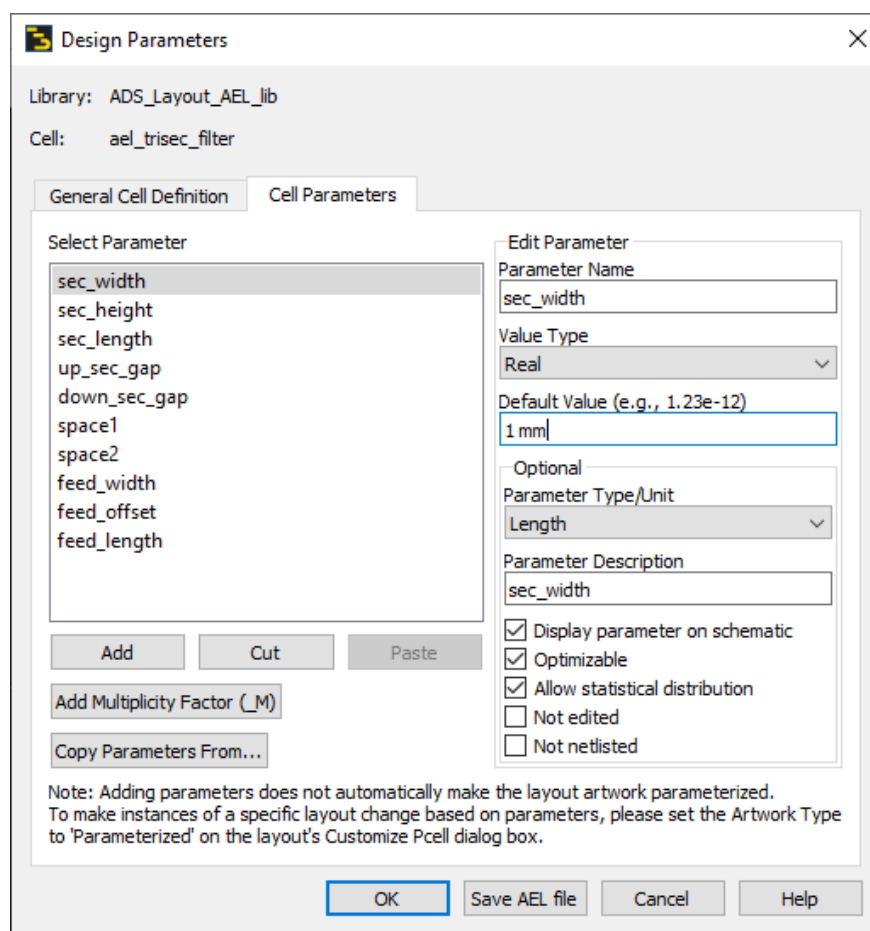
Пусть мы хотим создать автоматическую генерацию топологии трехзвенного фильтра следующего вида:



Все три звена имеют одинаковые внешние размеры (`sec_length` и `sec_height`) и ширину (`sec_width`), но верхнее отличается зазором в кольце (`up_sec_gap` и `down_sec_gap`). Нижние звенья расположены на расстоянии `space2` друг от друга, верхнее отодвинуто на `space1` от нижних. Линии запитки имеют ширину `feed_width` и длину `feed_length` и сдвинуты относительно оси нижних звеньев на `feed_offset` (может быть отрицательным, если надо сдвигать вниз).

Порядок создания топологической ячейки, задаваемой AEL-функцией

1. Создать ячейку-шаблон с пустой топологией. Назовем ее `ael_trisec_filter`. Для нее по File - Design parameters создать все внешние параметры со значениями по умолчанию. Порядок созданных параметров имеет значение, т.к. он должен совпадать с порядком входных переменных для AEL-функции.

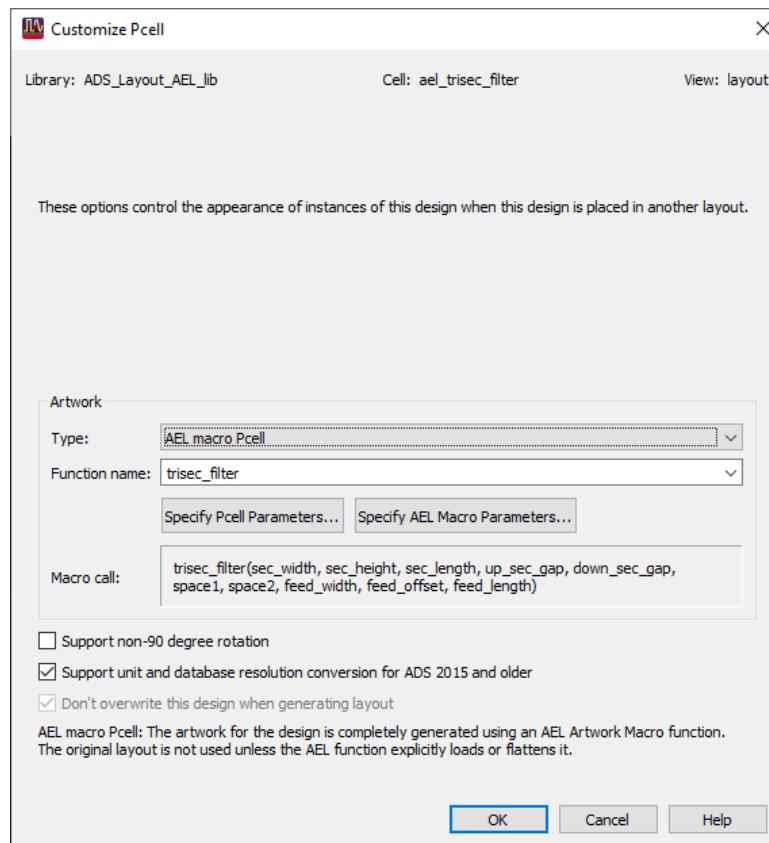


В качестве параметров со значениями по умолчанию возьмем

Таблица 1. Параметры генерируемой топологии

Параметр	Описание	Значение по умолчанию
sec_width	Ширина полоска секции	1 мм
sec_height	Высота секции (по центру полоска)	10 мм
sec_length	Длина секции (по центру полоска)	20 мм
up_sec_gap	Зазор в верхней секции	2 мм
down_sec_gap	Зазор в нижних секциях	4 мм
space1	Зазор между верхней секцией и нижними	1 мм
space2	Зазор между нижними секциями	2 мм
feed_width	Ширина полоска запитки	0.6 мм
feed_offset	Сдвиг полоска запитки	-4 мм
feed_length	Длина полоска запитки	5 мм

2. По команде File – Customize Pcell выставить тип «AEL Macro» и ввести в поле Function Name имя AEL-функции «trisec_filter». Ее пока нет и в ADS она не загружена, поэтому при сохранении вылетит ошибка. Проигнорируем ее.

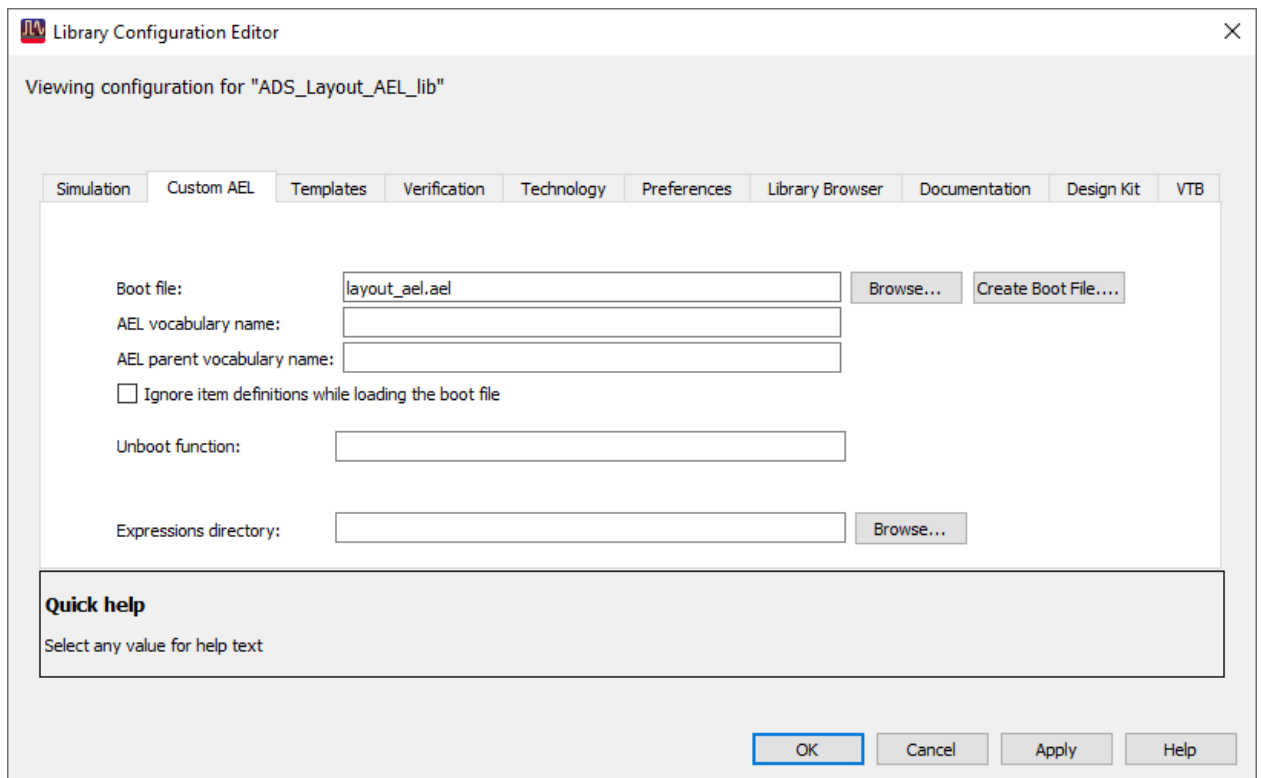


3. Создадим текстовый файл с именем «layout_ael.ael» в папке проекта _wrk в подпапке _lib. В этом файле будут находиться все написанные для текущего проекта AEL-функции.

4. Перед использованием AEL-функции должны быть скомпилированы и загружены в ADS. Если AEL-функция предполагается использовать во многих проектах, то это можно делать через редактирование глобальных переменных USER_AEL, LOCAL_AEL или AEL_PATH. Если AEL-функция будет использоваться только в текущем проекте (как у нас сейчас), то делается это следующим образом:

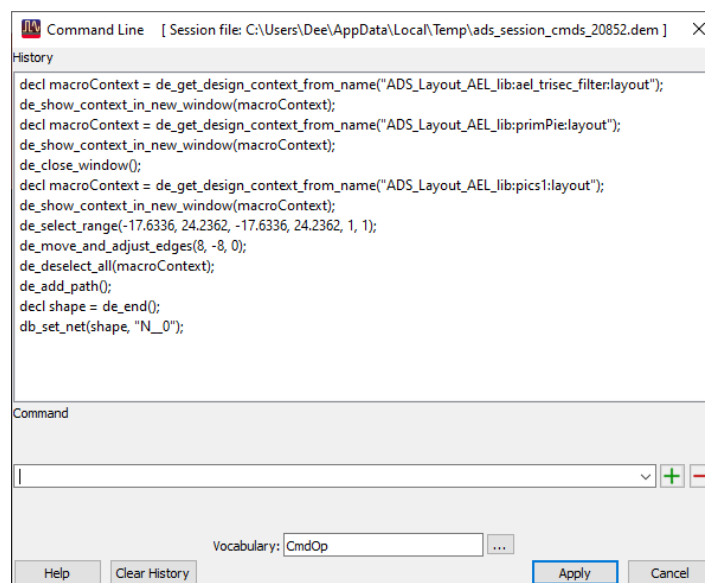
В основном окне ADS на вкладке Library View правой клавишей по библиотеке вызвать Configure Library...

Перейти на вкладку Custom AEL и в поле Boot file: выбрать созданный «layout_ael.ael». При выборе согласиться с перезаписью.



Теперь при перезагрузке проекта будут перекомпилироваться и перезагружаться все AEL-функции, внесенные в файл «layout_ael.ael». Для быстрой перезагрузки проекта будем пользоваться меню в основном окне ADS File – Recent Worspaces...

Также можно перезагружать и перекомпилировать AEL-функции через консоль. Вызвать ее можно в основном окне ADS команда Tools – Command Line...



В консоли пишутся все выполненные ADS операции (можно использовать, чтобы узнать, как выглядит нужная команда), также можно задавать свои.

Команда для загрузки файла `load("myfunc.ael")`.

5. Теперь можно редактировать «layout_ael.ael». Делать это можно любым текстовым редактором типа Блокнот (Notepad).

Пойдем по содержанию файла:

В первую очередь в ael-файле объявляется функция, которая будет использоваться для генерации топологии (мы указали «trisee_filter»). Порядок аргументов должен совпадать с порядком параметров, объявленных в ячейке.

```
defun trisee_filter (sec_width, sec_height, sec_length, up_sec_gap,  
down_sec_gap, space1, space2, feed_width, feed_offset, feed_length)
```

```
{
```

Загрузка контекста и указание слоя металлизации

```
decl designContext = de_get_current_design_context();
```

```
decl cond_layerID = db_get_layerid(designContext, "cond", "drawing");
```

Проверка параметров на корректность. Нужно проверить:

- все параметры, кроме feed_offset, должны быть больше 0
- вырезы в секциях не больше, чем длина секции с ее учетом ширины
- запитка не вылезает за границу длины секции

```
//parameter checks
```

```
if (sec_width <= 0)
```

```
    warning(stderr, 1, fmt_tokens(list(errList, "sec_width <= 0")), "Parameters  
Error");
```

```
if (sec_height <= 0)
```

```
    warning(stderr, 2, fmt_tokens(list(errList, "sec_height <= 0")), "Parameters  
Error");
```

```
if (sec_length <= 0)
```

```
    warning(stderr, 3, fmt_tokens(list(errList, "sec_length <= 0")), "Parameters  
Error");
```

```
if (up_sec_gap <= 0)
```



```

        warning(stderr, 4, fmt_tokens(list(errList, "up_sec_gap <= 0")),
"Parameters Error");
if (down_sec_gap <= 0)
    warning(stderr, 5, fmt_tokens(list(errList, "down_sec_gap <= 0")),
"Parameters Error");
if (space1 <= 0)
    warning(stderr, 6, fmt_tokens(list(errList, "space1 <= 0")), "Parameters
Error");
if (space2 <= 0)
    warning(stderr, 7, fmt_tokens(list(errList, "space2 <= 0")), "Parameters
Error");
if (feed_width <= 0)
    warning(stderr, 8, fmt_tokens(list(errList, "feed_width <= 0")), "Parameters
Error");
if (feed_length <= 0)
    warning(stderr, 9, fmt_tokens(list(errList, "feed_length <= 0")),
"Parameters Error");
if (up_sec_gap >= sec_length - seq_width)
    warning(stderr, 10, fmt_tokens(list(errList, "up_sec_gap >= sec_length -
seq_width")), "Parameters Error");
if (down_sec_gap >= sec_length - seq_width)
    warning(stderr, 11, fmt_tokens(list(errList, "down_sec_gap >= sec_length
- seq_width")), "Parameters Error");
if (abs(feed_offset) + 0.5*feed_width > sec_length + 0.5*sec_width)
    warning(stderr, 12, fmt_tokens(list(errList, " abs(feed_offset) +
0.5*feed_width > sec_length + 0.5*sec_width ")), "Parameters Error");

```

Пересчет значений переменных во внутренние единицы измерения (чтобы не зависеть от технологических параметров проекта). Делается с помощью множителя `mks2uu`, определяемого из контекста проекта

```

// unit conversion
decl mks2uu = db_get_mks_to_uu_factor(designContext);
sec_width = sec_width * mks2uu;
sec_height = sec_height * mks2uu;
sec_length = sec_length * mks2uu;

```

```

up_sec_gap = up_sec_gap * mks2uu;
down_sec_gap = down_sec_gap * mks2uu;
space1 = space1 * mks2uu;
space2 = space2 * mks2uu;
feed_width = feed_width * mks2uu;
feed_offset = feed_offset * mks2uu;
feed_length = feed_length * mks2uu;

```

Секции будем рисовать через Path. Начнем с левой. Зададим ей ширину sec_width и квадратный угол.

```

// path parameters
db_set_path_width(designContext, sec_width);
db_set_path_corner(designContext, DB_SQUARE_CORNER);

```

Рисуем Path по точкам, но без смещения влево.

```

// create left rect
de_add_path();
de_add_point(0, down_sec_gap/2);
de_add_point(0, sec_length/2);
de_add_point(-sec_height, sec_length/2);
de_add_point(-sec_height, -sec_length/2);
de_add_point(0, -sec_length/2);
de_add_point(0, -down_sec_gap/2);
de_end();

```

Пририсовываем к левой секции линию запитки с шириной feed_width.

```

// create left feedline
db_set_path_width(designContext, feed_width);
de_add_path();
de_add_point(-sec_height-sec_width/2, feed_offset);
de_add_point(-sec_height-sec_width/2-feed_length, feed_offset);
de_end();

```

Объединим созданное и присвоим переменной leftRect

```

// merge left rect with feed line
de_select_all();
decl leftRect = de_union();

```

Сдвинем левую секцию на e место, создадим копию и отзеркалим ее в правую часть. Также снимем выделение.

```
// move left section to its place and create right section (copy and mirror)
de_move(-(space2+sec_width)/2, 0);
decl rightRect = de_copy(0, 0);
de_mirror_y(0, 0);
de_deselect_all();
```

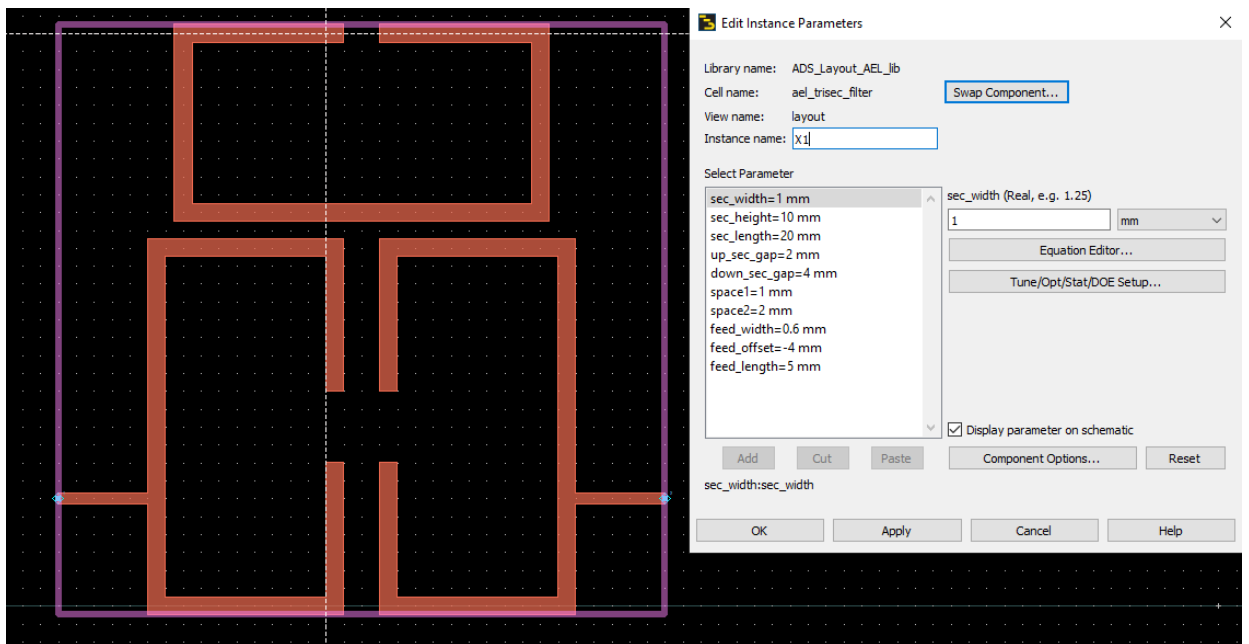
Теперь создадим верхнюю часть, затем сдвинем ее вверх на нужное место.

```
// create upper section in center and move
db_set_path_width(designContext, sec_width);
de_add_path();
de_add_point(up_sec_gap/2, sec_height);
de_add_point(sec_length/2, sec_height);
de_add_point(sec_length/2, 0);
de_add_point(-sec_length/2, 0);
de_add_point(-sec_length/2, sec_height);
de_add_point(-up_sec_gap/2, sec_height);
decl upperRect = de_end();
db_select(upperRect);
de_move(0, sec_width + sec_length/2 + space1);
```

В конце создадим пины в точках соединения.

```
// create pins
db_create_pin (designContext, -(feed_length + sec_width + sec_height +
space2/2), feed_offset, 180, cond_layerID, 1);
db_create_pin (designContext, (feed_length + sec_width + sec_height +
space2/2), feed_offset, 0, cond_layerID, 2);
}
```

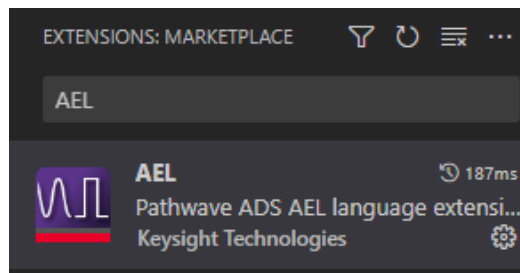
AEL-функция написана. Теперь, если его перезагрузить проект, создать топологию верхнего уровня «trisee_filter_testbench» и в нее внести ячейку «ael_trisee_filter», то сгенерируется топологический компонент.



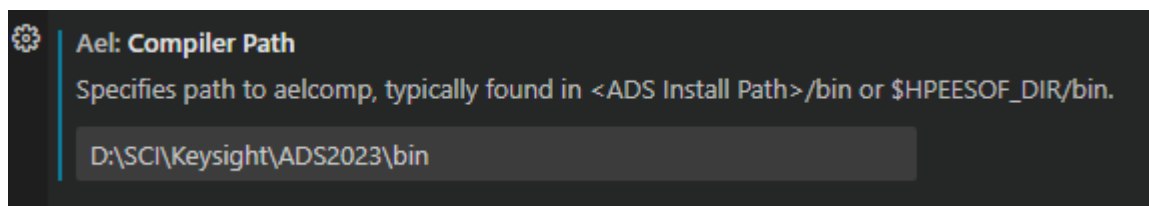
Использование редактора VSCode

Начиная с версии ADS2022 возможно использовать популярный бесплатный редактор кода VSCode [7]. Чтобы VSCode умел работать с ael-файлами нужно сделать несколько настроек:

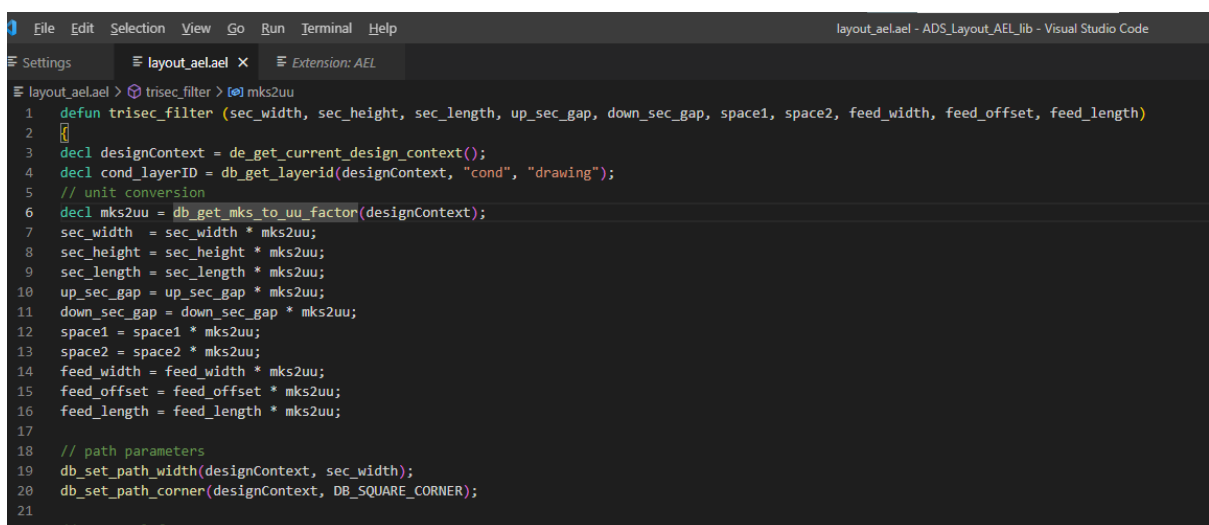
1. Установить из маркетплейса VSCode расширение AEL (Pathwave ADS AEL language extension...).



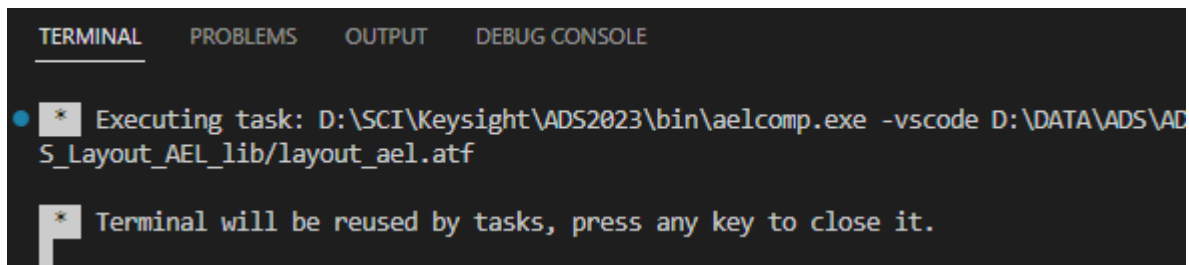
2. В настройках расширения AEL прописать путь к папке bin в папке с установленным ADS (<ADS Install Path>/bin или \$HPEESOF_DIR/bin), где находится компилятор ael-языка.



После установки расширения, при редактировании ael-файлов в VSCode появится подсветка кода.



Запуск компилятора написанного кода делается по Terminal – Run Build Task (Ctrl+Shift+B).



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

* Executing task: D:\SCI\Keysight\ADS2023\bin\aelcomp.exe -vscode D:\DATA\ADS\AD
S_Layout_AEL_lib/layout_ael.atf

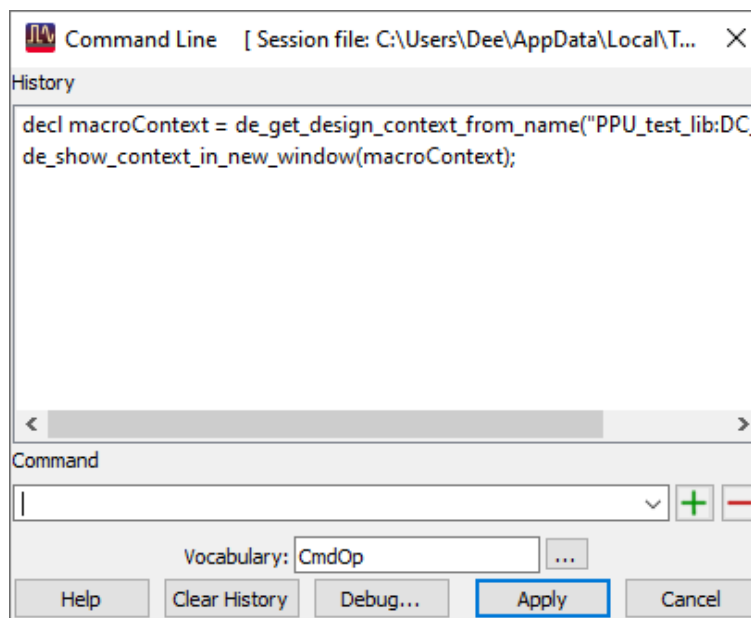
* Terminal will be reused by tasks, press any key to close it.
```

Перед загрузкой обновленного кода в ADS нужно заранее скомпилировать ael-файл в atf-файл, автоматически проверка актуальности atf-файла не ведется.

При разработке доступен дебаггер в VSCode. Возможно два режима подключения:

1. Автоматический вызов дебаггера VSCode при выполнении команд в ADS. Для этого в ael-коде должен присутствовать вызов функции `attach_vscode()`; Как выполнение дойдет до этой строчки, автоматически запускается дебаггер.

2. Также можно принудительно вызвать дебаггер из консоли ADS по кнопке Debug.



Задание на выполнение

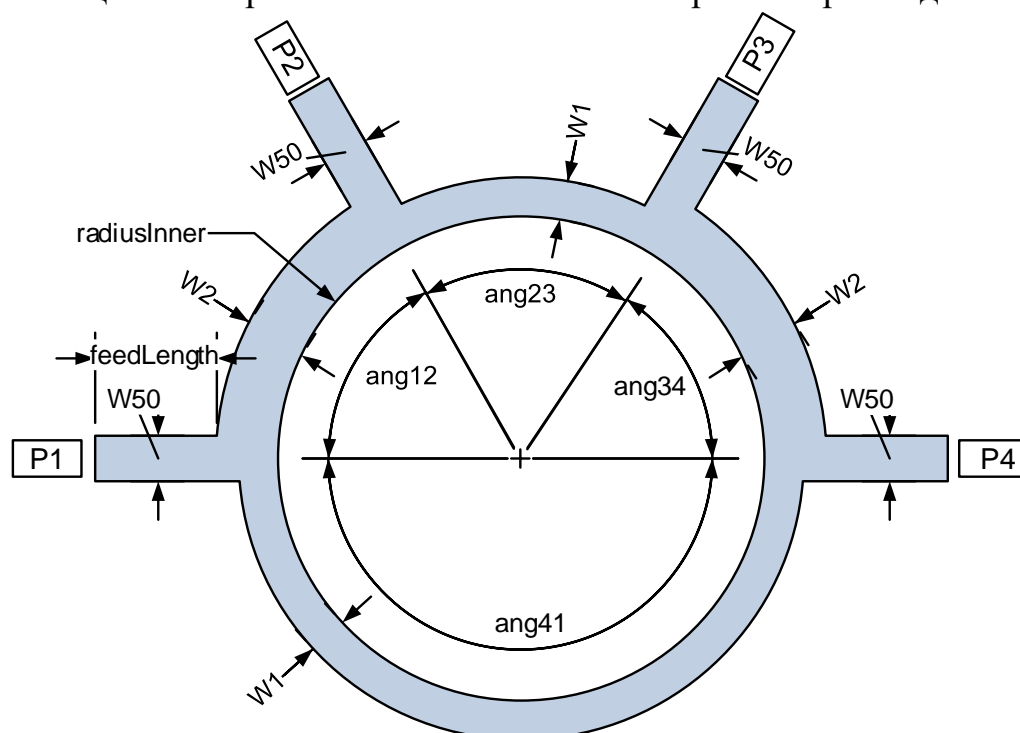
В соответствии с вариантом написать код автоматического создания топологии с применением языка AEL. Варианты заданий приведены в таблицах ниже.

При выполнении ориентироваться на методику выполнения.

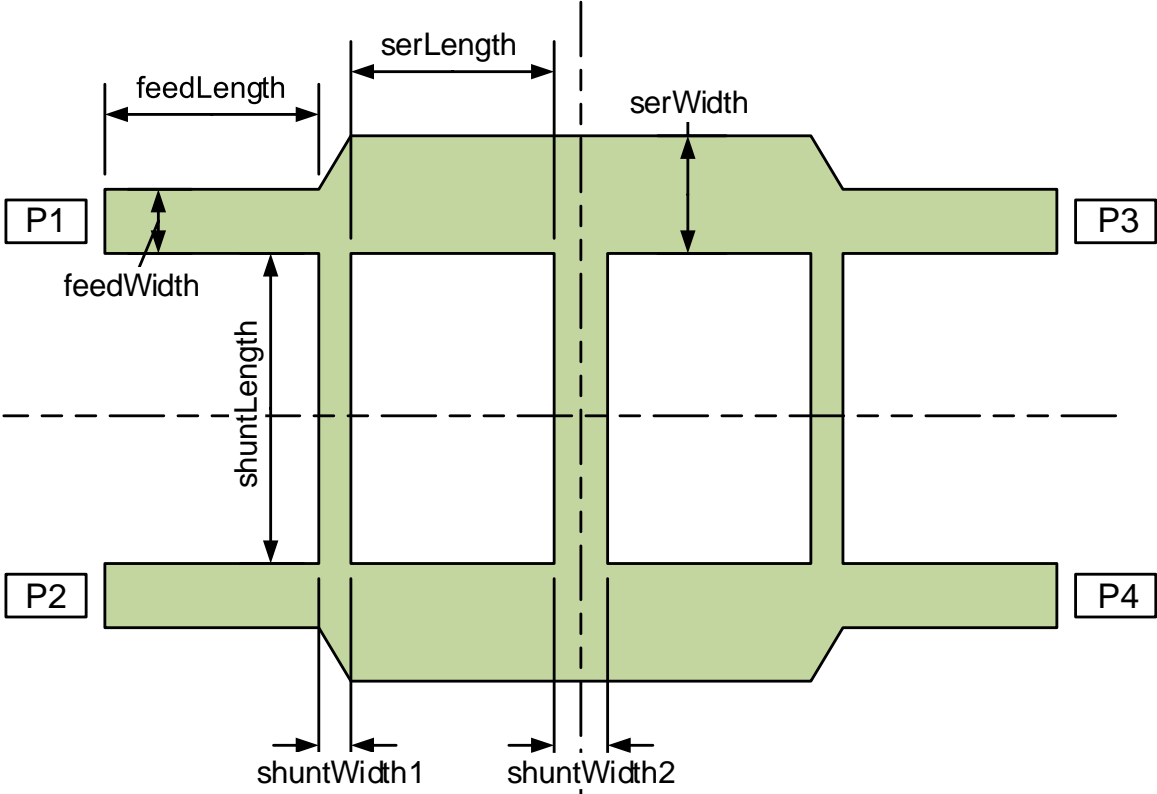
Таблица 2. Вариант задания 1

Вариант 1.

Кольцевой направленный ответвитель с неравномерным делением.

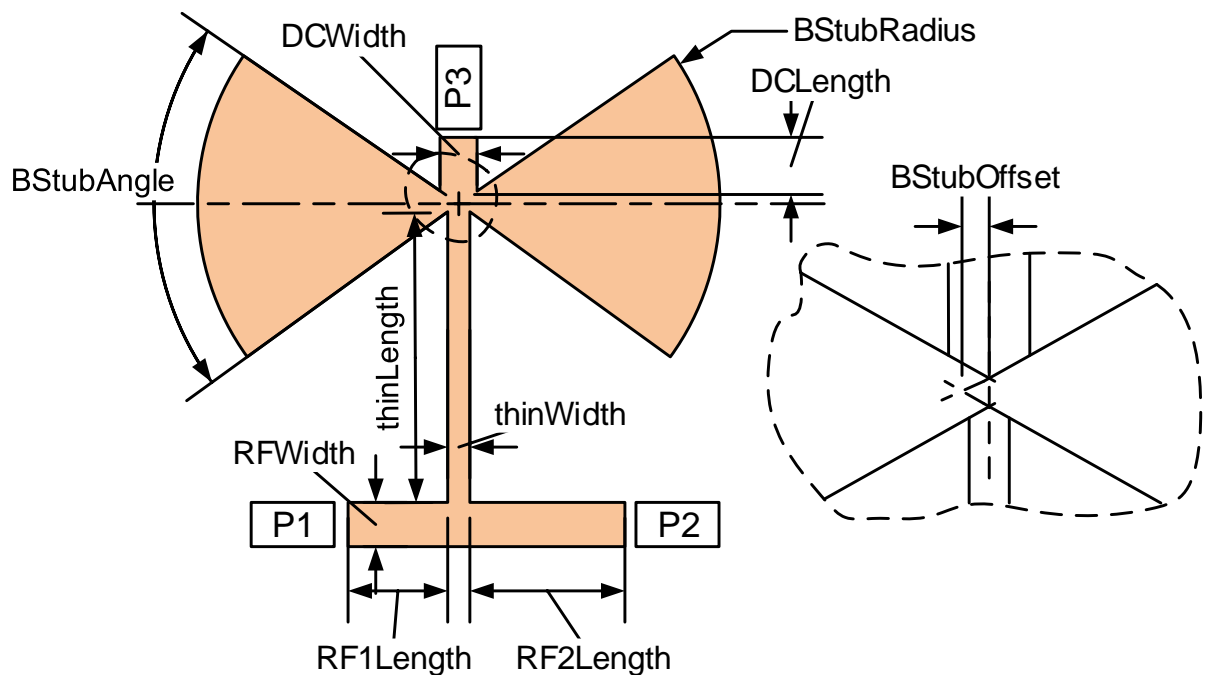


Имя	Пояснение	Тип	Значение
feedLength	Длина запитки	Длина	2,5 мм
W50	Ширина запитки	Длина	1,1 мм
W1	Ширина участков 23 и 41	Длина	0,4 мм
W2	Ширина участков 12 и 34	Длина	0,8 мм
radiusInner	Радиус внутреннего кольца	Длина	7,5 мм
ang12	Угол между пинами 1 и 2	Угол	60°
ang23	Угол между пинами 2 и 3	Угол	60°
ang34	Угол между пинами 3 и 4	Угол	60°
ang41	Угол между пинами 1 и 4	Угол	180°
Дополнительная проверка: $ang12 + ang23 + ang34 + ang41 = 360^\circ$			

<p>Вариант 2.</p> <p>Двухкаскадный двушлейфный направленный ответвитель с неравномерным делением.</p> 			
Имя	Пояснение	Тип	Значение
feedLength	Длина запитки	Длина	2,5 мм
feedWidth	Ширина запитки	Длина	1,2 мм
shuntLength	Длина параллельных шлейфов	Длина	7 мм
shuntWidth1	Ширина внешних параллельных шлейфов	Длина	0,3 мм
shuntWidth2	Ширина внутреннего параллельного шлейфа	Длина	0,8 мм
serLength	Длина последовательных шлейфов	Длина	6,8 мм
serWidth	Ширина последовательных шлейфов	Длина	1,9 мм
<p>Дополнительные проверки:</p> <p>$serWidth > feedWidth$</p> <p>$feedWidth > shuntWidth2$</p> <p>$shuntWidth2 > shuntWidth1$</p>			

Вариант 3.

Цепь подачи питания с широкополосным шлейфом типа бабочка.



Имя	Пояснение	Тип	Значение
RFWidth	Ширина ВЧ-линий	Длина	0,45 мм
RF1Length	Длина ВЧ-линии со стороны входа 1	Длина	1,1 мм
RF2Length	Длина ВЧ-линии со стороны входа 2	Длина	2,5 мм
thinWidth	Ширина тонкого шлейфа	Длина	0,15 мм
thinLength	Длина тонкого шлейфа	Длина	6,5 мм
DCWidth	Ширина линии со стороны входа 3	Длина	0,5 мм
DCLength	Длина линии со стороны входа 3	Длина	2,5 мм
BStubAngle	Угловой размах широкополосного шлейфа	Угол	120°
BStubRadius	Радиус широкополосного шлейфа	Длина	4 мм
BStubOffset	Смещение центра сектора широкополосного шлейфа относительно оси. < 0 – смещение внутрь (сектора накладываются друг на друга) > 0 – смещение наружу (сектора разъезжаются относительно друг на друга)	Длина	-0,4 мм

Дополнительная проверка:

Нижние углы секторов широкополосного шлейфа не касаются ВЧ-линий

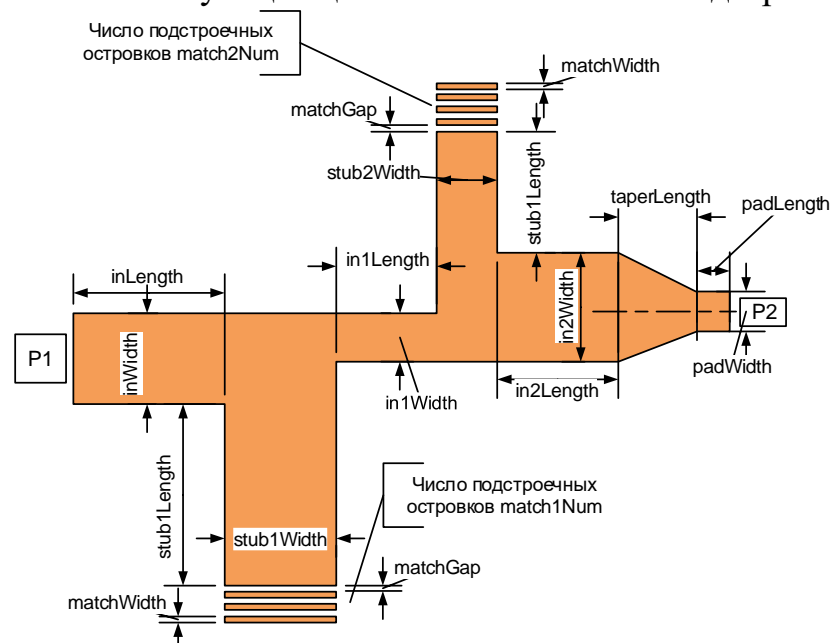
Вариант 4.
Кольцевой развязанный делитель мощности.

Имя	Пояснение	Тип	Значение
feedLegth	Длина подводящих линий	Длина	2,5 мм
feedWidth	Ширина подводящих линий	Длина	0,8 мм
ArcRadius	Радиус (по центру) дуг	Длина	7 мм
ArcWidth	Ширина дуг	Длина	0,5 мм
ResPadWidth	Ширина падов резистора	Длина	0,7 мм
ResPadSpace	Зазор между падами резистора	Длина	0,8 мм
ResPasLength	Длина падов резистора	Длина	0,4
ResNom	Номинал резистора	Сопротивление	100 Ом

Установить компонент резистора с падами «R_Pad1»
Дополнительные проверки:
Нужно убедиться, что души и подводящие линии 2 и 3 пересекаются в пределах падов резистора

Вариант 5.

Микрополосковая согласующая цепь с возможностью подстройки.



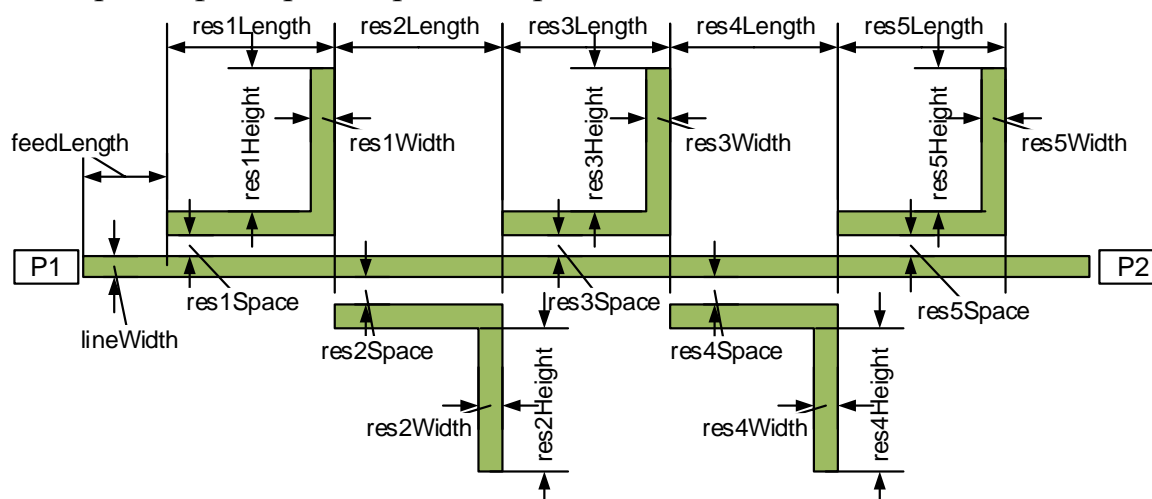
Имя	Пояснение	Тип	Значение
inLength	Длина входной ВЧ-линии	Длина	2,5 мм
inWidth	Ширина входной ВЧ-линии	Длина	1,1 мм
stab1Length	Длина первого шлейфа	Длина	6 мм
stab1Width	Ширина первого шлейфа	Длина	2,0 мм
in1Length	Расстояние между шлейфами	Длина	3 мм
in1Width	Ширина участка между шлейфами	Длина	0,8 мм
stab2Length	Длина второго шлейфа	Длина	4 мм
stab2Width	Ширина второго шлейфа	Длина	0,6 мм
in2Length	Длина участка от второго шлейфа	Длина	1,0 мм
in2Width	Ширина участка от второго шлейфа	Длина	0,9 мм
taperLength	Длина перехода	Длина	0,3 мм
padLength	Длина пада	Длина	0,4 мм
padWidth	Ширина пада	Длина	0,25 мм
matchThick	Длина построечных участков	Длина	0,2 мм
matchGap	Зазор между подстроечными участками	Длина	0,15 мм
match1Count	Число подстроечных участков в первом шлейфе	Безразмерное	3
match1Count	Число подстроечных участков во втором шлейфе	Безразмерное	4

Таблица 7. Вариант задания 6

<div>Вариант 6. Балун в микрополосковом исполнении.</div> <div></div>			
Имя	Пояснение	Тип	Значение
balFeedWidth	Ширина входа симметричной линии	Длина	0,4 мм
balFeedLength	Длина входа симметричной линии	Длина	2,5 мм
balFeedGap	Зазор симметричной линии	Длина	0,3 мм
bal1Length	Длина связанного участка со стороны первой симметричной линии	Длина	4 мм
bal2Length	Длина связанного участка со стороны второй симметричной линии	Длина	4,2 мм
coupWidth	Ширина линий связанного участка	Длина	0,7 мм
coupGap	Зазор между линиями на связанном участке	Длина	0,4 мм
unbalLength	Длина связанного участка несимметричной линии	Длина	8,3 мм
unbalFeedWidth	Ширина входа несимметричной линии	Длина	0,7 мм
unbalFeedLength	Длина входа несимметричной линии	Длина	0,7 мм
Miter	Срез на несимметричной линии	Длина	0,2 мм
viaDiameter	Диаметр отверстия	Длина	0,3 мм
viaOffset	Отступ отверстия от конца симметричных линий	Длина	0,35 мм
<div>Дополнительные проверки:</div> <div><ul style="list-style-type: none">- Miter срезает не весь угол полностью- Отверстия не выходят за границу линий</div> <div>Отверстия размещать окружностями в слое «hole».</div>			

Таблица 8. Вариант задания 7

Вариант 7.
Режекторный фильтр на L-резонаторах.



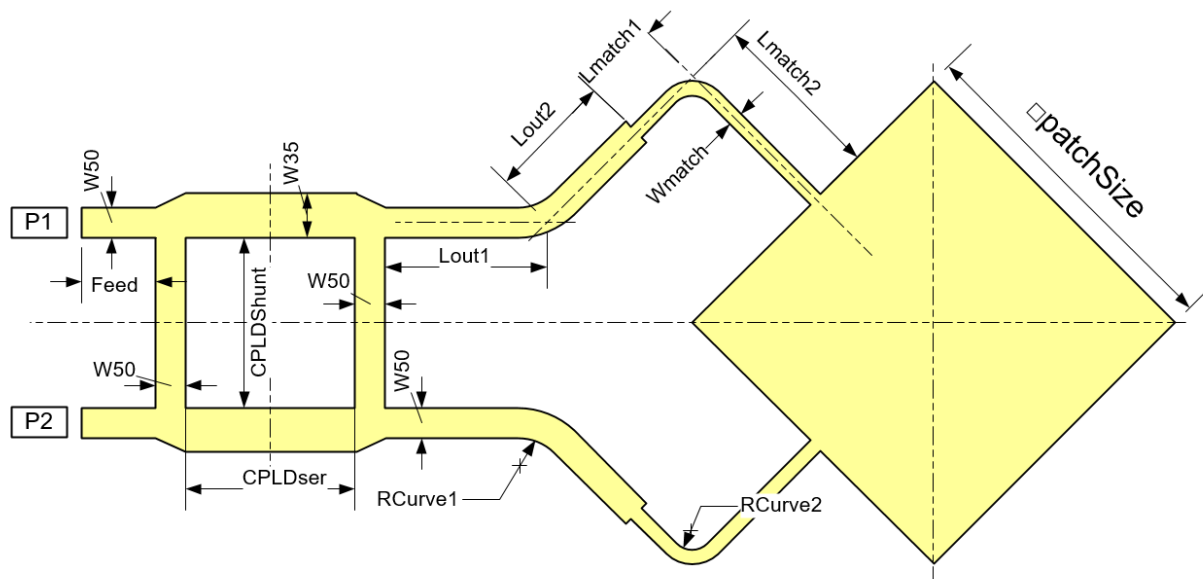
Имя	Пояснение	Тип	Значение
lineWidth	Ширина основной линии	Длина	1,1 мм
res1Length	Длина первого резонатора	Длина	7,9 мм
res1Height	Высота первого резонатора	Длина	8,3 мм
res1Width	Ширина первого резонатора	Длина	1,0 мм
res1Space	Отступ первого резонатора	Длина	0,3 мм
res2Length	Длина второго резонатора	Длина	7,7 мм
res2Height	Высота второго резонатора	Длина	8,2 мм
res2Width	Ширина второго резонатора	Длина	0,9 мм
res2Space	Отступ второго резонатора	Длина	0,2 мм
res3Length	Длина третьего резонатора	Длина	7,5 мм
res3Height	Высота третьего резонатора	Длина	8,0 мм
res3Width	Ширина третьего резонатора	Длина	0,7 мм
res3Space	Отступ третьего резонатора	Длина	0,1 мм
res4Length	Длина четвертого резонатора	Длина	7,7 мм
res4Height	Высота четвертого резонатора	Длина	8,2 мм
res4Width	Ширина четвертого резонатора	Длина	0,9 мм
res4Space	Отступ четвертого резонатора	Длина	0,2 мм
res5Length	Длина пятого резонатора	Длина	7,9 мм
res5Height	Высота пятого резонатора	Длина	8,3 мм
res5Width	Ширина пятого резонатора	Длина	1,0 мм
res5Space	Отступ пятого резонатора	Длина	0,3 мм
feedLength	Длина линий запитки	Длина	2,5 мм

<p>Вариант 8.</p> <p>Двухкаскадный направленный ответвитель на кольцевых резонаторах.</p> <p>feedLength</p> <p>resGapBetween</p> <p>lineWidth</p> <p>P1</p> <p>P2</p> <p>resSpace</p> <p>resLength</p> <p>resHeight</p> <p>resArcRad</p> <p>P3</p> <p>P4</p> <p>resWidth</p>			
Имя	Пояснение	Тип	Значение
lineWidth	Ширина основной линии	Длина	0,8 мм
resLength	Длина резонатора	Длина	4,5 мм
resHeight	Высота резонатора	Длина	3,6 мм
resWidth	Ширина линии резонатора	Длина	0,4 мм
resGapBetween	Зазор между резонаторами	Длина	0,2 мм
resSpace	Зазор между основной линией и резонатором	Длина	0,15 мм
resArcRad	Радиус сглаживания кольца резонатора	Длина	1,0 мм
feedLength	Длина линии запитки	Длина	1,5 мм
<p>Дополнительная проверка:</p> <p>Кольцо резонатора корректно формируется, т.е.</p> <p>$resArcRad \leq \frac{1}{2} * \min(resLength, resHeight)$</p>			

Таблица 10. Вариант задания 9

Вариант 9.

Металлизация патча с круговой поляризацией с двойной запиткой с формированием сдвига фаз с помощью двушлейфного направленного ответвителя.



Имя	Пояснение	Тип	Значение
patchSize	Размер патча	Длина	12 мм
W50	Ширина 50 Ом	Длина	1,1 мм
W35	Ширина 35 Ом	Длина	1,9 мм
Wmatch	Ширина согласующих участков	Длина	0,35 мм
CPLDser	Длина последовательной секции НО	Длина	7,9 мм
CPLDshunt	Длина параллельной секции НО	Длина	8,0 мм
Lout1	Длина первой промежуточной части	Длина	5 мм
Lmatch1	Длина первой части согласующей части	Длина	2,5 мм
Lmatch2	Длина второй части согласующей части	Длина	5,5 мм
RCurve1	Радиус поворота промежуточной части	Длина	1,3 мм
RCurve2	Радиус поворота согласующей части	Длина	0,9 мм
Feed	Длина линии запитки	Длина	3,0 мм

Дополнительные проверки:

Длина второй промежуточной части Lout2 высчитывается из остальной геометрии, должна быть $< 2 * RCurve1$

Ввод согласующего участка в центр стороны патча

Переход с промежуточной части в согласующую соосен

Этапы выполнения:

1. В соответствии с методикой подготовить ячейку-шаблон с пустой топологией (именование ячейки `ael_ИмяТопологии`):

- задать параметры ячейки в соответствии с заданием;
- выставить тип ячейки - параметризованный AEL Macro и задать имя AEL-функции;

2. Создать файл с ael-функциями, разместить его в пределах библиотеки проекта (подпапка `_lib`). Именование файла «myLayout.ael».

3. Привязать к библиотеке созданный ael-файл.

4. Создать функцию генерации топологии на основе языка AEL. В функции должны быть реализованы:

- проверки на реализуемость топологии (корректность параметров)
- перенормировка длин из mks в um
- создание топологии
- создание пинов

При написании AEL-функции не обязательно сразу описывать все создание топологии, можно писать код постепенно и с помощью операций пп.5 и 6 проверять, что получается.

5. Подготовить ячейку верхнего уровня с топологией (именование ячейки `ИмяТопологии_Top`), в нее внести как подсхему ячейку-шаблон.

6. В ячейке верхнего уровня, изменяя параметры компонента топологии, убедиться в корректности ее генерации или выводе ошибок, включая явно некорректные значения параметров.

Требования к отчёту

Отчет о выполненной лабораторной работе должен представлять из себя структурированное описание всей выполненной работы, в том числе должен содержать:

- Титульный лист, с наименованием работы, указанием исполнителя и даты выполнения.

- Цель (-и) работы.
- Список использованных инструментов в лабораторной работе.
- Разделы работы, которые рекомендовано формировать в соответствии с этапами выполнения (аналогично методике и заданию на выполнение).
- Для каждого этапа необходимо привести все выполненные операции, которые могут включать в себя расчеты, подготовленные модели и анализ результатов моделирования.

Подробность описания этапов выполненной работе студент определяет самостоятельно. Самое простое правило, которому надо следовать – человек, незнакомый с выполненной работой (но знакомый с использованным инструментом) должен быть в состоянии по отчету повторить данную работу и понять, что в ней происходит.

Готовый отчет необходимо экспортировать в pdf.

Архив проекта ADS лучше всего делать встроенным инструментом File – Archive Workspace из основного окна ADS. При архивации проекта можно выбрать, какие составляющие проекта добавлять в архив. Если какие-то ячейки или результаты расчета не нужны, то их можно исключить из архивирования.

Созданный архив будет иметь расширение *.7zads и является фактическим 7z-архивом.

По окончании выполнения лабораторной работы и подготовки отчета, отчет и архив проекта надо выложить в ОРИОКС в домашнее задание в дисциплину, привязав к контрольному мероприятию ЗЛР (Защита лабораторных работ). Именование отчета и архива проекта должно давать возможность точно понять, к какой теме лабораторной работы они относятся (Например, AFU_Lab_BCoupler вместо непонятного Lab1 или Workspace1).

Задание на самостоятельную работу

- 1) Подготовка к лабораторному занятию

При подготовке к выполнению лабораторной работы необходимо продумать шаблон отчета, при необходимости внести краткие теоретические сведения, продумать и наметить количество, вид и расположение таблиц и графиков с измеренными данными. Для получения допуска необходимо подробно изучить теоретический материал.

2) С использованием навыков, полученных в лабораторной работе, выполнить соответствующий этап БДЗ.

Контрольные вопросы

1. Объясните, зачем можно использовать AEL-функции для генерации параметризованных топологий? Почему не обойтись существующими полосковыми компонентами?
2. Поясните, как нужно подготовить ячейку для привязки к ней AEL-функции для генерации топологии.
3. Что означают префиксы «db_» и «de_» в ael-функциях?
4. Что такое контекст выполнения функции?
5. Найдите в справке команды выделения объектов по заданным координатам. Как нужно корректно настроить фильтры перед их применением?
6. Зачем нужно проводить проверку параметров топологии перед ее генерацией?
7. Поясните отличия единиц mks (система СИ), uu (пользовательские единицы технологии) и dbu (единицы разрешения технологии).
8. Поясните порядок создания полигонов.
9. Что такое примитив Path? Чем он отличается от Trace и полигона?

Литература

1. Банков, С. Е. Электродинамика для пользователей САПР СВЧ : учебник / С. Е. Банков, А. А. Курушин. — Москва : СОЛОН-Пресс, 2017. — 316 с. — ISBN 978-5-91359-236-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/107661> (дата обращения: 02.02.2021). — Режим доступа: для авториз. пользователей.

Перечень ресурсов сети «Интернет»

2. Сборник примеров работы в ADS «ADS Example Book: Focused on RF and Microwave Design», доступен после свободной регистрации <https://www.keysight.com/main/editorial.jsp?cc=RU&lc=rus&ckey=2704333&id=2704333&cmpid=zzfindeesof-ads-rfmw-examples>

3. База знаний Образовательного центра Keysight EEsof EDA Knowledge Center, доступен после свободной регистрации, <http://edadocs.software.keysight.com/display/support/Knowledge+Center>

4. Тематический раздел «Rf & Microwave Design» форума electronix.ru, доступен после свободной регистрации, <https://electronix.ru/forum/index.php?showforum=63>

5. Интернет-энциклопедия разработчиков СВЧ-аппаратуры «Microwaves101» <https://www.microwaves101.com>

6. Бесплатный текстовый редактор Notepad++ <https://notepad-plus-plus.org/>

7. Редактор кода Visual Studio Code <https://code.visualstudio.com/>

Каналы Youtube с видеоуроками по Keysight Advanced Design System

8. Канал youtube образовательного центра Keysight EEsof EDA <https://www.youtube.com/user/KeysightEESOF>

9. Канал youtube Anurag Bhargava образовательного центра <https://www.youtube.com/user/BhargavaAnurag>

10. Канал youtube Keysight EEsof EDA Field <https://www.youtube.com/c/EEsofAETips>

Разработчик:

Ст. преподаватель Института МПСУ

Приходько Д.В.