FlashPOM Designer

System specification document

Project MATEO

http://FlashPoM.zcu.cz

Project team

prof.Václav Skala

ing.Jan Kaiser

ing.Vojtěch Hladík

ing.František Novák

Plzeň 17.7.2007

Jan Kaiser

1 Introduction

Purpose of this document is to specify requirements on the software editor for chip design.

Software enables the user to design the chip layer by layer. Each layer has a thickness associated with it and a set of geometrical shapes. If the layer belongs to class 'electrodes' it can be a glass layer or a conductor layer. (Electrodes are actually fabricated by stacking a patterned conductor on a glass substrate). If the layer is a micro fluidics layer, the shapes in it will be trenches, holes, etc. One device could have multiple layers of each type stacked one on top of the other. Software output is file in SVG like format that is ready for chip fabricating process.

Software should do some rule checks and cost estimations and include them in the descriptor files. The rule checks will ensure that the device can actually be fabricated with technology developed within the FlashPOM project and the cost estimation will provide price estimate based on the structures put in the design.

Finally, the GUI should enable the user to get the best possible idea of the 3D appearance of the device he or she has fabricated. Top view and cross-section are a must but the possibility of doing 3D rendering with rotation in space, etc. is welcome.

The software should be easy to use and downloadable off the website.

1.1 Problem Statement

General problems to be solved by the software are the following:

- to provide reliable and easy to use software for chip designing
- to provide output in SVG like format

While there are numerous vector graphic editors that can be used for chip designing they are general and aimed to artists. Software developed within this project is aimed on chip designers. It provides solution with functional improvements over existing tools. On the other hand unnecessary functions for chip design are to be omitted (i.e. graphic objects color, various fill patterns…)

1.2 System Personnel

System personnel involved in the FlashPOM designer are the following:

1.2.1 FlashPOM project leader

Dr. Enric Claverol-Tinturé ectmail@eel.upc.es

Enric is leader of the FlashPOM project.

1.2.2 FlashPOM designer project manager at UWB

Prof. Ing. Václav Skala, Csc. skala@kiv.zcu.cz

Václav is leader of the designer part of FlashPOM project.

1.2.3 FlashPOM designer system analyst

Jan Kaiser kaiserj@kiv.zcu.cz

Jan is responsible for the design and development of FlashPOM designer.

1.2.4 FlashPOM designer system developer

Vojtěch Hladík vojta.hladik@email.cz
František Novák novak_frantisek@centrum.cz

Vojta and František are students at the University of West Bohemia and developers of the FlashPOM designer.

1.2.5 FlashPOM designer end users

Michael Riss mriss@pcb.ub.es

Carlos Fernandez Escuin kescuin@gmail.com

Michael is designer of biological chips. He is the typical end user of developed application.

Carlos is working on the laser device for chip layers fabricating. His device works with output of developed application.

1.3 Operational Setting

FlashPOM designer is intended to design chip layers on common desktop computer. There are no specific operational settings requirements.

1.4 Impact Analysis

The positive potential impact of the FlashPOM designer as a functioning system is to increase user productivity during chip design process.

1.5 Related Systems

Open source and commercial software packages that were used till this time include:

- Macromedia Freehand
- xFig
- CorelDraw
- InkScape

There also exist similar commercial project for PCB designing and manufacturing. On the www.pcbexpress.com is easy to find two software packages. ExpressSCH is a schematic design program while ExpressPCB is used for designing circuit board layout. Both programs are small, very simple and intuitive. User can check the price of designed circuit board via cost function that is updated from the internet and finally order the PCB.

2 Functional requirements

2.1 User Interface Overview

Screen of typical working environment is shown in Figure 2-1. Window of the application consists of menu bar, tool bars, and tool windows, tabbed windows for chip design and status bar. All these parts can be moved or turned off to meet user requirements on their placement. Most of the commands are accessible from menu, toolbars and are also reachable using keyboard shortcuts.

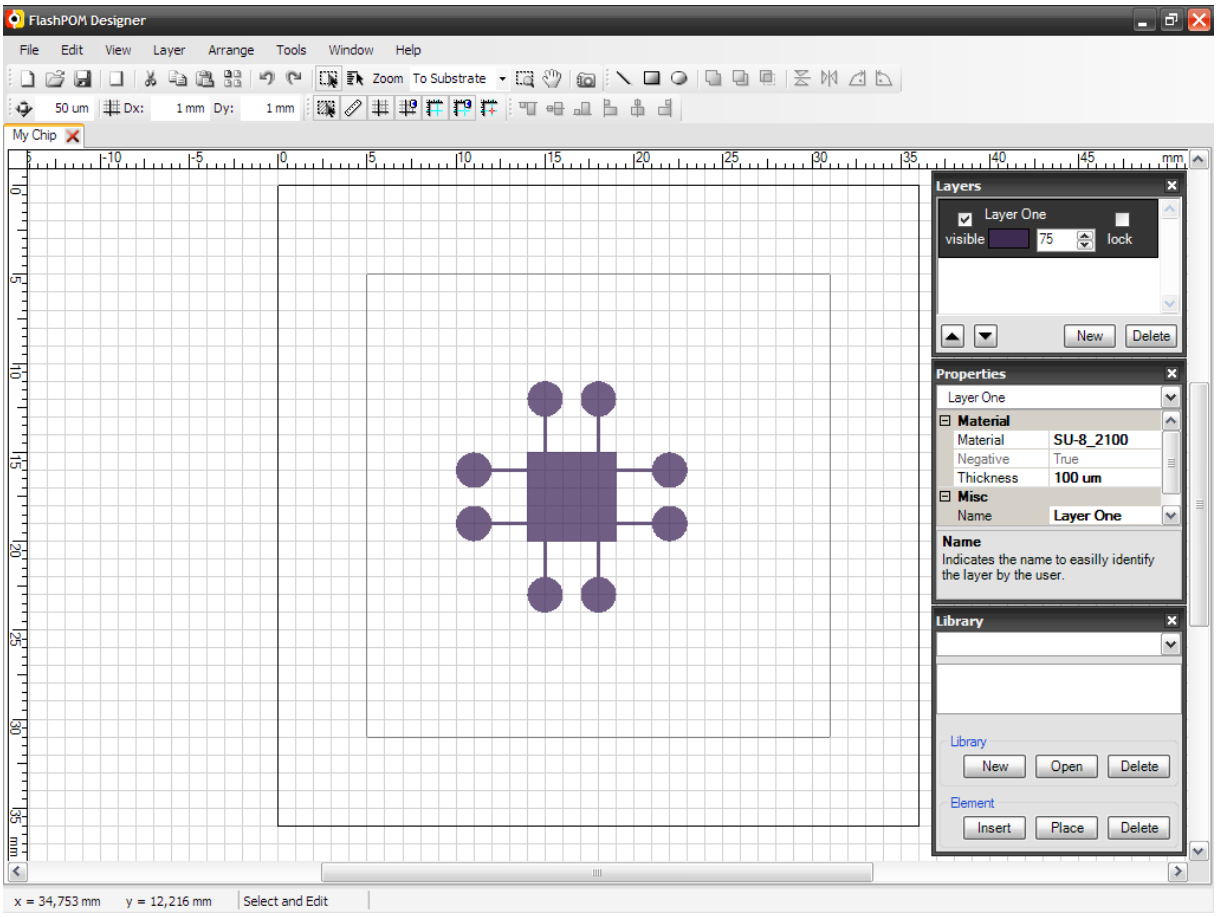

Figure 2-1: Application screen

2.1.1 Menu bar

The Menu bar is a text menu with shortcuts to most functions of the applications. Command shortcuts are placed to appropriate groups.

Figure 2-2: Menu bar

2.1.1.1 File

The File group contains typical commands for manipulating data files and performing other system-level functions.

2.1.1.1.1 New

File->New creates a new empty chip design. Wizard that will guide user thorough entering necessary chip properties will pop-up. These include selection from available substrate materials and available layer materials. Lists of available materials will be stored in configuration file that can be regularly updated over the internet.

2.1.1.1.2 Open

File->Open opens an existing chip design from a previously saved file.

2.1.1.1.3 Save

File->Save saves the currently active chip design on the file from which it was opened or on a new file if it was created from New. Precise description of the proprietary file format is to be developed and described.

2.1.1.1.4 Save As

File->Save As allows the current chip design to be saved on different file from which it was opened or most recently saved upon. Precise description of the proprietary file format is to be developed and described.

2.1.1.1.5 Close

File->Close will close active chip design project.

2.1.1.1.6 Import from FPSVG

File->Import from FPSVG imports chip design from the SVG like file.

2.1.1.1.7 Export to FPSVG

File->Export to FPSVG exports chip design to SVG like file which is to be loaded to laser device for chip fabricating. For FSVG file format specification see 2.2.1.

2.1.1.1.8 Recent Files

File->Recent Files contains files that were recently opened. Number of showed files can be set in application settings.

2.1.1.1.9 Print

File->Print will show dialog for printing current chip design.

2.1.1.1.10 Exit

File->Exit ends the application. There should be check if user wants to exit application without saving last changes on chip design file.

2.1.1.2 Edit

The Edit menu contains commands for manipulating chip elements during editing.

2.1.1.2.1 Undo

Edit->Undo undoes the most recently completed undoable command. Number of undo steps can be set in application settings.

2.1.1.2.2 Redo

Edit->Redo redoes the most recently undone command. Number of redo steps can be set in application settings.

2.1.1.2.3 Cut

Edit->Cut removes and copies the currently selected element to windows clipboard.

2.1.1.2.4 Copy

Edit->Copy copies currently selected element to windows clipboard.

2.1.1.2.5 Paste

Edit->Paste inserts most recently cut or copied element from windows clipboard to active layer.

2.1.1.2.6 Delete

Edit->Delete removes currently selected element.

2.1.1.2.7 Snap to Grid

Edit->Snap to Grid enables/disables snapping to grid.

2.1.1.2.8 Snap to Guidelines

Edit->Snap to Guidelines enables/disables snapping to guidelines.

2.1.1.3 View

The view menu contains command that may help user to make his working environment friendly.

2.1.1.3.1 Rulers

View->Rulers shows or hides rulers in the chip design window.

2.1.1.3.2 Grid

View->Grid shows or hides grid in the chip design window.

2.1.1.3.3 Guidelines

View->Guidelines shows or hide guidelines. Guidelines are lines pulled out of the ruler used for aligning objects.

2.1.1.3.4 Zoom In

View->Zoom In zooms in view on chip design window. For zoom definition see 2.3.3.1.

2.1.1.3.5 Zoom Out

View->Zoom Out zooms out view on chip design window. For zoom definition see 2.3.3.1.

2.1.1.3.6 Fit to Substrate

View->Fit to Substrate fits zoom such that whole substrate is visible.

2.1.1.3.7 Fit to Height

View->Fit to Height zooms such that substrate fits the window on height.

2.1.1.3.8 Fit to Width

View->Fit to Width zooms such that substrate fits the window on width.

2.1.1.4 Layer

The Layer menu contains commands for adding and deleting layers in active chip design project.

2.1.1.4.1 New Layer

Layer->New Layer command shows dialog window that guides user in adding new layer to active chip design.

2.1.1.4.2 Delete Layer

Layer->Delete Layer command deletes active layer from the current chip design.

2.1.1.5 Arrange

The Arrange menu contains commands for manipulating selection of objects.

2.1.1.5.1 Group

Arrange->Group groups selected elements such they behave as a one unit. Operations that are performed on a group are applied equally to each of its objects.

2.1.1.5.2 Ungroup

Arrange->Ungroup is a reverse command to group. It divides the group to former elements.

2.1.1.5.3 Ungroup All

Arrange->Ungroup All divides group tree to former elements, does not matter how many group commands were applied before (i.e. this is particularly helpful if user wants to break apart the group that was made of a group and element)

2.1.1.5.4 Weld

Arrange->Weld command combines selected elements to new one.

2.1.1.5.5 Trim

Arrange->Trim command trims one element from another one.

Typical use is:

- select objects you want to trim
- trim command
- select the object that trims the selection

2.1.1.5.6 Intersect

Arrange->Intersect command produce new element that is intersection of selected elements.

2.1.1.5.7 Align Left

Arrange->Align Left command aligns two elements such that left sides of respective bounding boxes coalesce.

2.1.1.5.8 Align Right

Arrange->Align Right command aligns two elements such that right sides of respective bounding boxes coalesce.

2.1.1.5.9 Align Top

Arrange->Align Top command aligns two elements such that top sides of respective bounding boxes coalesce.

2.1.1.5.10 Align Bottom

Arrange->Align Bottom command aligns two elements such that bottom sides of respective bounding boxes coalesce.

2.1.1.5.11 Align Centers Horizontally

Arrange->Align Centers Horizontally command aligns two elements such that centers of respective bounding boxes coalesce in horizontal direction.

2.1.1.5.12 Align Centers Vertically

Arrange->Align Centers Vertically command aligns two elements such that centers of respective bounding boxes coalesce in vertical direction.

2.1.1.6 Tools

Tools menu contains tools for setting the working environment of the editor, advanced visualization of the chip design, checking and estimating price of the chip design. There are also commands for updating over the internet.

2.1.1.6.1 Settings

Tools->Settings pops-up settings dialog where will be possible to set working environment settings, such as color, length of history, default directories and settings of particular functions.

2.1.1.6.2 Cut View

Tools->Cut View shows window with visualization of the cut thorough the chip design.

2.1.1.6.3 3D View

Tools->3D View shows window with 3D visualization of the chips design.

2.1.1.6.4 Check Constraints

Tools->Check constraints start utility that checks current chip design whether it is ready for manufacturing.

2.1.1.6.5 Estimate Chip Cost

Tools->Estimate Chip Cost estimates price for current chip design.

2.1.1.6.6 Update Production Settings File

Tools->Update Production Settings File checks if there is new file with material definitions and laser device parameters available and offers the download of the file.

2.1.1.6.7 Update the FlashPOM Editor

Tools->Update the FlashPOM Editor checks whether new version of the software is available and offers the download.

2.1.1.7 Window

The Window menu contains commands for showing or hiding tool windows and toolbars.

2.1.1.7.1 Library

Window->Library shows or hides Library tool window.

2.1.1.7.2 Layers

Window->Layers shows or hides Layers tool window.

2.1.1.7.3 Properties

Window->Properties shows or hides Properties tool window.

2.1.1.7.4 Standard

Window->Standard shows or hides Standard toolbar.

2.1.1.7.5 Creation Tools

Window->Creation Tools shows or hides Creation Tools toolbar.

2.1.1.7.6 Helpers

Window->Helpers shows or hides Helpers toolbar.

2.1.1.7.7 Fast Settings

Window->Fast Settings shows or hides Fast Settings toolbar.

2.1.1.7.8 Align

Window->Align shows or hides Align toolbar.

2.1.1.7.9 Status bar

Window->Status bar shows or hides Status bar.

2.1.1.7.10 Reset Window Positions

Window->Reset Window Positions sets tool window positions to default values.

2.1.1.8 Help

2.1.1.8.1 About

Help->About command shows dialog window with common information about the application,

2.1.1.8.2 User Reference

Help->User Reference command opens help for the application. This should be in .mht or .pdf format.

2.1.2 Toolbars

Toolbars are menus with graphical buttons that allows user easy and fast access to most utilized functions.

2.1.2.1 Standard toolbar

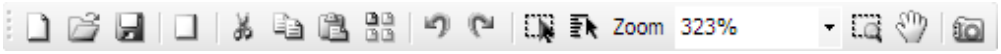

Figure 2-3: Standard toolbar

2.1.2.1.1 New Project

See 2.1.1.1.1

2.1.2.1.2 Open Project

See 2.1.1.1.2

2.1.2.1.3 Save Project

See 2.1.1.1.3

2.1.2.1.4 New Layer

See 2.1.1.4.1

2.1.2.1.5 Cut

See 2.1.1.2.3

2.1.2.1.6 Copy

See 2.1.1.2.4

2.1.2.1.7 Paste

See 2.1.1.2.5

2.1.2.1.8 Distribute

Distribute function helps user to make copies of selected element placed into rectangular array.

2.1.2.1.9 Undo

See 2.1.1.2.1

2.1.2.1.10 Redo

See 2.1.1.2.2

2.1.2.1.11 Select and Edit

Select and edit tools will enable user to select element or group of elements either by clicking on each or by selection rectangle.

2.1.2.1.12 Select by Name

Select by name starts up the dialog window that helps user to select element by name or placement in layer.

2.1.2.1.13 Zoom

Magnify the view on chip design window. Precise value can be set from keyboard or selected from listbox. For zoom definition see 2.3.3.1.

2.1.2.1.14 Zoom Area

Zoom area tool zooms rectangular selection to fit window of the screen.

2.1.2.1.15 Move Working Area

Move working area helper enables user to move working area such that design part of interest is in the center of the screen.

2.1.2.1.16 Take Snapshot

Take snapshot command takes the snapshot of current workspace and offers user to save it.

2.1.2.2 Creation Tools toolbar

Creation Tools toolbar contains tools for creating and editing objects.

Figure 2-4: Creation Tools toolbar

2.1.2.2.1 New Line

New Line tool enables user to add line to active layer.

2.1.2.2.2 New Rectangle

New Rectangle tool enables user to add rectangle to active layer.

2.1.2.2.3 New Ellipse

New Ellipse tool enables user to add ellipse to active layer.

2.1.2.2.4 Weld

See 2.1.1.5.4

2.1.2.2.5 Trim

See 2.1.1.5.5

2.1.2.2.6 Intersect

See 2.1.1.5.6

2.1.2.2.7 Flip Vertically

Flip vertically command flips the selected object around the centre from top to bottom.

2.1.2.2.8 Flip Horizontally

Flip horizontally command flips the selected object around the centre from left to right.

2.1.2.2.9 Rotate 90° clockwise

Rotate 90° clockwise command turns the object around the transformation centre point by 90° clockwise.

2.1.2.2.10 Rotate 90° degrees counter-clockwise

Rotate 90° counter-clockwise command turns the object around the transformation centre point by 90° counter-clockwise.

2.1.2.3 Fast Settings toolbar

Fast settings toolbar allows user to enter nudge distance and grid distance parameters directly what is faster than entering them in the Tools->Settings dialog.

Figure 2-5: Fast Settings toolbar

2.1.2.3.1 Nudge distance

When element is selected user can move it by pressing cursor keys. The distance object is moved is called nudge distance and can be entered directly using fast setting toolbar.

2.1.2.3.2 Grid distance

Grid distance is the interval between respective horizontal and vertical grid lines.

2.1.2.4 Align toolbar

Align toolbar contains functions to align one element to another.

Figure 2-6: Align toolbar

2.1.2.4.1 Align Left

See 2.1.1.5.7

2.1.2.4.2 Align Right

See 2.1.1.5.8

2.1.2.4.3 Align Top

See 2.1.1.5.9

2.1.2.4.4 Align Bottom

See 2.1.1.5.10

2.1.2.4.5 Align Centers Horizontally

See 2.1.1.5.11

2.1.2.4.6 Align Centers Vertically

See 2.1.1.5.12

2.1.2.5 Helpers toolbar

Helpers toolbar contains tools that helps user to select and position objects precisely.

Figure 2-7: Helpers toolbar

2.1.2.5.1 Select elements inside selection rectangle only

Using rectangular selection user can select objects that are intersected by the selection rectangle as well as objects that are inside. While this option is turned on only objects inside rectangular selection are selected.

2.1.2.5.2 Show/Hide Rulers

See 2.1.1.3.1

2.1.2.5.3 Show/Hide Grid

See 2.1.1.3.2

2.1.2.5.4 Snap to Grid

See 2.1.1.2.7

2.1.2.5.5 Show/Hide Guidelines

See 2.1.1.3.3

2.1.2.5.6 Snap to Guidelines

See 2.1.1.2.8

2.1.2.5.7 Show/Hide Cross Cursor

Cross cursor is tool that can help user in certain situations. It draws cursor horizontal and vertical line above the whole workspace.

2.1.3 Tool windows

2.1.3.1 Properties

Purpose of the properties window is to show and allow user to modify properties of selected object (chip, layer, element, grid, guideline). Listbox is filled with flatten list of all objects which properties can be changed.

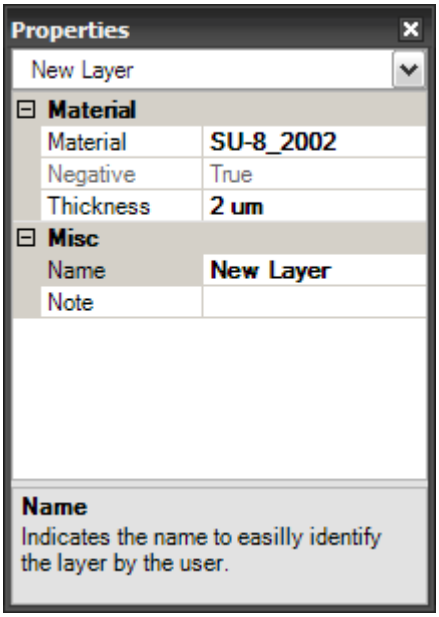

Figure 2-8: Properties tool window

2.1.3.2 Layers

The Layers tool window provides view on all chip layers. Appearance of each layer (color, transparency, visibility) in chip design window can be set there. There is also possibility to lock layer to avoid unwanted element manipulation.

Order of layers can be changed by up/down buttons or by drag and drop.

New layer can be added or deleted by New/Delete button.

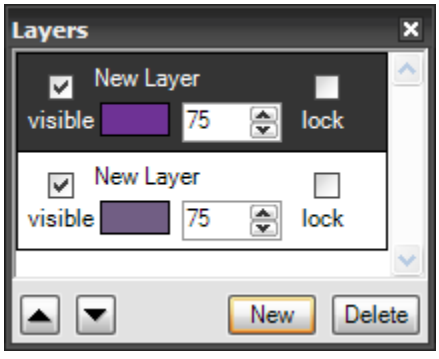

Figure 2-9: Layers tool window

2.1.3.3 Objects Library

The Library tool window manages user element libraries. New library is created by New button. Created library is deleted by Delete button. Any element can be inserted into created library by selecting element and pressing insert button. Element can be placed to chip layer by pressing Place button or by drag and drop. Elements can be also deleted from the library by Delete button.

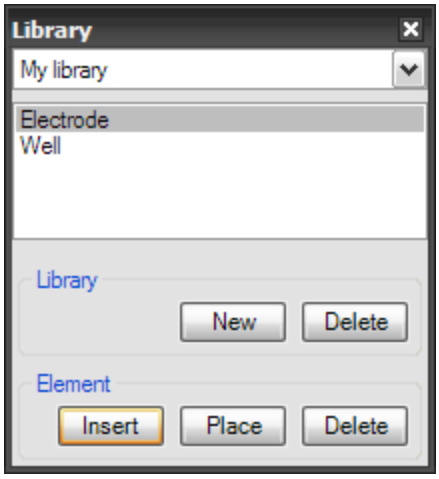

Figure 2-10: Objects Library tool window

2.1.4 Working area window

Working area window is the drawing area of the application. It is the place where user designs the chip. Depending on settings in Layers tool window there can be viewed all layers (with different colors and transparency). Rulers, grid a guidelines can be shown to make placing of objects easier and more precise.

Multiple projects can be opened, then for each project new window with new tab is added.

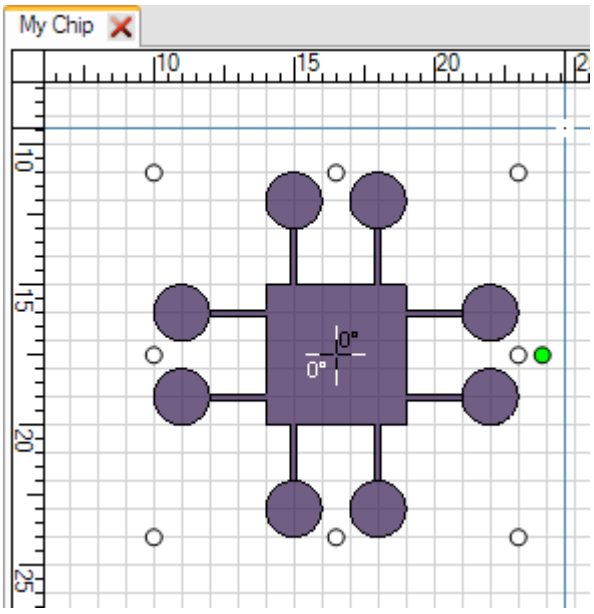

Figure 2-11: Working area tabbed window

2.1.5 Status bar

Status bar displays brief information about selected objects, current mouse position, relevant commands or fast hints.

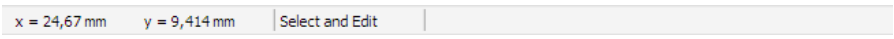

Figure 2-12: Status bar

2.1.6 Keyboard shortcuts

Keyboard shortcuts provide fast access to most important functions of the application.

File	
Ctrl N	New
Ctrl O	Open
Ctrl S	Save
Ctrl Shift S	Save As
Ctrl Q	Close
Ctrl P	Print
Alt F4	Exit

Edit	
Ctrl Z	Undo
Ctrl Y	Redo
Ctrl X	Cut
Ctrl C	Copy
Ctrl V	Paste
Del	Delete

View	
Ctrl +	Zoom In
Ctrl -	Zoom Out
Ctrl *	Fit to Substrate

Arrange	
Ctrl G	Group
Ctrl U	Ungroup
Ctrl W	Weld
Ctrl T	Trim
Ctrl I	Intersect

Align	
Ctrl Alt L	Align Left
Ctrl Alt R	Align Right
Ctrl Alt T	Align Top
Ctrl Alt B	Align Bottom
Ctrl Alt H	Align Centers Horizontally
Ctrl Alt V	Align Centers Vertically

Window	
Ctrl Shift B	Library
Ctrl Shift L	Layers
Ctrl Shift P	Properties
Ctrl Shift C	Cut View
Ctrl Shift 3	3D View

New Element		
Ctrl	Rectangle	Proportional size (Square)
	Ellipse	Proportional size (Circle)

Edit Element		
Ctrl	Rotation	Step 15 degrees
	Size	Proportional size change
	Move	Move in vertical or horizontal direction only

2.2 File formats

2.2.1 FPSVG - Output language for vector image definition, based on SVG (Scalable Vector Graphics)

2.2.1.1 Introduction

In this document is explained the necessary syntax in order to define the output file for a chip specification file. We must consider that chip is made on different layers. In order to specify every layer we use a graphic language derived from XML, the SVG (Scalable Vector Graphics).

This language has been adapted to our needs but it is very near to the standard SVG language.

File (and also the designed chip) is divided into different layers and every layer into different shapes in order to describe the draw. Finally every layer has, also, some necessary options like, substrate depth and others. The output file will include the specification for the chip, layers and shapes. Every path, shape, or layer, ends with a carry return. The whole syntax is specified on the following document.

2.2.1.2 Basic Shapes

Properties can be of the following data types:

boolValue: bool value (i.e.: style=”fill:true;”)

intValue: 32bit integer value (i.e.: numberOfLayers=“5”)

floatValue: float value,

dot is the decimal separator, four decimal positions (i.e.: width=“1.2345”)

strValue: string value (i.e.: material=”SU-8”)

2.2.1.2.1 Rectangle

<rectangle x=”floatValue” y=”floatValue” width=”floatValue” height=”floatValue” style=”fill:boolValue; stroke-width:floatValue; rotate:floatValue;” />

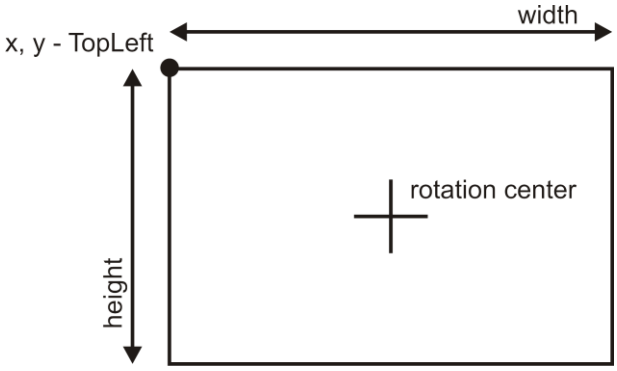

Figure 2-13: Rectangle properties

x, y: upper-left point of the rectangle

width: rectangle width

height: rectangle length

fill: kind of fill

stroke-width: line width

Rotation center is implicitly defined as x = topLeft.X + width/2, y = topLeft.Y + height/2.

2.2.1.2.2 Circle

<circle cx=”floatValue” cy=”floatValue” r=”floatValue” style=”fill:boolValue; stroke-width:floatValue; rotate:floatValue;” />

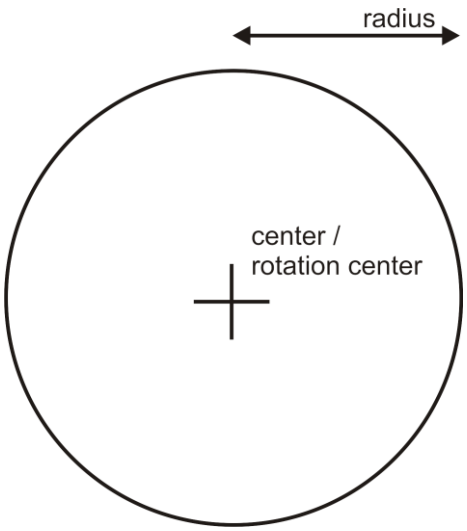

Figure 2-14: Circle properties

cx: center x

cy: center y

r: radius

rotation center is implicitly defined as x = cx, y = cy.

2.2.1.2.3 Ellipse

<ellipse cx=”floatValue” cy=”floatValue” rx=”floatValue” ry=”floatValue” style=”fill:boolValue; stroke-width:floatValue; rotate:floatValue;” />

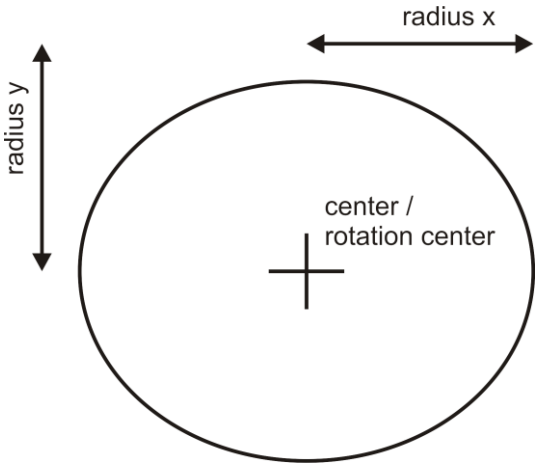

Figure 2-15: Ellipse properties

cx: ellipse center x

cy: ellipse center y

rx: radius x

ry: radius y

Rotation center is implicitly defined as x = cx, y = cy.

2.2.1.2.4 Line

<line x1=”floatValue” y1=”floatValue” x2=”floatValue” y2=”floatValue” style=”stroke-width:floatValue;” />

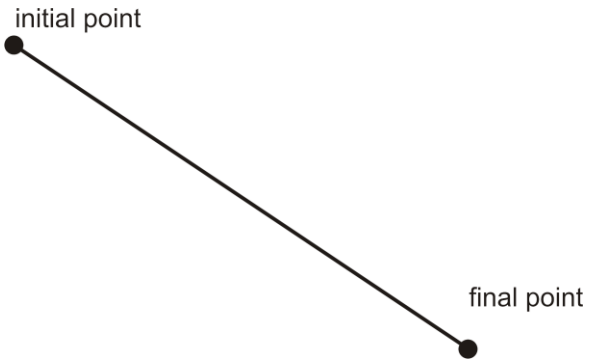

Figure 2-16: Line properties

(x1,y1): initial point

(x2,y2): final point

stroke-width: line width

2.2.1.2.5 Shape options:

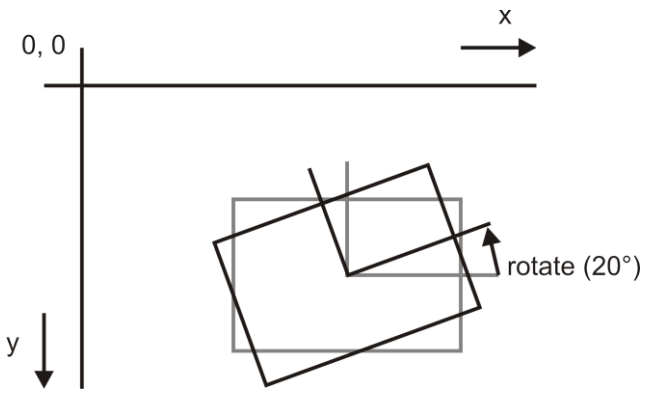

Figure 2-17: Rotation

rotate(angle): rotate the shape on the angle specified (in degrees)

stroke-width: line width

fill: 0 : shape is empty

1: shape is filled

Please note that chip is aligned so that top left is in the upper left corner. X axis grows to the right and Y axis grows down.

2.2.1.3 Path command

Command used in order to draw shapes that no include basic shapes, the path command uses the following syntax:

<path style=”fill:boolValue; stroke-width:floatValue; rotate:floatValue;” d="strValue" />

Rotation center is implicitly defined as a center of internally computed path bounding box(minX, minY, maxX, maxY).

Path (d) command options

M: moves from the actual point to the point x,y without marking a line.

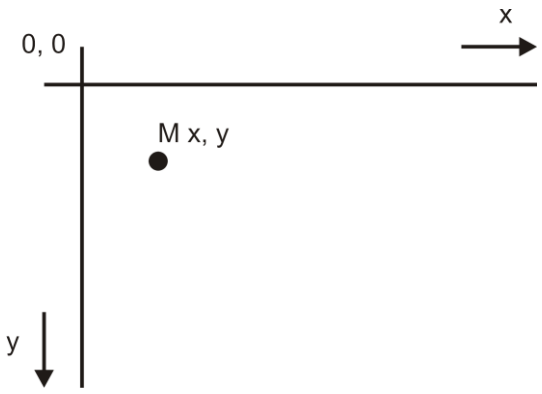

Figure 2-18: Path move command

Syntax: *M x, y (floatValue, floatValue)*

L: draws a line from the actual point to x,y

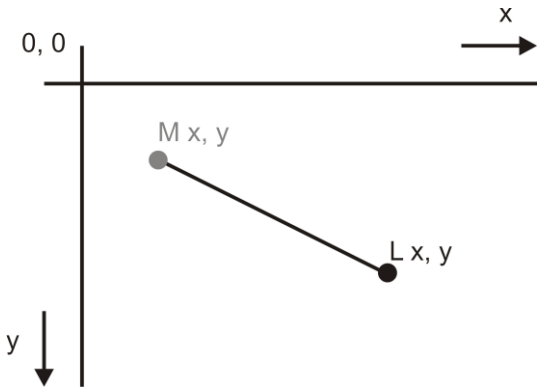

Figure 2-19: Path line command

Syntax: *L x, y (floatValue, floatValue)*

A: draws an elliptical arc from actual position to x,y with the following syntax:

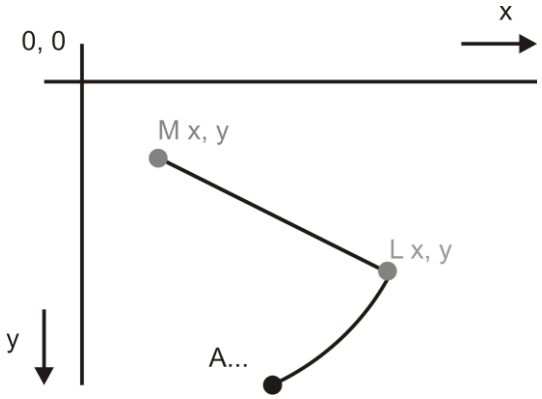

Figure 2-20: Path arc command

Syntax: *A rx ry x-axis-rotation large-arc-flag sweep-flag x y)*

rx: ellipse radius x *(floatValue)*

ry: ellipse radius y *(floatValue)*

x-axis-rotation: x axis rotation respect x axis *(floatValue)*

large-arc-flag: If 0: choose the shorter arc *(intValue)*

if 1: choose the longer arc

sweep-flag: If 0: negative arc *(intValue)*

if 1: positive arc

x,y: final point of the arc

Two points define two ellipses, we will define the arc chosen with the flags, there is an explanation of the flag options in Figure 2-21.

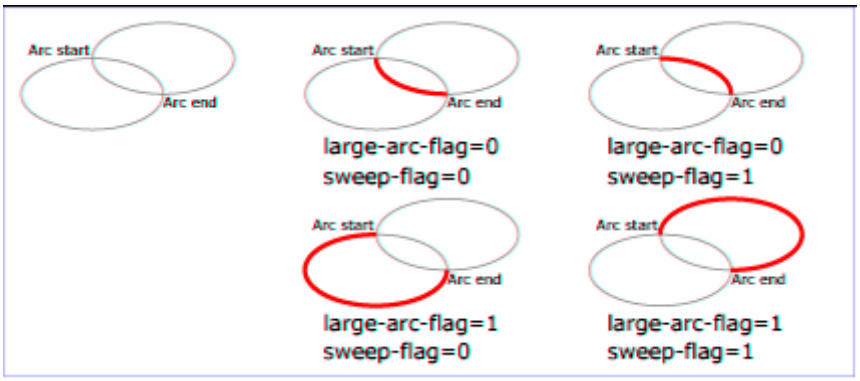

Figure 2-21: Flags explanation

Z: Draws a line from final to initial point of the shape, closes the shape.

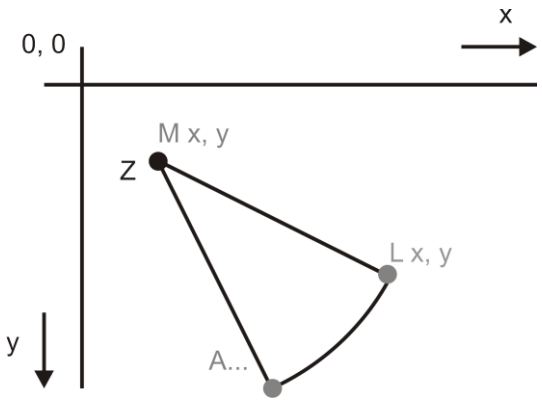

Figure 2-22: Path close command

Path command can include fill and stroke-width options.

2.2.1.4 File Syntax

A chip file will be composed by several layers that will include a drawing, made by different basic shapes or path.

File is a standard xml file thus it starts with xml header:

<?xml version="1.0" encoding="utf-8"?>

In order to specify the start and ending of a chip file use:

<chip name=”strValue” material=”strValue” shape=”strValue” [width=”floatValue” height=”floatValue” | radius=”floatValue”] materialThickness=”floatValue" numberOfLayers=”intValue”> /* start of the chip

/* layer list

</chip> /* chip end

Volume will be formed by different layers. Every layer should specify the substrate depth of the layer, layer width and layer length, and also the material of the layer. Define the layers using the following syntax:

<layer name=”strValue” material=”strMaterial” thickness=“floatValue” negative=”boolValue” number=“intValue”> /* layer start

/* shapes and paths list

</layer> /* layer ending

Differences between this language and standard SVG

- Fill is not a color (for example in SVG fill:green;) , fill it's only a flag.

- Start and ending of the file are different in SVG. We have adapted them to our chip-oriented function.

- We have deleted some extra options from standard SVG like *transparency*.

- In SVG the value comes with the unit (for example : x=”13cm”) in our case all **units** are in **micrometers** and we eliminate this from our language.

In order to see the original SVG image, we should make very simple changes in our language:

- Add the unit (micrometer in our case) to the value.

- Change layer headers and volume headers for valid XML headers.

- Change the fill flag for a color, for example: fill:black;

For a better comprehension we attach here an example of a complete file for the sample design shown on Figure 2-23:

```
<?xml version="1.0" encoding="utf-8"?>

<chip name="My Chip" material="CPyrex" shape="Circular" diameter="76.2"
materialThickness="1500000" numberOfLayers="2">

  <layer name="Layer One" material="SU-8_2002" thickness="2000" negative="True" number="0">

    <rectangle x="2.1E+07" y="1.1E+07" width="6000000" height="6000000" style="fill:True;
    stroke-width:100; rotate:0;" />

    <circle cx="3.3E+07" cy="1.4E+07" r="3000000" style="fill:True; stroke-width:100;
    rotate:0;" />

    <line x1="3.9E+07" y1="1.7E+07" x2="4.5E+07" y2="1.1E+07" style="stroke-width:100;" />

  </layer>

  <layer name="Layer Two" material="SU-8_2002" thickness="2000" negative="True" number="1">

    <ellipse cx="2.6E+07" cy="2.1E+07" rx="5000000" ry="2000000" style="fill:True;
    stroke-width:100; rotate:0;" />

    <path style="fill:True; strokeWidth:100; rotate:0;" d=" M4E+07 2E+07 L4,3E+07 2E+07
    L4,3E+07 2,1E+07 L4E+07 2,1E+07 L4E+07 2,3E+07 L3,5E+07 2,3E+07 L3,5E+07 1,9E+07 L4E+07
    1,9E+07 L4E+07 2E+07Z" />

  </layer>

</chip>
```

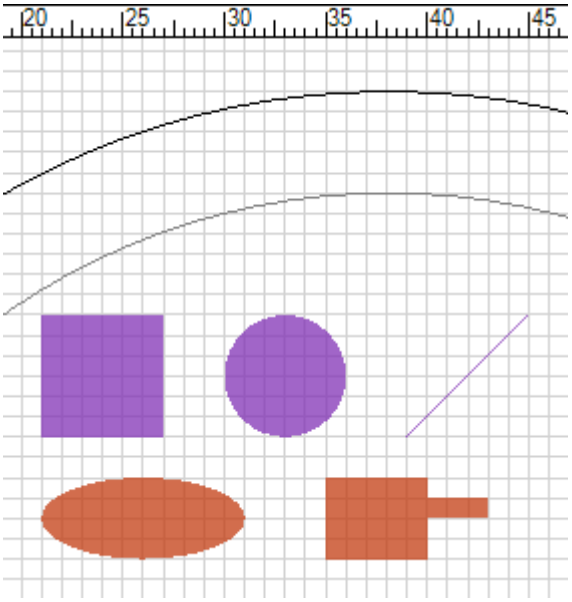

Figure 2-23: Sample design

2.2.2 FPM – FlashPOM proprietary file format

FPM file format is binary format for saving chip design projects. It is compressed XML image of the design and user settings.

2.2.3 File with available materials and cost function parameters

This file is intended to be placed on the internet together with timestamp file so the user is allowed to update the file regularly.

File is in standard XML format and contains following data.

2.2.3.1 Laser resolution

Value represents resolution of the laser device in nm units.

2.2.3.2 Unit surface price

Value means price for area of 1nm^2 in Eur.

2.2.3.3 Substrate materials

For every material there are listed following properties:

2.2.3.3.1 Name

Name is identification of the material and has to be unique.

2.2.3.3.2 Shape

Substrate material can have circular or rectangular shape.

2.2.3.3.3 Radius

If material is circular shaped then Radius specifies size of the layer in millimeter units.

2.2.3.3.4 Substrate Width

If material is rectangular shaped then Width specifies the size/width of the layer in millimeter units.

2.2.3.3.5 Substrate Height

If material is rectangular shaped then Height specifies the size/height of the layer in millimeter units.

2.2.3.3.6 Material Thickness

Value defines Thickness of the substrate material in micrometer units.

2.2.3.3.7 Border distance

This value will be obtained experimentally. Currently it is set to about 5mm what is reasonable reserve. It defines distance in millimeter units from the border of the material where material properties have homogenous characteristics.

2.2.3.3.8 Price

Price of the material is defined in Euros.

2.2.3.4 Layer materials

For every material there are listed following properties:

2.2.3.4.1 Material Name

Name is identification of the material and has to be unique.

2.2.3.4.2 Negative

Value defines whether material is negative or positive. Positive material means that what gets exposed by laser will go away. Allowed values are true/false.

2.2.3.4.3 Resolution

Resolution of the material defines the smallest size of the dot that can be made on the layer. Value is defined in micrometer units.

2.2.3.4.4 Minimum Thickness

Minimum Thickness stands for minimal manufacturable thickness of the material. Value is in micrometer units.

2.2.3.4.5 Maximum Thickness

Maximum Thickness stands for maximal manufacturable thickness of the material. Value is in micrometer units.

2.2.3.4.6 Price

Price of the material is defined in Euros.

2.2.3.5 Compatible materials

Part compatible materials contain enumeration of compatible material couples. While designing the chip it depends on which layer is lower and which upper. Compatible couple record reflects this need.

For a better comprehension, example of a complete file:

```
<?xml version="1.0"?>

<productionSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <laserResolution>200</laserResolution>

  <unitSurfacePrice>1E-06</unitSurfacePrice>

  <substrateMaterials>

    <substrateMaterial name="CPyrex" shape="Circular" radius="20" substrateWidth="0"
    substrateHeight="0" materialThickness="1500" borderDistance="5" price="20" />

    <substrateMaterial name="RPyrex" shape="Rectangular" radius="0" substrateWidth="36"
    substrateHeight="36" materialThickness="1500" borderDistance="5" price="20" />

      <substrateMaterial name="CSilicon" shape="Circular" radius="50.8" substrateWidth="0"
      substrateHeight="0" materialThickness="300" borderDistance="5" price="40" />

    </substrateMaterials>

    <layerMaterials>

      <layerMaterial name="SU-8_2002" negative="true" resolution="1" minimumThickness="2"
      maximumThickness="3" price="50" />

      <layerMaterial name="SU-8_2100" negative="true" resolution="1" minimumThickness="100"
      maximumThickness="270" price="70" />

      <layerMaterial name="SPR-220-7.0" negative="false" resolution="1" minimumThickness="5.5"
      maximumThickness="10" price="50" />

    </layerMaterials>

    <compatibleMaterials>

      <compatibleCouple upper="CPyrex" lower="RPyrex" />

      <compatibleCouple upper="RPyrex" lower="CPyrex" />

      <compatibleCouple upper="RPyrex" lower="SU-8_2002" />

      <compatibleCouple upper="SU-8_2100" lower="SU-8_2002" />

    <compatibleCouple upper="SU-8_2100" lower="SPR-220-7.0" />

  </compatibleMaterials>

</productionSettings>
```

2.2.4 Timestamp file

Time stamp file is loaded on web server together with program installation files production settings file. Application compares this file to latest updates and depends whether to update application or production settings file.

Sample of timestamp file:

```
<?xml version="1.0"?>

<dateTime>2007-05-31T14:57:59.5218714+02:00</dateTime>
```

2.3 System-Specific Requirements

2.3.1 Elementary objects

Elementary objects that can be easily placed by user to chip layer are line, rectangle and ellipse.

2.3.2 Program functions

2.3.2.1 Cost function

Cost function will be fixed in program. Parameters of the function are to be stored in the file with available materials and cost function parameters, see 2.2.2.

Equation for computing cost of the chip is following:

$$chipprice = substrateprice + \sum_{layers} layerprice + \sum_{layers} unitsurfaceprice * elementssurface$$

Figure 2-24: Equation for chip price with negative layers.

$$chipprice = substrateprice + \sum_{layers} layerprice + \sum_{layers} wholesurfaceprice - (unitsurfaceprice * elementssurface)$$

Figure 2-25: Equation for chip price with positive layers.

2.3.2.2 Constraints

Program should have function that will check whether chip designed by user can be manufactured.

2.3.2.2.1 Layer compatibility

Program has to check whether stacked layers are compatible between themselves.

2.3.2.2.2 Boundary

Program has to check whether all elements are inside safe substrate boundary.

2.3.2.2.3 Overlapping elements within one layer

Program should check whether element overlaps another one. This in production phase prevents laser to burn on one position two times.

2.3.2.2.4 Support in layer below

Program should check whether elements in upper layer have support elements in layer below.

2.3.2.3 Print

User has to be able to print designed chip. Dialog window pops-up and user can adjust layout of printed page.

2.3.3 Chip design functions

2.3.3.1 Zoom

Zoom function allows user to magnify the view on chip. 100% of zoom means that 1mm on the computer monitor equals to 1mm on chip substrate.

2.3.3.2 Distribute

Distribute function helps user to make copies of selected element placed into rectangular array. Number of copies in horizontal/vertical direction is to be set in distribute dialog. Horizontal/vertical distance between respective copies can be set there as well. User can choose whether the distance is measured as a distance between element centers or as a distance between element bounding box borders.

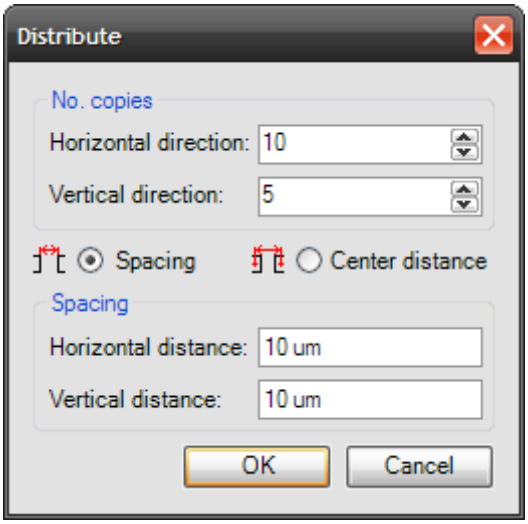

Figure 2-26: Distribute dialog

2.3.4 Chip visualization functions

2.3.4.1 Cut View

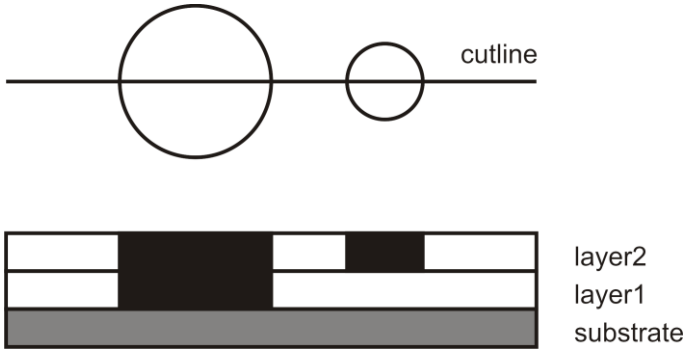

Figure 2-27: Visualization of the chip cut

User can select the cut line, than in special window cut thorough all chip layers will be shown.

2.3.4.2 3D chip visualization

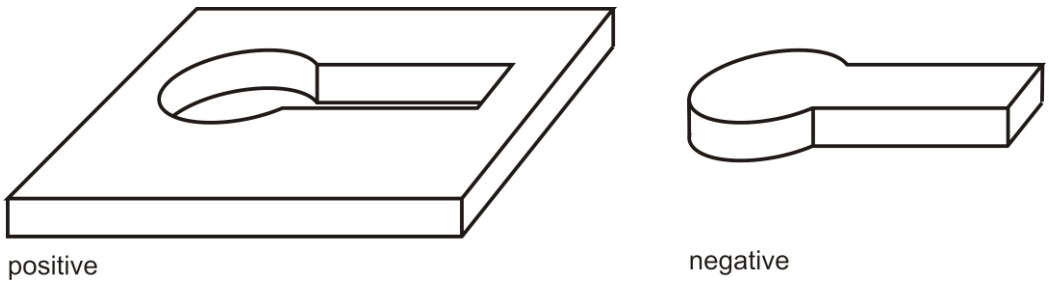

Figure 2-28: 3D chip visualization of positive and negative layer

Program should have a function to show 3D preview of the chip or at least 3D preview of the area of interest on the chip. User will be allowed to switch on/off visualization of each layer.

2.3.5 Update

Update function should allow user to update production settings file and the application.

3 Non-Functional Requirements

3.1 System-Related Non-Functional Requirements

3.1.1 Operational Environment

3.1.1.1 Hardware Platform

Intel Pentium III processor or compatible
Direct3D compatible graphics card (only in case of 3D Visualization)

3.1.1.2 Software Platform

Microsoft .NET Framework 2.0
Microsoft Direct3D 9.0c for Managed Languages (only in case of 3D Visualization)

3.1.2 Time plan

11-13.9.2006	Kick Off Meeting Barcelona
25.9.2006	Expected induction meeting with CZ team members
23.10.2006	Start of the BETA version implementation
1.12.2006	BETA version testing
15.12.2006	Release of the BETA version
1.1.2007	Start of the RELEASE version implementation
11.6.2007	RELEASE version testing
30.6.2007	Release of the RELEASE version

3.2 User Interview Transcripts

Q: Should 3D Visualization be part of the editor or standalone application?
A: Some kind of 3D Visualization would be nice, especially when we are making papers or posters. But it is not a must. We don't care if rendering part will be the part of application. In example it can be done as export to file with POV-RAY format. For the same reason it would be nice to have an export to EPS format. Not if this means that the program doesn't run without Direct3D.

Q: How many undo/redo steps there should be?
A: Yes, we would like to have undo/function. We should be able to set number of steps in program properties.

Q: Is there requirement on some kind of versioning?
A: No.

Q: Future enhancements. I.e. C-like chip definition language, Input/output ports on user defined objects etc.
A: Maybe in future versions, but it's not a question of the day.

Q: After some discussion we came to conclusion that all chip layers will be aligned to top left corner, where will be the coordinate's origin. What is not clear if each layer of the chip can have different size (i.e. if layers will be made from different materials)?
A: Yes all the layers have the same size.

But - and this here is new - layer != mask.

Sometimes we take a small area of the layer and copy it several times to the mask (in arrays). We do this when the features in the layer are very small compared to the whole chip size. Then we create arrays of these features to increase the yield of the fabrication process.

Therefore I propose:

- **a layer object with width,height for designing the chip with its features**
- **a region (bounding box) can be selected which encloses the features of interest. This area is then copied as a pattern into a**
- **mask object**

The mask has its own width,height. The pattern from the layer object gets copied into the mask multiple times in array form. Please have look at the attachment Mask-Layer-Relation.pdf for a visual description of the relation. As you can see it would be nice to have the option to add bounding boxes and text to the mask. This way it's easier to distinguish the masks (in the end they look very similar).

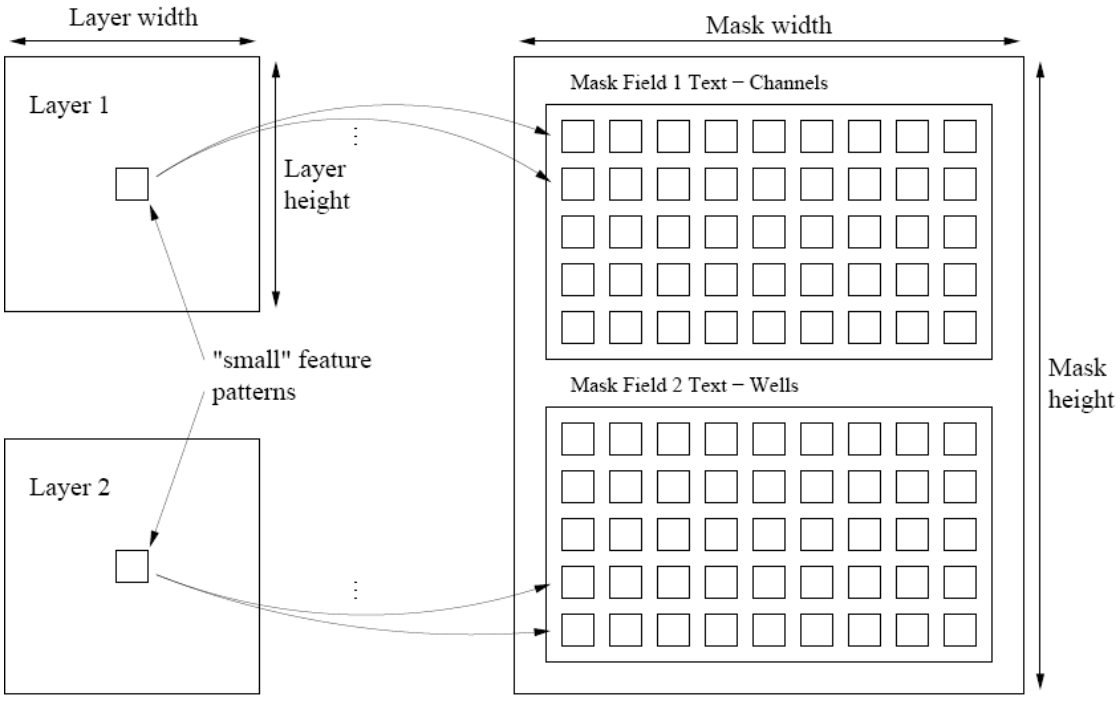

Figure 3-1: Mask-Layer-Relation

Q: What are the constraints while designing chip (i.e. what is the minimal distance between the placed rectangles, lines (electrodes))?
A: It depends on the mask fabrication process. With our current foil mask the resolution is effectively somewhere around 5-10µm. The laser process might be better. But I wouldn't spend time and energy on this at the moment. In my opinion this is a feature for future versions.

Q: Is there any constraint on line crossing? Is it possible that one line cross another? What the program should do (not allow this situation / mark these objects when "check button" will be pressed / do nothing – user is smart enough to avoid this situation)?
A: When doing masks in Freehand I often let lines cross. But these are all lines that belong to the same structure e.g. an electrode. Have a look at the attachment Line-Overlap.pdf to see why.

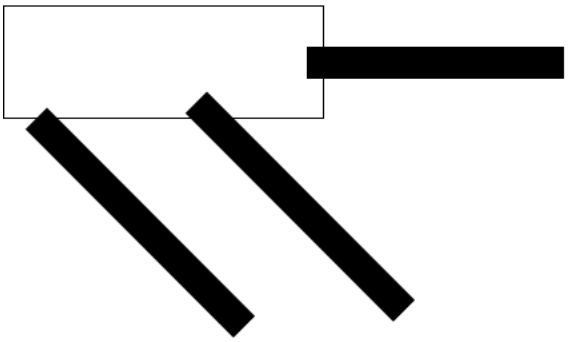

Figure 3-2: Line overlap

The crossing on the left is not "clean" and therefore I move the line deeper into the other object to get a cleaner connection.

I also do it sometimes with connections like the one on the right which should not be critical. But I saw some Postscript-Interpreters handle such small features and the results were terrible. They are at the limits of their resolution and then rounding errors can lead to two such features separating in the final print. Therefore I introduce an overlap in a attempt to counter such rounding errors.

I wouldn't do checking in this version.

Another feature might be a de-overlapping function which will be needed for the laser control. For the printed masks it does not matter if an area is painted black twice but if the laser exposes the photo resist two times at the same place this could cause problems. See attachment De-overlapping.pdf.

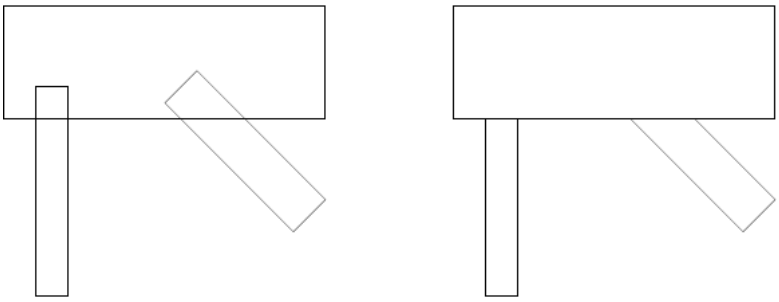

Figure 3-3: De-overlapping

But I don't know where is the best place to implement this functionality, in the design program or in the laser control program?

Q: Should be the constraints restrictive (i.e. while designing chip should be user allowed placing objects closer than is allowed – than there can be button after it's activation dangerous objects will be marked)?
A: No, not in this version. And then only warnings, no restrictions. If the user wants to shoot in his foot the program shouldn't hold him back. ;)

Q: What are proposed cost estimations constraints?
A: Right now we don't have any sensible cost function. The printed masks cost the same - regardless what's on them. The laser might be different, some masks might be faster to expose than others. But this will turn out during the first trials with the laser control program.
So drop this feature for this version.

Q: What should be the resolution – this means what is the smallest recognizable unit and what is the maximal size of layer (i.e. Let's say that we will use 32bit unsigned integer (0-4294967296) for storing position information, smallest unit is 1nm, than maximum size of the layer is 4,294m. Is it enough?) ?
A: This should be enough, yes. Btw. the range for an unsigned int only reaches to 4294967295. Don't forget the zero. ;)

Q: What means default 100% zoom (i.e. 1px~10nm, or should it be option to set in program settings)?
A: 1px~10nm is surely too small but I also don't know a good default value. It should be an option then to set the value for 100%. Zooming in and out happens quite frequently and comfort for this function would be nice. Maybe you can some extra buttons for it in the toolbar which set the zoom factor to a pre-defined value (quick zoom buttons). The pre-defined values should be an option in the program settings.

Q: This should be kind of advanced option maybe in future versions. Some professional solutions (i.e. AutoCad) have possibility to enter the objects thorough command line. Should it be helpful to have something like console window from where user will be able to control the program only with keyboard and knowledge commands? (Hardness of implementation of this feature depends on functionality we want. It is not so hard to have basic commands like "Add Rectangle 10 10 100 50 0.5" but is harder to have complete script language based i.e. on python.)
A: It should be possible to edit the numerical values of the control point coordinates. But a command interpreter like this is not necessary in the first version.

Q: Should there be a Print function?
A: No.

Q: What should be the maximal/minimal zoom (i.e. 10% to 1000% or equivalent 1nm~1px to 1cm~1px)?
A: 1nm~10px to 10cm~1px - a bit wider. ;)

Q: We are not sure if has any sense to have preview of each layer in Window layers tool window because these images will be so small that in most cases nothing will be seen here. One solution is to display some surroundings of the place that was changed last time.
A: Yes drop the previews. And I'm also not convinced of the usability of the other solution with the recently changed surroundings. Maybe you can add a highlight function? E.g. a double-click onto the layer field causes the features belonging to this layer to be high lighted. This could answer the "To what layer does this feature belong again?" question. ;)

Q: What is the requirement on transformations? The deal is that with i.e. scale, rotate about arbitrary degree, etc. we have to compute in floats and we loose the precision.
A: True but I fear it's inevitable.

- **Scaling *is not* necessary.**
- **Mirroring *is* necessary.**
- **Rotations *are necessary* with arbitrary degrees. :(**

Can you use doubles internally and then convert back to unsigned ints?

Q: Do you missing some "window" or function from your favorite editor?
A: Apart from what I mentioned above, no. The biggest new feature might be the text function for the masks. Here I don't need all the fancy fonts a simple but scalable font is ok.

Q: There was an idea to have for every layer an export checkbox (or something) in Layers window, so that only selected layers will be than exported. Is it valuable function?
A: No, that was my first idea for the masks. Have a look at the mask idea above and forget about this idea.

Q: Will it be handy if every object will have notes field in properties? To write some notes about the particular object, i.e. for other persons who will work on the design.
A: This might be nice to have but not really necessary.

Q: When we rotate the group of objects (2) and then change the size of thegroup (3) the former rectangle in the selection is not more a rectangle – it has to be transformed to path (to be precise to parallelogram). But this is one way process - if we transform the primitive to path then there is no way how to transform it back to rectangle (what is more convenient for laser device). There are two possible solutions:

*** Allow only symmetric size change on group of objects - thus the rectangle still remains to be a rectangle**

*** Allow the transformation to path - with no way back to former primitives**

What is more suitable?

A: I like the idea of retaining the shape type. You should be able to stretch a rectangle along any of the two axis but there is no need to allow the user to drag an individual corner and turn it into a parallelogram.