

MICROPROCESSOR LABORATORY

African Regional Course on Advanced VLSI Design Techniques

24 November - 12 December 2003

Kumasi - Ghana

Exercise 1

Exploring a Chip Layout

Exercise Description

In this exercise you will edit a full chip layout with a specific Alliance tool called **Graal**. This tool allows you to explore the layout of a chip seeing most of the standard layers needed by the designer to implement a CMOS chip.

In this exercise you will

- Edit a CMOS Chip layout.
- Identify the chip core and the ring of Pads.
- Explore the hierarchical structure of the layout.
- Explore the electrical connectivity of different layout elements.
- Edit a standard cell layout.
- Recognise simple CMOS logic gates by inspecting its layout.
- Get used to invoke man pages of standard cells.

(*****)

Exercise guide

Legend



Give the command that appear immediately after this symbol, at the command line.



Explanation of a topic



A VLSI chip is supposed to process information in a wide sense. To do that a large number of transistors are built in and interconnected in order to handle the binary information. A quantum of information is represented by a bit which can assume two values only. Each of these values is physically represented by an electric signal (*voltage*), which assumes two values: *Low* or *High*. The logic-electrical elementary structures are the *gates*. The physical implementation of a *logic gate* is a circuit that “*maps*” a Boolean operation. Some of the main characteristics of these gates are:

- Small area
- Easy interconnection
- High speed
- Low power consumption

These gates are designed to exchange information among them inside the chip, and then they are not prepared to exchange information with the external world in an efficient way. A typical parasitic capacitance of an external chip connection is about 2 or 3 orders of magnitude larger than a typical internal one. Sometimes there are input signals that are spread inside the core of the chip, and due to this, the large associated parasitic capacitance could degrade the signal at some critical point compromising the correct behavior of the chip. As an example of this consider potential critical signals like “*set*”, “*reset*” or “*clock*”. A correct handling of input output signals at chip level is mandatory in order to ensure a correct functioning chip.

Alliance has a library (*padlib*) of specific cells to interface the core of the chip with the external world. These cells are *Pad Drivers*. There are different types of them according to its function like:

- *Input*
- *output*
- *input-output*
- *power-supplies*
- *clock*.
- ...

A common characteristic to all pad-drivers is that they have an uncover square piece of metal (the *pad*) for bonding. The area of this square metal is typically $\sim 100\mu s \times \sim 100\mu s$.

The output drivers must be able to force fast electrical transitions between logic levels loading big parasitic capacitance. During these transitions a large peak of current is required from the power supplies and consequently the voltage of these power supply lines can vary due to a non-zero resistance connection to the voltage source. Then the voltage of the power supply lines for the drivers become *dirty* and it is not advisable to use it to feed the core where many delicate electrical transitions take effect. Hence for the core there are separate power supplies.

In alliance the distribution of power is facilitated by mean of rings around the core. The same strategy for one privileged signal (typically *clock*, although could be any other). For this signal there is a *pad driver* (*pck_sp*) which takes the external signal to drive the ring and another *pad driver* (*pvsseck_sp*) that takes the signal from the ring and drives it to the core.

There are five concentric metallic rings: *vdde* & *vsse* for pad-drivers, *vddi* & *vssi* for the core, and the *clock* ring. All pad-drivers contain five piece of metal in order to generate these rings when they are disposed all around the core. In a structural description this means that at least five ports must be specified for any Pad-driver.

Begin by creating a design directory, at a convenient position in your workspace:



```
mkdir chip-layout
```

Change into this directory:



```
cd chip-layout
```

Copy in this directory the file *chip.ap* (The path of this file will be indicated by the tutors in the Laboratory).



```
cp /<path>/chip.ap .
```

Now we can edit the layout of the chip with **Graal**, the Alliance hierarchical symbolic layout editor. Give the following command at the command line



```
graal &
```

Click on the *file* section of the menu and then click *open* to load *chip.ap*. What is shown is the top-level view of the symbolic layout of a four-bit microprocessor. Now you can play with the options and some of the tools from the menu. Here is a menu list with some brief description of some of the tool you will need in this exercise.

File: **open:** to load an “.ap” file.

quit: to end the graal session.

Tools: **peek (unpeek):** to show the layers.

Flat (unflat): To eliminate one hierarchical level.

equi (unqeu): to render evident the electrical connections.

real flat: to eliminate the hierarchical structure.

View: **fit:** to fit the layout in the windows.

zoom (in, out)- mooz: To zoom in out.

arrows: to move according to the arrow direction.

layers: to show/hide specified layers.

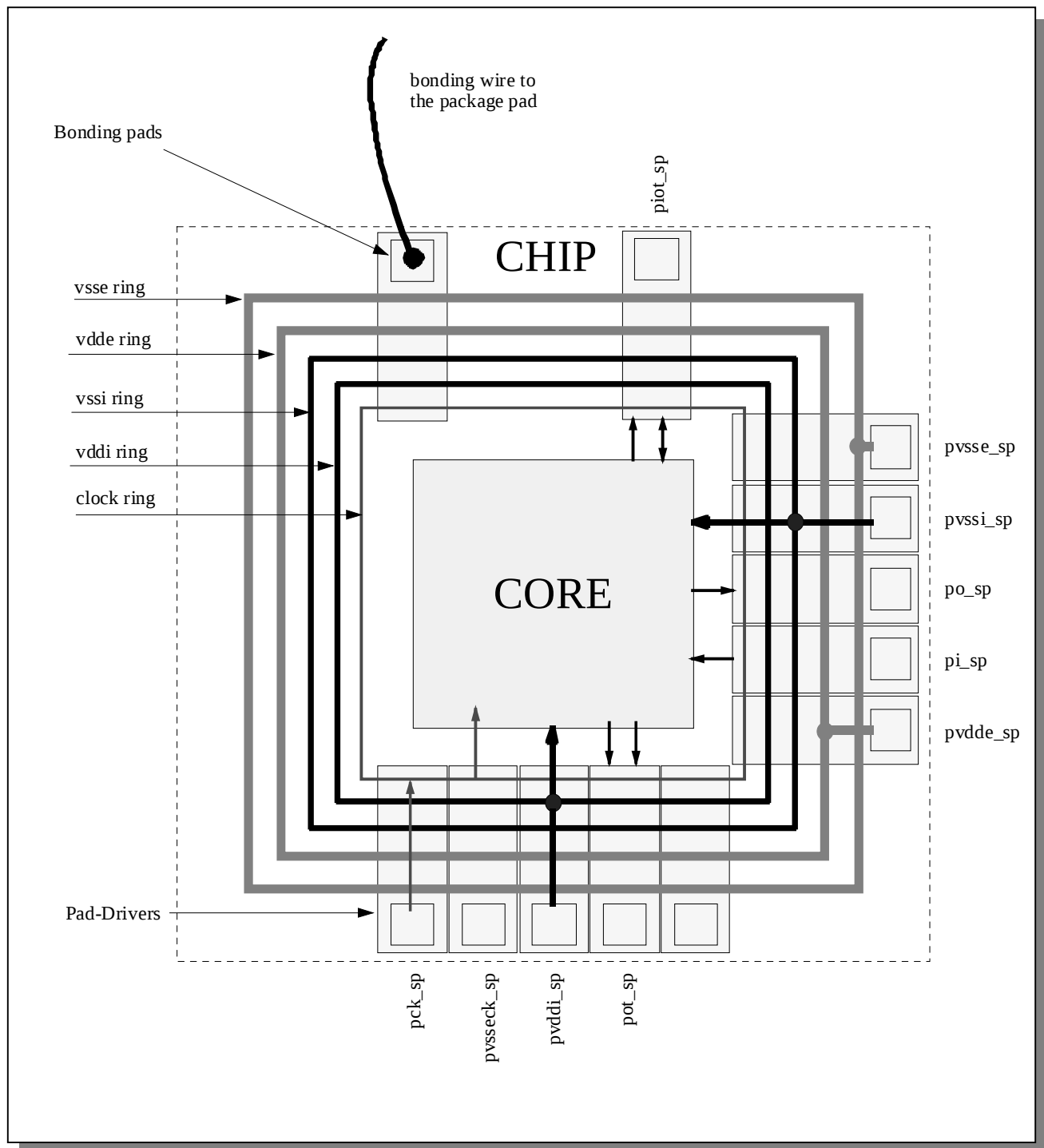
Windows: **Identify:** to identify cells, model, instance name, layers, etc.

For more information about **Graal** menu type



```
man graal
```

By inspecting the layout identify the core and the pads. Compare the pads trying to recognize its functionality by mean of its connections with the metallic rings. To do this see figure 1 and use the functions: *flat*, *zoom-in*, *equi*, *layers*, etc.



Now you can load standard cells from the *Alliance Standard Cell Library*. Copy some physical description files (<filename>.ap) from “/alliance/share/cells/sclib” in to your working directory. For example we can copy a simple *inverter* gate:



```
cp /alliance/share/cells/sclib/n1_y.ap .
```

Copy also from the previous location the following files:

no2_y.ap	two input <i>nor</i> gate
na2_y.ap	two input <i>nand</i> gate
a2_y.ap	two input <i>and</i> gate
o2_y.ap	two input <i>or</i> gate
xr2_y.ap	<i>exclusive-or</i> gate
msx_y.ap	<i>D-flip-flop</i> with complementary outputs.

You can have more information about the previous cells by invoking the corresponding man pages. Just type for example



```
man no2_y
```

Once you have edited the cell with **Graal**, use the “*layer*” tool to inspect the different layers: *diffusions*, *polysilicon gates*, *metals*, *vias*, etc. Identify the P-MOS and N-MOS transistors and how they are connected in order to perform its logic function. Try several standard cells till becoming familiar with them, and **don’t forget** to make extensive use of the Alliance man pages.

