CSE3142 — Integrated Circuit Design

Designing a Complete Integrated Circuit: from VHDL to circuit layout

Practical Experiment 5 **Duration**: two weeks

5.1 About this tutorial

In this practical exercise you will design a complete integrated circuit consisting of two main parts:

- The **core logic**. We will use VHDL tools to synthesize a simple arithmetic unit as an example of a combinational circuit. VHDL offers a more efficient way of circuit specification than the schematic entry.
- A **padframe** containing I/O pads from the selected library.

The following documents will be helpful to clarify any outstanding problems:

- The Prac4 and Prac3 manuals from `http://www.csse.monash.edu.au/~app/CSE3142/Pracs/prac4.pdf`
- Chapter 9: *Generating Padframes* from the manual *Designing ASICs with the ADK Design Kit and Mentor Graphics Tools*.

  `file:/sw/adk/doc/html/adk.9.html`, or

  `file:/sw/adk/doc/pdf/adk.pdf`

5.2 Setup for VHDL synthesis

Modify/check your `$HOME/mgc/mgcrc` to read:

```
# mgcrc   20 09 2002
##  Mentor Graphics  ICFlow 2002.20
#   source /u/mentor/ic.env
   source /sw/mentor/mentor.env
##  ASIC Design Kit  (ADK)
##     and  FPGA advantage (Hds, Modeltech, Spectrum)
  source /sw/adk/adk.env
##  AMS   (eldo and others)
 source /sw/ams/ams.env
setenv ICDES  $HOME/mgc/ICdes
setenv MGC_WD .
```

5.3 Compiling a VHDL description of an arithmetic unit

- Create directories `$ICDES/prac5` , `$ICDES/prac5/src` , `$ICDES/prac5/edif` , `$ICDES/prac5/vhdlout`

- **Preparing a VHDL specification**

  In directory `$ICDES/prac5/src` create the file `uAU.vhd` (u - your initials) with the following VHDL program (I used the name ArU. Remember to modify this name to uAU). This VHDL file describes a simple arithmetic unit that we will synthesize and create an integrated circuit layout.

```
 --                    uAU.vhd
 library ieee ;
 use ieee.std_logic_1164.all ;
 use ieee.numeric_std.all ;
 entity ArU is
   port(
        A  : in   std_logic_vector (3 downto 0) ;
      opc  : in   std_logic_vector (1 downto 0) ;
        B  : out  std_logic_vector (7 downto 0)
       );
 end ArU ;

 architecture  RTL  of  ArU  is
   signal  A8 , B8 : unsigned (7 downto 0) ;
   signal  A4 : unsigned (3 downto 0) ;
 begin
  A8 <= unsigned ("0000" & A) ;
  A4 <= unsigned ( A) ;
  with  opc  select
    B8 <= A8          when "00" ,
          A8 + 1      when "01" ,
          A4*A4       when "10" ,
          A4*(A4 + 1) when  others ;
  B <= std_logic_vector (B8);
 end RTL ;
```

  Note that in the above program arithmetic operations are handled by the package `numeric_std` The source code of the package can be found in `/sw/fpgadv/Hds/hdl_libs/ieee/numeric_std`

  You might like to inspect this package and note that arithmetic operations are performed on the unsigned, signed and integer types. Conversion between types is handled by suitable conversion functions.

We will be using two tools: one for **compilation** , debugging and simulation, the other for the **synthesis**. These tools can be invoked both from relevant GUIs and from the command line windows. Here we will be using a **command line approach**.

- **Setting up the VHDL compiler environment**

  From the directory `$ICDES/prac5` invoke:

  > `vlib $ICDES/prac5/work`
  > to create a VHDL "library" directory `work`. Note a special file `_info` in `work`.

  > `vmap work $ICDES/prac5/work`
  > to create/modify a `modelsim.ini` file in your `$ICDES/prac5` directory. Examine this file and note the presence of the above mapping.

- **Compilation**

  From the directory `$ICDES/prac5` invoke:

  ```
  vcom -source -work work $ICDES/prac5/src/uAU.vhd
  ```

  This will compile your VHDL file into the predefined `work` library. If there are any errors (typos ?), fix them.

  Examine the contents of the `work` directory.

  If compilation is correct we can assume that simulation at this stage is not required due to the simplicity of VHDL specification.

5.4 Synthesizing the core logic

The synthesis will be performed by a LeonardoSpectrum synthesizer. We start with preparation of the following `synth` script in the `$ICDES/prac5` directory (remember to modify the name 'ArU' accordingly):

```
## synth   ADK synthesis script
set vhdl_write_component_package FALSE
set vhdl_write_use_packages { library ieee, adk ;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all ;
use adk.all ; }

set edifout_power_ground_style_is_net TRUE
load_library tsmc035_typ

analyze src/ArU.vhd -format vhdl -work work
elaborate ArU -architecture rtl -work work
```

```
ungroup -all -hierarchy
optimize -ta tsmc035_typ -effort standard -macro -area
report_area -cell synthRep.txt

write ./edif/ArU.edf -format edif
write ./vhdlout/ArU.vhd -format vhdl
```

- From the directory `$ICDES/prac5` invoke the Leonardo synthesizer:

  ```
  spectrum -file synth
  ```

  Assuming that there are no errors a lot of warning will be generated regarding DRC violation. These can be safely ignored. The synthesis report is in the file `synthRep.txt` and should be similar to the following:

```
*****************************************************
Cell: ArU    View: RTL    Library: work
*****************************************************
    Cell      Library        References     Total Area

    ao32      tsmc035_typ     1 x      2      2 gates
   aoi21      tsmc035_typ     6 x      1      7 gates
   aoi22      tsmc035_typ     1 x      1      1 gates
   inv01      tsmc035_typ    11 x      1      8 gates
   mux21      tsmc035_typ     5 x      2      9 gates
  nand02      tsmc035_typ    11 x      1     11 gates
  nand03      tsmc035_typ     2 x      1      2 gates
   nor02      tsmc035_typ     9 x      1      9 gates
   oai21      tsmc035_typ     7 x      1      9 gates
   oai22      tsmc035_typ     3 x      1      4 gates
   xnor2      tsmc035_typ    14 x      2     27 gates
    xor2      tsmc035_typ     1 x      2      2 gates

Number of ports :                      14
Number of nets :                       77
Number of instances :                  71
Number of references to this view :     0

Total accumulated area :
 Number of gates :                     93
 Number of accumulated instances :     71
```

- The specification of the synthesized circuit is now given in the **EDIF format** by the file `$ICDES/prac5/edif/ArU.edf`. It is a text file that you should examine.

The EDIF file needs to be converted into a Mentor Graphics schematic which is given in the **EDDM format**. The conversion function $ADK/bin/edif2eddm refers to the library of parts in a requested technology through a configuration file $ADK/lib/adk_enread.cfg.

In order to convert EDIF to EDDM, from the directory `$ICDES/prac5/edif` invoke:

```
edif2eddm ArU.edf
```

This is a very "active" script. It creates a directory `$ICDES/prac5/edif/work` and in this directory a Mentor Graphics component **ArU**. A schematic of this component will have a name **RTL**.

The created schematic and an associated symbol specify the **core logic** of our integrated circuit.

- **Examining the synthesized schematics.**

  From the directory `$ICDES/prac5/edif/work` invoke the Design Architect:

  ```
  adk_daic &
  ```

  In the Design Architect window, from the **session_palette**, select Open > Schematic and in the Open Schematic dialog box specify (using the navigator) **ArU** as a component name. Select *Options* and in the Open Schematic Options dialog box chose **RTL**.

  This will open a schematic ArU sheet1. You might like to execute **max wi** and **vie al** to see the complete schematic as in Figure 1. It is a logic gate diagram consisting of

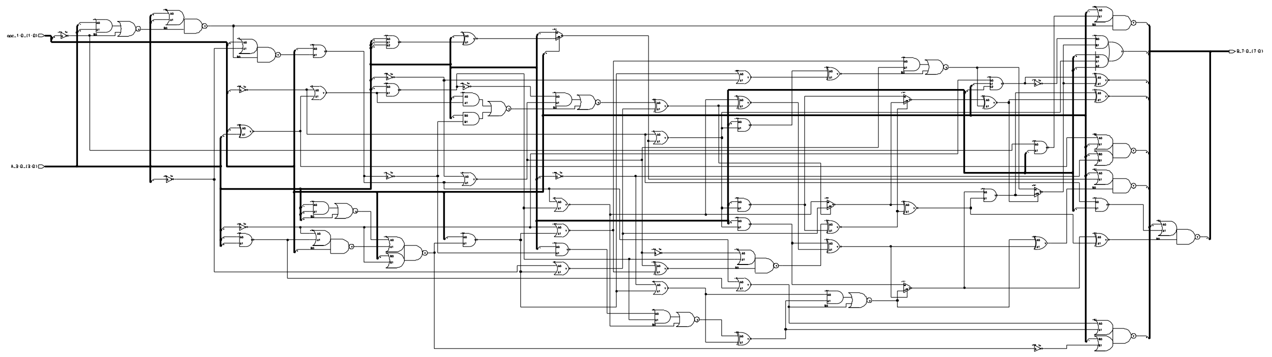

Figure 1: A logic-level schematic of the ArU component.

  components from the standard cell library and to see details you need lots of zooming in.

  For demonstration, I also synthesized a sub-circuit performing the squaring S = A*A. This is included in Figure 2. Can you explain why S(1) is nowhere to be seen?

- **Check & Save** the schematic. You will see lots of warnings related to the technology parameters.

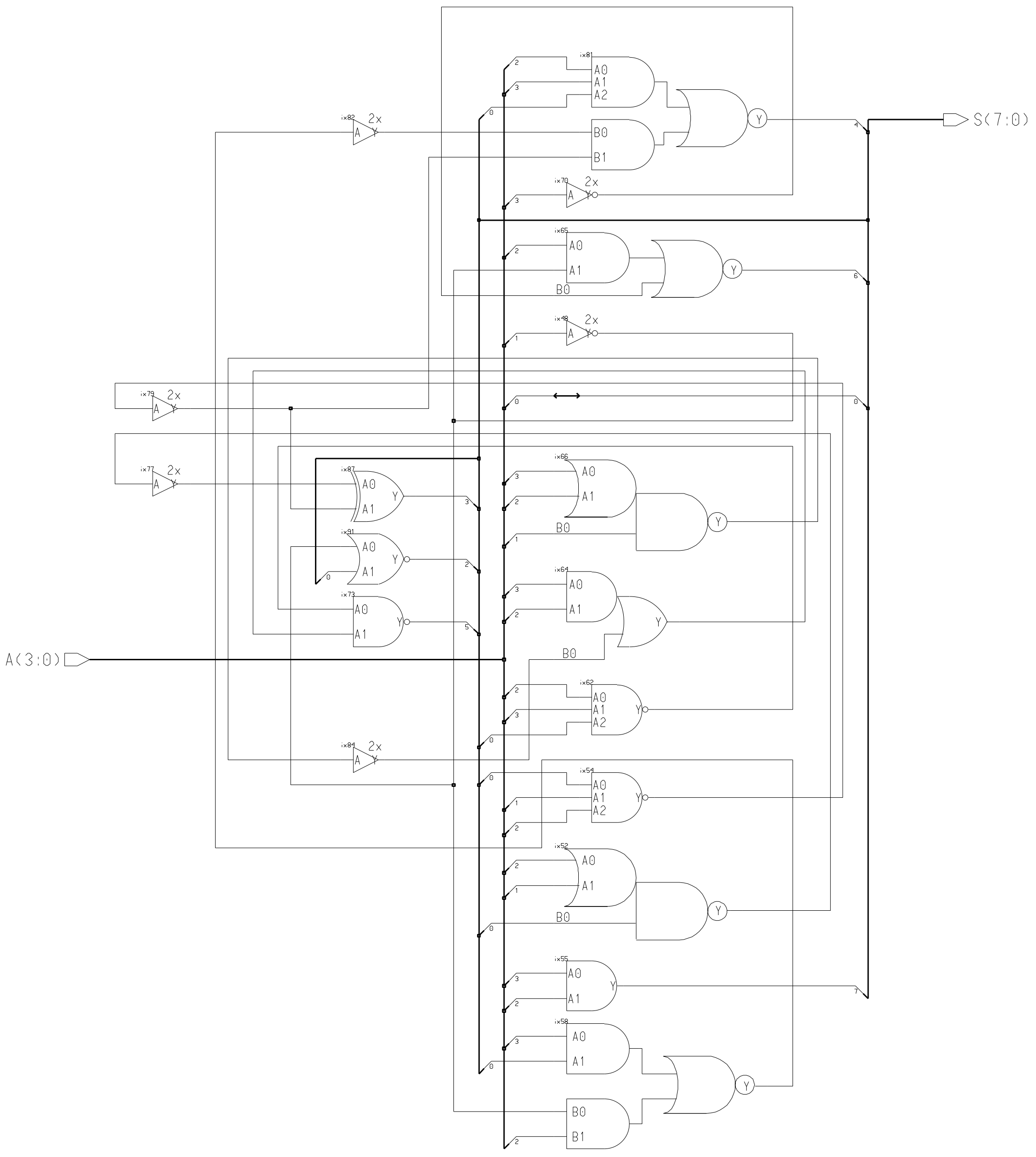

Figure 2: A schematic of a squaring circuit, S = A*A

- **Miscellaneous** > **Generate Symbol**. Generate, check and save the symbol. Close the symbol window.
- **Viewpoint generation**. Form the schematic_edit palette select

  Simulation > TSMC0.35

  This will create viewpoints required in simulation and layout generation.
- You can exit the Design Architect.

5.5 Automatic generation of a layout of the core logic

We use standard cell design methodology to generate layout of the core logic as in the previous tutorial.

- From the `$ICDES/prac5/edif/work` directory invoke

  adk_ic &

- From the Session palette select Create. In the dialog box enter the following (use the Navigator where possible):

  Cell Name : ArU
  Attach Library : $ADK/technology/ic/tsmc035
  Process : $ADK/technology/ic/tsmc035
  Rules File : $ADK/technology/ic/tsmc035.rules
  Connectivity ? : With connectivity
  Eddm Schematic Viewpoint : ArU/layout (the layout creation viewpoint of ArU)
  Logic Loading Options ...
  Logic Loading: Flat

  All other options should be left at default values.

  An empty cell window will now appear having the name ArU. In the top left bar of the window it should read Process: tsmc035(-R).

- **Floorplanning**

  From the IC Palettes go to the Place & Route palette and select Autofp.

  In the Autofloorplan Options dialog box, allow all options to default simply by clicking on OK.

  Use **View** > **All** from the pull-down menu to see the automatically generated floor plan.

  You will see a series of boxes enclosed by solid green bars along each edge. The boxes indicate the rows into which cells will be organized. The solid bars indicate edges of the cells along which physical ports will be placed.

  If you have only one row, something somewhere went wrong. Find and fix it.

- **Standard Cell Placement**

  In the Place & Route palette go to the Autoplc section and choose StdCel. Accept the default options in the dialog box.

  You should see individual cells placed in the floorplan boxes.

  Cell locations are determined by their interconnectivity. Cells which share connections are placed near one another.

  You may wish to experiment with the results obtained thus far by selecting different Autofloorplan and StdCel Autoplace options.

- **Port Placement**

  From the Autoplc section of the Place & Route palette select Ports.

  You can allow the options to default for now, but you may wish to experiment with them later also. You will see lightly shaded areas along the port bars at the edge of the layout.

  From the point of view of the structure of the complete integrated circuit it **very** important to know the **relative location of the ports**. Select/unselect (F2 key) connection to every port and obtain the relative port positions which might be similar to the following:

	B(6) B(7) B(2) opc(0))	
A(2)		B(1)
A(1)		VDD
A(3)		B(0)
GND		A(0)
	B(5) B(3) B(4) opc(1)	

  Obtain **your** version of the above table.

- **Routing**

  Now, from the Autorou section of the Place & Route palette select All, and accept the defaults in the dialog box.

  This should route all connections in our cell.

  You will see the cell rows move apart and additional metal runs appear in the channels between rows.

  Identify which metal layers are used for horizontal and vertical connections.

- **Compacting** The autorouter's objective is to complete as many connections in the design as is possible, with less regard for wasted area. For this reason, a layout **compactor** is available which "squashes" things together where there is any little bit of unused space.

  To take advantage of this tool, from the PR Edit section of the Place & Route palette select Compct.

  We want to compact in both the horizontal and vertical dimensions, so do this twice: once **down** and then again in the **left** direction.

  Do not do this more than once along a given axis. Not only does it rarely result in improvement, it can lead to layout errors in some versions of the compactor.

- Verify that after routing and compacting the relative port positions have not changed.

- Save the layout by selecting

  File > **Cell** > **Save Cell** > **Current context**

You may save the layout and exit the ICgraph session at any time. To re-load the layout later, from the Session palette choose Open.

If you wish to make changes to the cell, you must also select

File > Cell > Reserve > Current context

- To remove the green remnants of the floorplan (layer 4097, "row") can be removed by removing type in

  set vi l 1-200 4104-4113

  This completes the automatic layout of the core logic which might be similar to that in Figure 3.

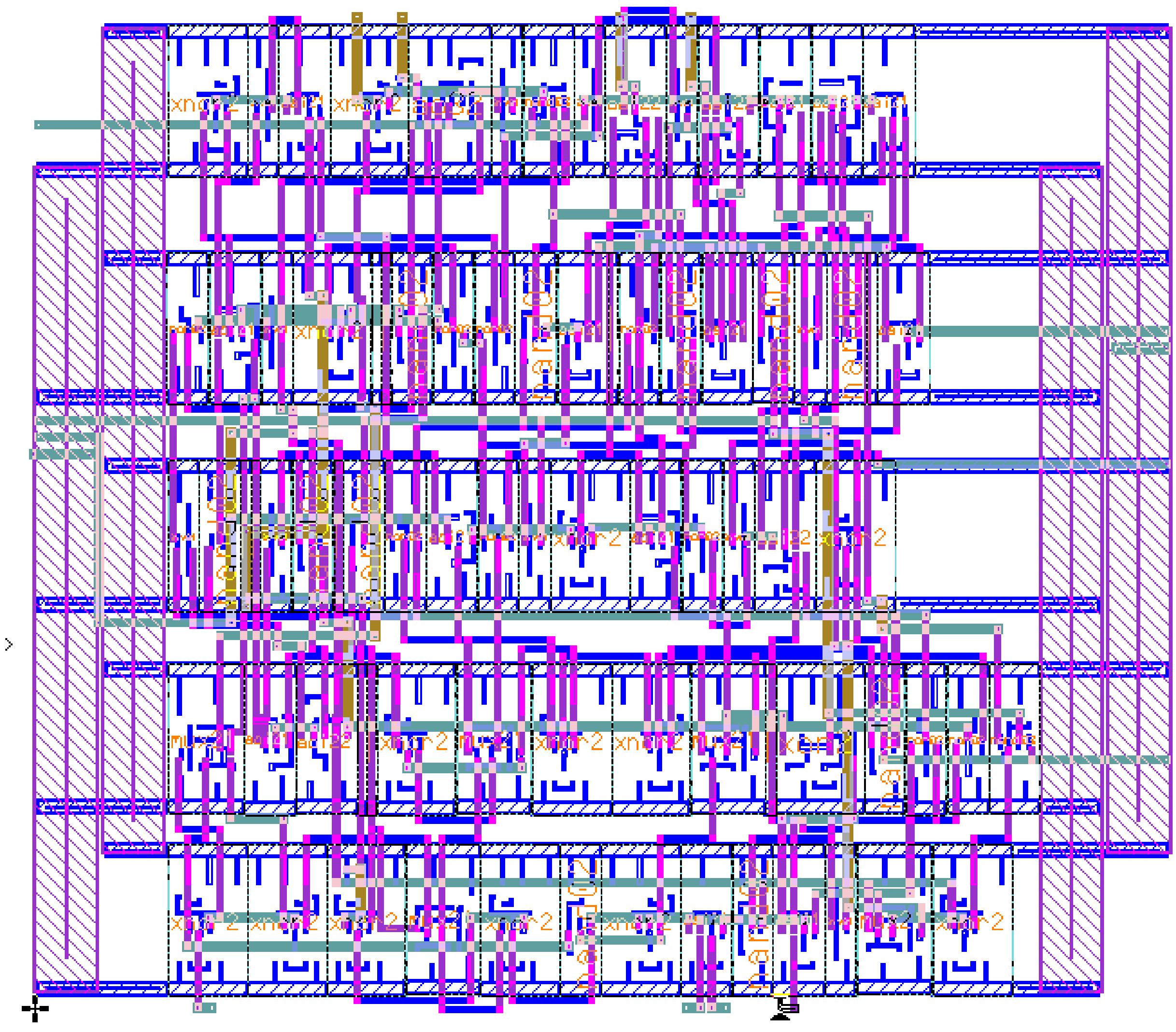

Figure 3: The automatically generated layout of the core logic

5.6 Schematic of the complete chip

- From the directory `$ICDES/prac5/edif/work` invoke the Design Architect:

  ```
  adk_daic &
  ```

- **Adding a property to the ArU cell symbol**

  The core logic needs one property to be attached to its symbol. In order to do that from the session_palette open the **ArU** symbol, and select the symbol body. Then from the pull-down menu select

  Add > **Properties...**

  and in the Add Multiple Properties dialog box specify:

Property Name	phy_comp
Property Value	ArU

  and place the text inside the symbol. This property informs the tools where to find the **layout** associated with the component ArU (with respect to $MGC_WD). The layout has been generated in sec. 5.5.

- Check and save the symbol.

Now we have to create a new component/schematic consisting of the **core logic** and the **I/O pads** . This is a labourious task. You can speed things up by creating a suitable AMPLE script(s). For details, consult
`/sw/mentor/mentor/shared/pdfdocs/ample_user.pdf`

- From the session_pallete

  Open > **Schematic**

  and in the Open Sheet dialog box specify the new component, say **PuAU** (I will be using the name **PArU**). The new schematic sheet will be opened.

- You will need 6 input pads (In), 8 output pads (out) and 2 power pads (GND and VDD) as in Figure 4. The pads are specific for the given technology and must agree with the technology of the design. For the TSMC0.35 technology schematic symbols of I/O pads exist in `$ADK/lib/pads/tsmc035`. You can navigate to the pad library starting from the schematic_edit palette and

  Library > Standard Cells > Pads > TSMC 0.35

  to reach the ADK TSMC 0.35 Pads palette.

  Place one input pad **PadInC** (called In in the palette menu) to obtain the dimensions of the pad, say, $w \times h$, delete the pad and execute the AMPLE script **PArUa.do** similar to the following:

```
// dof PArUa.do
{ local n, m , w, h ;
 w=2.75; h=1.75;
 $set_active_symbol("$ADK/lib/pads/tsmc035/PadInC", "", [], "");
  for (n = 0; n<= 5; n = n + 1)
  {
    m = h*n ;
    $place_active_symbol([0, w+m], @perproperty, [], @noflip, 0);
  }
}
 $$add_instance("$HOME/mgc/ICdes/prac5/edif/work/ArU","",[4, 4.5],
                     @perproperty, void, [], @noflip, 0);
 $view_all();
 $unselect_all();
```

That should place all six input pads in one column, and in addition the cell representing the core logic, **ArU**.

- In a similar way place the output and power pads. Next, wire the core logic and pads together for the complete circuit to look as the one in Figure 4
- At this stage we have not yet allocated the ports of the core logic to the pins of the padframe. The properties ‘PINXX’ of the I/O pads must be changed so that ‘XX’ is replaced with the pin number of the padframe.

  In order to allocate pins of the padframe to ports of the core logic we prepare the following diagram:

	15	···	6	
16		B(6) B(7) B(2) opc(0)		5
	A(2)		B(1)	⋮
⋮	A(1)	CORE	VDD	1
	A(3)		B(0)	40
	GND		A(0)	⋮
25		B(5) B(3) B(4) opc(1)		36
	26	···	35	

Based on the above diagram I arrived at the following pin–port allocation:

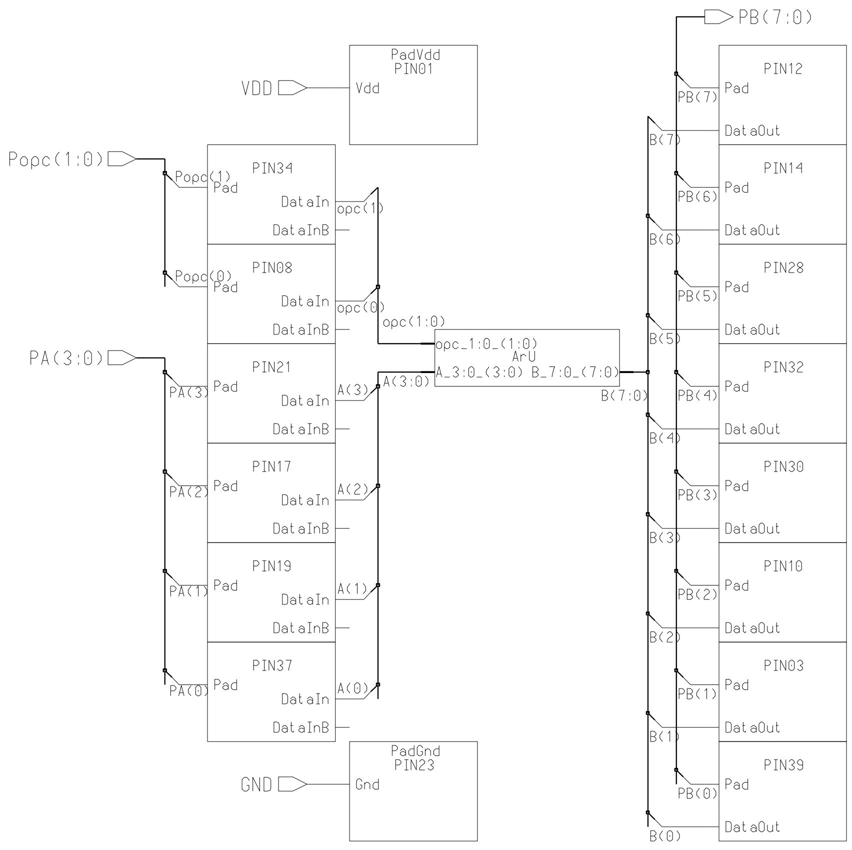

Figure 4: A schematic of the integrated circuit with I/O pads.

Port	Pin
VDD	1
opc(1)	34
opc(0)	8
A(3)	21
A(2)	17
A(1)	19
A(0)	37
GND	23

Port	Pin
B(7)	12
B(6)	14
B(5)	28
B(4)	32
B(3)	30
B(2)	10
B(1)	3
B(0)	39

Your pin allocation table will be similar (but NOT identical!).

- Change the property ‘PINXX’ using the shift+F7 key to replace the ‘XX’ with the pin number as in the above table. Performing this task from a suitable script is highly advisable. Such a script can be build around the following instructions:

```
for (i=0; i < 6; i=i+1)
 {
  $select_area([[x, y], [x, y]]);
  $change_property_value("INST", $format("PIN%02d",pn[n]), @string);
  y = y+h;
  n=n+1;
 }
```

- Check CAREFULLY, fix ALL the errors and save the schematic.
- **Generate symbol** for the complete chip. Check and save it. Close the symbol window.
- **Creation of viewpoints**

  Finally, for the PArU component representing the complete chip create viewpoints needed for simulation and layout creation. To do this, from the **schematic_edit** palette select

  Simulation > **TSMC0.35**

  This will specify the TSMC0.35 technology and create the required viewpoints.

5.7 Layout of the complete chip

Creation of the layout of the complete chip follows the Schematic-Driven-Layout (**SDL**) methodology.

- From the `$ICDES/prac5/edif/work` directory invoke

  adk_ic &

- From the Session palette select Create. In the dialog box enter the following (use the Navigator where possible):

  Cell Name : PArU
  Attach Library : $ADK/technology/ic/tsmc035
  Process : $ADK/technology/ic/tsmc035
  Rules File : $ADK/technology/ic/tsmc035.rules
  Connectivity ? : With connectivity
  Eddm Schematic Viewpoint : PArU/layout (the layout creation viewpoint of PArU)
  Logic Loading Options ...
  Logic Loading: Flat

  All other options should be left at default values.

  An empty cell window will now appear having the name PArU.

- (almost exact quote from ADK.pdf)

  Go to the **ADK Edit** palette menu and click on **Open** to open the logic source window. You should see your schematic in a new window.

- Select the core logic (if you have more than one piece, select them all). Then click on **Inst** on the **DLA Logic palette** to place these cells into your layout window. You may need to type in the **vie al** command or use the Shift-F8 key to see the layout of the core cell(s).

  Remember, you must have previously designed these cells and they must have the name of the **phy_comp** property on your symbol.

  From the pull-down menu select **Setup** > **SDL** and verify that the Search Path looks like this:

  $ADK/technology/ic/tsmc035_via $PROJECT/cells .

  The last '.' is important for it includes the current, or working directory. You should add any other paths you wish to search for your cells.

- **Padframe Generation**:

  Once you have placed the core logic block(s), you are ready to generate your padframe. Do not place the pads, yourself. From the pull-down menu select

  ADK > **Generate Padframe** > **0.35 micron tinychip**

  This will generate the padframe.

  Make life easy on yourself. NEVER EDIT OR RELOCATE PAD CELLS.

- View the entire cell to see the padframe and your core cell(s). You should see the entire padframe with corner pads, the pads you asked for and the spacer pads for all the empty spaces. You will also see overflows showing which ports get wired to which pins.

- At this point you might notice that your core logic should be rotated, flipped or moved to make routing easier. Do that, now, but be careful not to move the pads.

- After finalizing the placement of your core logic, you can route the pads to the core. You can do this manually or automatically.

- To autoroute, click on P&R on the ADK Edit palette menu to bring up the Place & Route palette menu.

  Then click on All under Autorou . On the prompt-bar that appears click **Options** and **unselect Expand Channels** from the menu. Click OK on the menu and prompt-bar to autoroute the pads.

- Now take a look at the final layout and fix any problems. For example, you might not like the width or placement of the power busses. You can edit these routes (or opt to manually route them, initially) until you are satisfied with the final result.

The final layout of the integrated circuit you designed is presented in Figure 5.

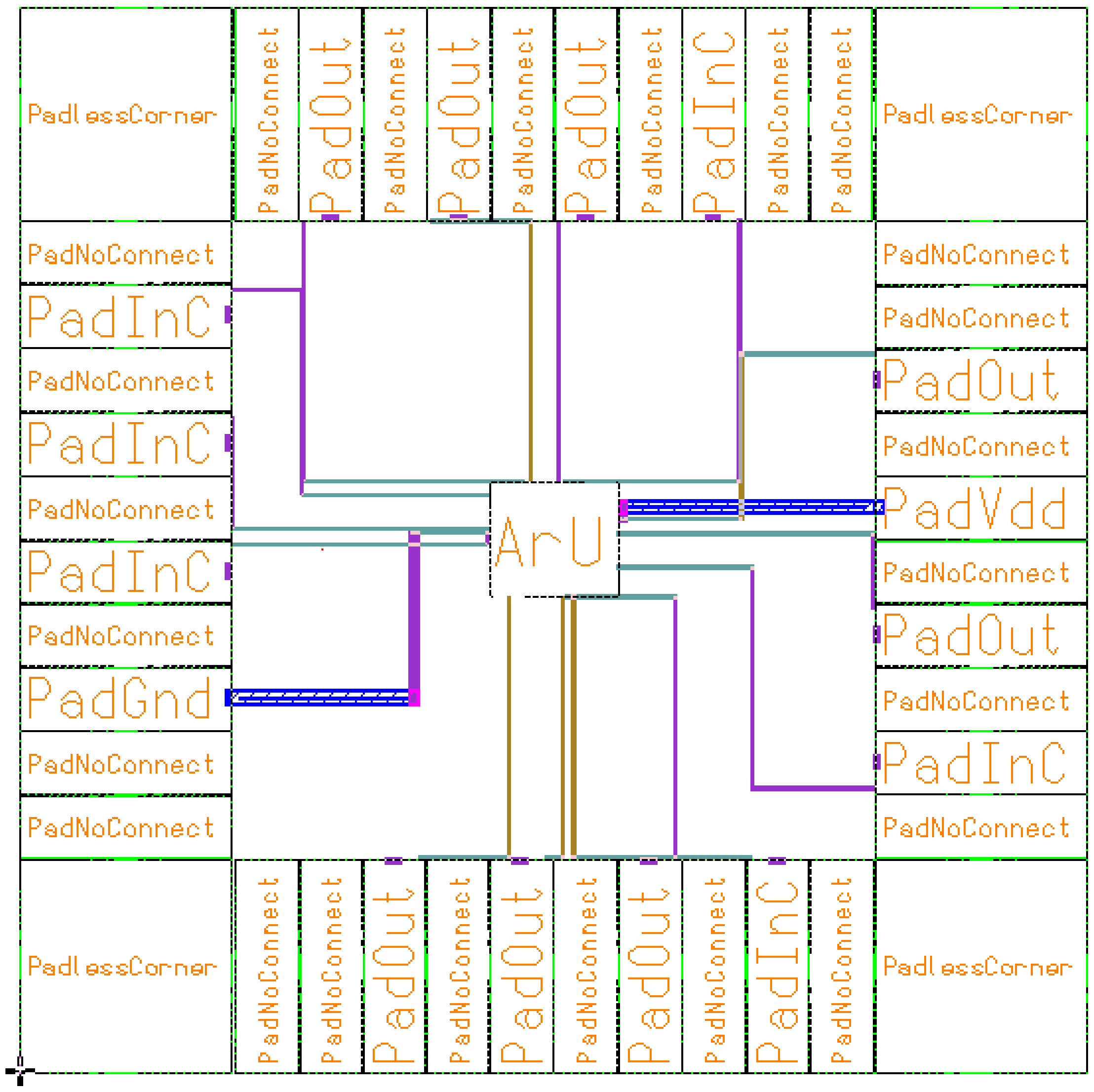

Figure 5: Layout of the complete integrated circuit

5.8 Post-Layout Simulation with QuickSimII

- If time permits and to get higher mark perform the layout-versus-schematic (LVS) checking.

  The design rule checking will not work with I/O pads.

- Finally, perform the post-layout simulation with QuickSimII as described in Chapter 11 of the “adk.pdf”.

5.9 Demonstration and Report

- Demonstrate your working circuits to your tutor.
- Write a brief report in which you include a short description of your activities and the results obtained. Include schematics, simulation results, layouts, stick diagrams and relevant explanations.