CSE3142 — Integrated Circuit Design

Standard-Cell Design Methodology

Practical Experiment 4 **Duration**: two weeks

4.1 About this tutorial

A typical Integrated Circuit design flow using the **Standard-Cell methodology** involves three major steps, namely:

- Front-End Design,
- Physical Layout Design,
- Layout Verification

Front-End Design addresses the problem of circuit specification. Two basic groups of methods are available, namely, a **schematic entry**, and a **VHDL description** of the design.

In this practical exercise we will:

- create and verify a hierarchical schematic of an array combinational circuit using components from a library of CMOS standard cells available in the ASIC Design Kit (ADK).
- convert the schematic of the circuit into a **layout** of an integrated circuit using a standard cell methodology,
- verify the geometric correctness of the layout,
- perform the Layout-Versus-Schematic (LVS) check,

We will use the

- adk_daic for the schematic entry, generation of viewpoints for simulation and layout, and for analog simulation,
- quicksim for functional simulation,
- adk_ic to perform the standard cell placement and routing, to verify geometric correctness of the layout (DRC), and to perform the Layout-Versus-Schematic (LVS) check.

Setting adjustment

Before you start, add to `$HOME/mgc/mgcrc` the following definition:

```
setenv LEGACY_MGC_HOME /sw/mentor/mentor
```

This is needed to run the QuickSim. Remember to open a new shell window.

The design

The design we will be creating is a 4-bit division-by-3 circuit, similar in its structure to a 4-bit adder. A 4-bit circuit **div3b4** is created from a 1-bit building blocks **div3b1**. Details of the 1-bit cell are given in the appendix.
Before you start the schematic entry, you have to derive logic equations for three signals generated by the 1-bit cell, namely, c(1), c(0), s.

4.2 Schematic entry of the 1-bit division-by-3 component, div3b1

The 1-bit division-by-3 circuit is a three-input (c(1:0), a), three-output (s, d1, d2) combinational circuit. Once you have the logic equations for three sub-circuits ready you can create the schematic of the **div3b1** component.

- Create a directory `$ICDES/prac4` and
  `setenv MGC_WD $ICDES/prac4`
  From this directory invoke the design architect entering `adk_daic &`.

- From the `session_palette` Open > Schematic, or select the item
  File > Open > Schematic...

  from the menu bar and specify:

  Component Name: `div3b1`

  in the dialog box. Click OK. This will open an empty schematic sheet `div3b1`. Zoom out $\times 2$.

- The sub-circuit generating the output s can be similar to that given in Figure 1. Note that the carry-in signals c(1:0) form a 2-bit bus.

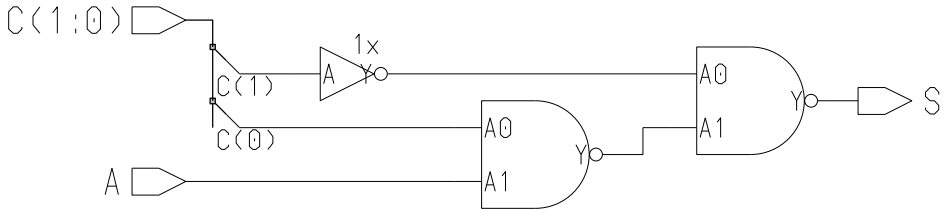


Figure 1: The output s of the 1-bit division-by-three component.

- In order to select cells from the library of standard cells, , select from the schematic_edit palette Library and then from the ic_library palette Standard Cells.

  It shows the Cell_Library palette menu. Select the Basic Logic Gates entry and find the nand02 gate. Place it in the schematic. This is equivalent to the following AMPLE function:

  ```
  $add_instance("$ADK/parts/nand02","",[1,1]);
  ```

- Complete the first part of the schematic to look as the one in Figure 1. Some helpful points are as follows:

  - Inverters can be accessed selecting Cell_Library > Inverters and Buffers.
  - The portin and portout symbols are available either from the ic_library, or from the Cell_Library or from the schematic_edit palette.
  - **Place** all components from the `$ADK/parts` library and the portin and portout symbols. Use the **copy** and **move** operation, suitable mouse **strokes** and **function keys** to facilitate the placement (and subsequent wiring up) of components.
  - **Wire up** the components. Invoke Add > Wire or Add > Bus/Bundle from the schematic_edit palette.
  - You wire between two points by clicking once on the starting point and twice on the ending point. A single click in between forms a bend in the wire. Click on the Cancel button when you have finished wiring. Unselect all by using the **F2** function key.
  - **Bus rippers 1**. Note that when you connect a single wire to a bus it is important to have the bus name assigned first. This is explained in the next item.
  - Assign the appropriate **names** to the input/output ports. These are initially assigned the **NET** property values. Select three **NET** property values using the **F1** function key. Next, select the item Text to go to the schematic_text palette from which you select Edit > Change Value.

    A dialog box will appear at the bottom of the screen. The value you are currently changing is highlighted in the schematic. Simply type in the required value (e.g. a) over the old one (NET) and hit the return key. In the schematic, the next value will be highlighted.
  - **Bus rippers 2**. Once you have a name like C(1:0) assigned to a pin, hence to a bus, you can connect single wires to a bus. Each time you will be prompted in the dialog box Choose Bus Bit for a bit number , e.g. 1 . This will create a bus ripper.
  - When finished this part, you might like to **Check & Save** the design. There should be no errors. You will get warnings about missing properties. These properties are set up in the viewpoints.

- Derive appropriate logic equations from the truth table given in the appendix.

  Complete the **div3b1** schematic by adding the d(1) and the d(0) circuits. Use a bus for d(1:0).

- **Check & Save** a free-of-errors schematic of the **div3b1** component.

- **Generate**, and **Check & Save** the **div3b1** symbol:

  From the top menu bar, select

  Miscellaneous > **Generate Symbol**.

  A dialog box will appear that gives a variety of options for the automatic creation of a symbol which encapsulates your logic diagram. The only option you might need to change is the Once generated... column where you might like to click on Save and Edit. If you wish to replace an existing symbol for this component, then you will also want to say Yes to Replace existing?.

- Add the name **div3b1** to the symbol box. Select

  Add > **Text**.

  Type in the name **div3b1** in the dialog box and place the text in the symbol.

- **Check & save** the symbol.

4.3 Digital simulation of the 1-bit component with QuickSim II

It is a very good idea to incrementally simulate and debug the design when we build it bottom-up.
Prior to simulation we need to specify the technology and create the necessary viewpoints. To do this, from the **schematic_edit** palette select

Simulation > **TSMC0.35**

This will specify the TSMC0.35 technology and create a number of **viewpoints**. We need the **tsmc035** viewpoint for use with the digital QuickSim II Simulator.
At this stage you might like to exit `adk_daic`.

- In the directory **$ICDES/prac4** define

  ```
  setenv MGC_WD $cwd
  ```

  and invoke the digital simulator, QuickSim II with an appropriate viewpoint

  adk_qs div3b1/tsmc035 &

  This will open the QuickSim II window with lots of errors messages in generated in the Info Messages window. This is due to the technology parameter being setup somewhere incorrectly.

- To fix the problem, click on the DESIGN CHG button in the Setup palette and select ADD PROPERTY icon from the Design Changes palette. It brings up a dialog box in which you have to add:

  New Property Name **technology**

  Property Value **tsmc035**

  It should clear all messages from the Info Messages window.

- Go back to the Setup palette and open your schematic by clicking on the **OPEN SHEET** icon. You may want to re-size the schematic window to make things more easily visible.

- Specify the signals to be traced. One way of doing it is to select signals in the schematic window in the order you would like them to be arranged in the trace window, e.g., c, a, s, d.

  Click on the **TRACE** button in the Setup palette. After the Trace window appears, de-select all signals (the F2 key).

- Specify the input signals (forces). One way of doing so is as follows:

  – Click on the **STIMULUS** button in the Setup palette and use the **PATTERN GENERATOR** to specify input waveforms for C. Refer to Figure 2 for inspiration.

  – Use the **ADD CLOCK** to specify input waveforms for A.

  Alternatively you can prepare a suitable simulation script.

An example of simulation results is given in Figure 2. Note that the output signals need

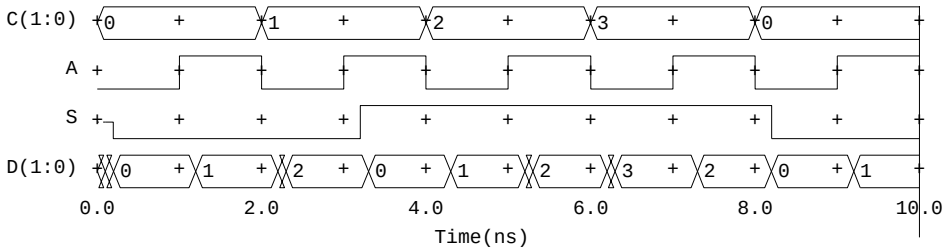


Figure 2: Results of simulation with QuickSim II

approximately 0.3ns to stabilise.

4.4 The 4-bit division-by-3 component

Once the 1-bit component works correctly, you can create a schematic for a 4-bit division-by-3 component, **div3b4**.

- From the directory **$ICDES/prac4** invoke the Design Architect:

  adk_daic &

  and open a new sheet with the component name **div3b4**.

- Use the **div3b1** component symbol as a building block for the **div3b4** schematic. You can instantiate the 1-bit component selecting from the schematic_edit palette

  Add $>$ **Instance**

- Wire the components up to obtain a schematic similar to that in Figure 3.

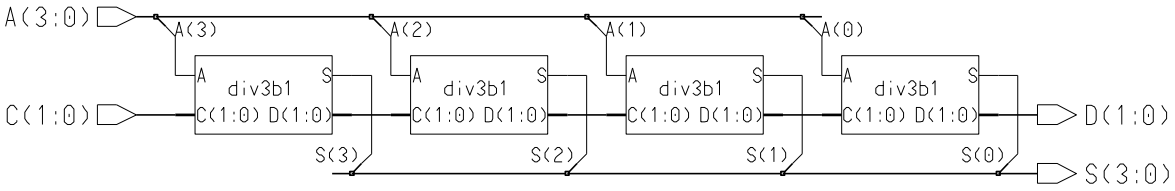


Figure 3: Schematic of the 4-bit division-by-3 component, **div3b4**

- Check, debug and save the **div3b4** schematic.
- Create a symbol for the **div3b4** component. Check and save the symbol.
- In order to specify the technology and create viewpoints for simulation and layout generation, select from the **schematic_edit** palette

  Simulation $>$ **TSMC0.35**

- You can exit the **Design Architect**.

4.5 Functional Simulation of the 4-bit component with QuickSim

Follow the steps describe in sec. 4.3 and simulate the 4-bit division-by-3 component, div3b4 with the tsms035 viewpoint. An example of simulation results is given in Figure 4.
There are 16 different values of the input $A(3:0)$ to be tested and three different values of the input carry $C(1:0)$. The input-output values are related by the following equation:

$$\frac{16 \cdot C + A}{3} = S + \frac{D}{3}$$

(see Appendix)

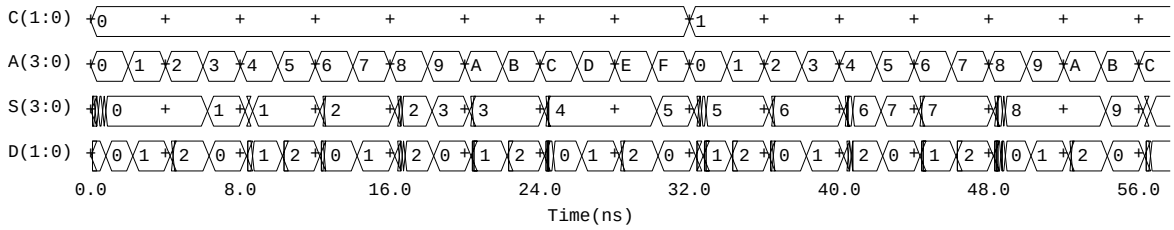


Figure 4: Simulation waveforms for the 4-bit division-by-3 component

4.6 Automatic Layout Generation

In this section you will be exposed to the IC Station tools oriented towards the Standard-Cell design methodology.

- From the `$ICDES/prac4` directory invoke

  adk_ic &

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

  Cell Name : div3b4
  Angle Mode : Ninety
  select button: With connectivity
  Attach Library : $ADK/technology/ic/tsmc035
  Process : $ADK/technology/ic/tsmc035
  Rules File : $ADK/technology/ic/tsmc035.rules
  EDDM Schematic Viewpoint : div3b4/layout
  Logic Loading Options
  Logic Loading: Flat (IMPORTANT!)

  All other options should be left at default values.

  A cell window will now appear having the name div3b4. 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.

- **Routing**

  At this point, it is assumed that you have arrived at a satisfactory initial placement for all cells of the layout.

  Prior to autorouting the interconnections, you may wish to observe a "rats nest" view of the signals connecting the various cells. This is sometimes useful for determining sources of routing congestion.

  – To observe a rats nest of signal connections, from the pull-down menu select

  Connectivity > **Net** > **Restructure** > **All Signals**

  It may look a little messy, but keep in mind that it does not change the layout whatsoever.

  – 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 the **div3b4** design.

  You will see the cell rows move apart and additional metal runs appear in the channels between rows. Note that one metal layer is used largely for horizontal runs and the other for vertical runs.

  – 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.

- The green remnants of the floorplan can be removed by removing the layer 4097, called "row". One way of doing it is to type in

  set vi l 1-200

- 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**

This completes the automatic layout of the div3b4 cell which might be similar to that in Figure 5.
In the next section we verify the created layout.

4.7 Manual inspection of the cell layout

Prior to formal verification of the layout, we will manually inspect it and compare with the schematic.

- **Load the cell schematic**

  From the IC Palettes select ADK Edit and then from the section SDL of the ADK Edit palette choose Open . A window with the schematic of the cell should appear.

  Arrange the layout and schematics window to see most of both.

  You might like to File > Open Down a selected **div3b1** component to go to the gate level.

- Note that objects in the schematic and layout windows are inter-related. Select/de-select components in the schematic window and observe the respective selections in the layout window.

  Do the same for nets and pins. It is convenient first to select

  Setup > **IC**

  and set Port/Pin Name Display under Display Setting to On and click OK . Now you will be able to see input and output pin labels as well as the VDD and GND labels (you may have to zoom in). You can also determine pin labels at wider zoom scales by simply clicking on them and observing the name on the bottom message bar.

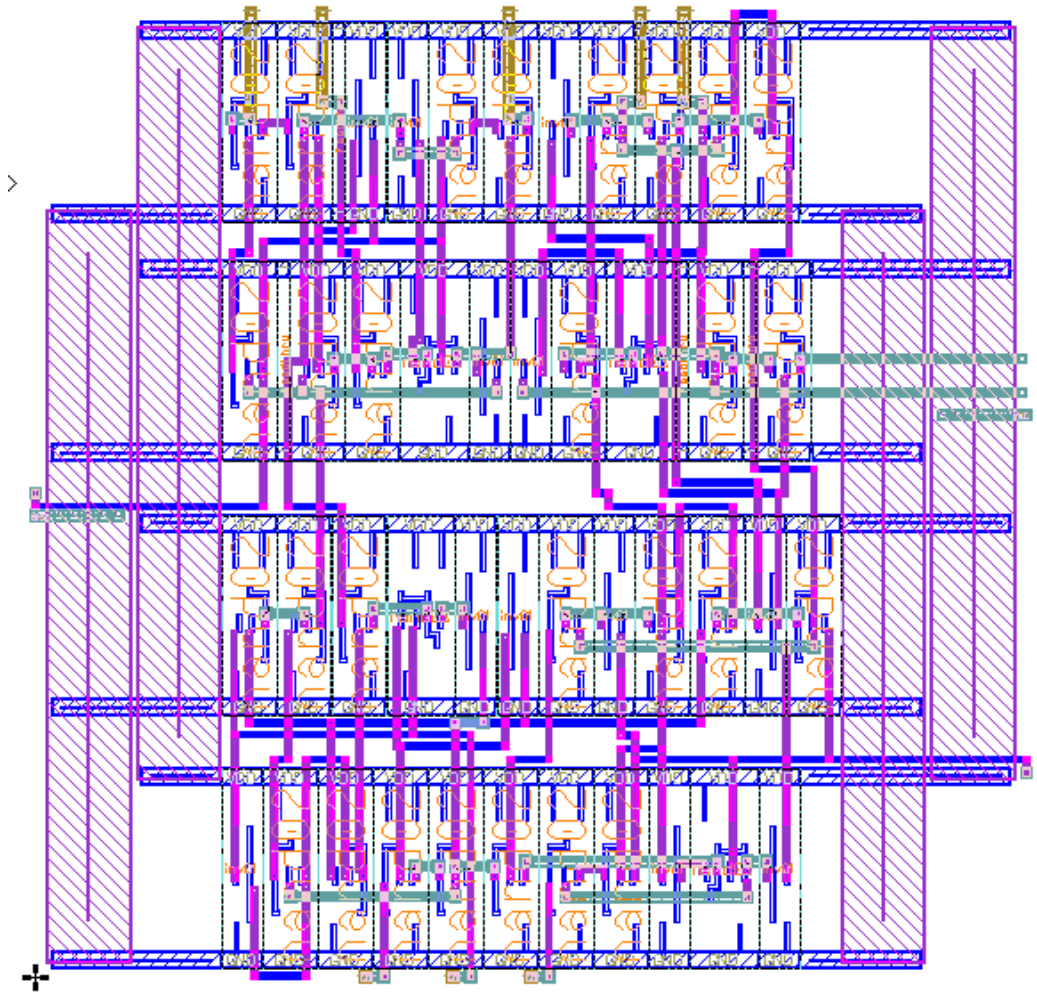


Figure 5: The automatically generated layout of the 4-bit division-by-3 component

- It might be helpful to bring up the layer palette. To do this, from the pull-down menu associated with the layout window select

  Other > **Layers** > **Show Layer Palette**

  and in the dialog box enter: 46-51, 61-62, or just type in

  sho la p 46-51 61-62

  The layout palette with the layers used in interconnections should appear.

- In a schematic window select a single NAND gate, and zoom in its layout in the layout window. In order to see the layout of the standard cell we have to "peek through the hierarchy". In order to do that, from the pull-down menu select:

  Context > **Hierarchy** > **Peek**

  and input 999 into the dialog box. Enter F2 key to de-select.

You should see now the full layout of the NAND standard gate. Try to zoom in to see individual transistors. Two pairs of transistors should be easy to identify. Try to make only selected layers visible toggling in the layer palette with the middle mouse button.

- At the conclusion of this part un-peek and un-select the layout, and close the schematic (from the SDL section of the ADK Edit palette select Logic > Close).

4.8 Layout verification

Although the automatic tools created the layout in the Correct by Construction mode, it is always a good idea to verify the layout for correctness in terms of both layout design rules and connectivity to insure consistency between the various tools used. This establishes a system of "checks and balances" that increases our overall confidence in the design. We will first check for layout design rule errors using ICrules, then verify the layout by double checking it against the transistor level representation of the circuit schematic.

- **DRC**: From the ADK Edit select the Drc item in the Verification section. A dialog box will appear at the lower left of the screen. Just click on OK to proceed with the check.

  When the check is complete, design rule errors which exist in the layout will be reported in the message bar at the bottom of the ICstation window. The message should read ... Total Results:0 .... However, you will get errors messages: Port must be completely covered with Metal. It seems that the check itself is incorrect.

  Upon a close inspection you might notice that the I/O ports are arranged on a 3rd metal layer.

- **LVS**: Next, verify the layout for consistency with the transistor level schematic of your design. Do this by returning to the IC Palettes and selecting the ICtrace(M) (mask level LVS) option.

  – Once in the **ICtrace(M)** menu, select **Load Rules** from the palette menu to load a rules file that includes data needed for LVS verification. Specify the rule file **$ADK/technology/ic/tsmc035.rules**

  – Select the **LVS** item in the ICtrace(M) palette to bring up the LVS dialog box. Specify:

  Source Name : $MGC_WD/div3b4/lvs,

  then click on the Setup LVS button. In the Setup LVS form, enter:

  Ground Names : VSS and GND
  Recognize Gates : Yes

  – When the check is complete, the bottom message bar will read

  Mask results database loaded

  To view the results, you can select

Report > LVS from the ICtrace(M) palette.

Look for that magic smiley face. If you get that big nasty X instead, go back and make certain that you have properly set up the LVS viewpoint and followed the above procedure precisely. When doing so, look for potential sources of discrepancies between your schematic and how the layout is generated.

- Save your completed project layout by selecting

  File > **Cell** > **Save Cell** > **Current Context**

It is difficult to accept but you have just arrived to the end of this practical exercise. I trust you have enjoyed it.

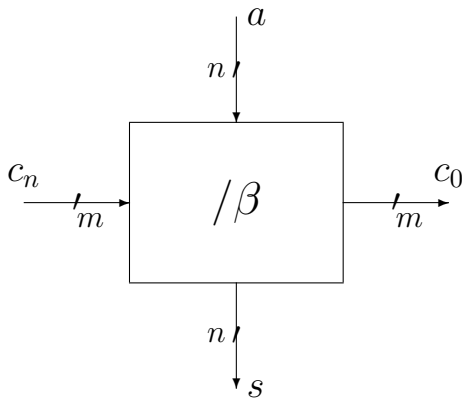
4.9 Demonstration and Report

- 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 regarding the structure of the library and your own inverters.
- In addition, give stick diagrams (use colour pens) of the 1-bit division-by-3 circuits which generates outputs **s** and **d(0)**. For each circuit sketch three stick diagrams using:
  - NAND gates,
  - a “composite” gate concept,
  - a “switch logic” concept.

Appendix: A Division-by-Constant Combinational Circuit

A general case

A combinational circuit which divides n-bit binary number by a 'small' constant β has a modular structure of a parallel recursive circuit, similar to the structure of an incrementer or an adder. The carry propagates from left to right, and its values are limited by the divisor, β.



$(2^n \cdot c_n + a)$ – dividend
β – divisor
s – quotient
c_0 – remainder

where

β is a 'small' integer constant,

$a, s \in \{0, \ldots, 2^n - 1\}$ are n-bit integers,

$c_n, c_0 \in \{0, \ldots, \beta - 1\}$ are m-bit integers, $2^m \geq \beta$

Input/output variables are related by the following equation which links divider, divisor, quotient and remainder:

$$\frac{2^n \cdot c_n + a}{\beta} = s + \frac{c_0}{\beta}$$

which can be written in a more convenient form as

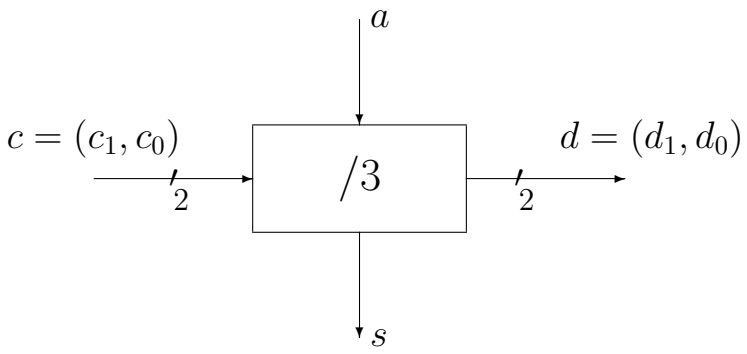
$$2^n \cdot c_n + a = \beta \cdot s + c_0 \tag{1}$$

The objective is to build a combinational circuit, which, given n-bit input a and possibly c_n, will generate the quotient s and the remainder, c_0.
It is possible to build such a circuits using 1-bit cells as explained in the next section.

A 1-bit division-by-3 circuit

In this section we will focus on the special case of division by 3. In this case, we have

$$n = 1; \quad \beta = 3; \quad m = 2; \quad \{2^m \geq \beta\}$$



Input and output carry signals, c, d, are 2-bit numbers which are less than the divisor, $\beta = 3$, that is:

$$c,\ d \in \{0,\ 1,\ 2\}$$

The I/O equation is now of the following form:

$$2 \cdot c + a = 3 \cdot s + d \tag{2}$$

Eqn (2) can be expressed in tabular forms:

Function Table

c	a	$2c + a$ $3s + d$	s	d
0	0	0	0	0
0	1	1	0	1
1	0	2	0	2
1	1	3	1	0
2	0	4	1	1
2	1	5	1	2
3	0	6	–	–
3	1	7	–	–

Truth Table

c_1	c_0	a	s	d_1	d_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	–	–	–
1	1	1	–	–	–

From the tables the logic equations for the three outputs can be derived using the Karnaugh map technique or any Boolean minimisation software tools. The equations should be of the following general form:

$$(s, d_1, d_0) = f(c_1, c_0, a)$$

In order to obtain an n-bit circuit we connect n 1-bit components as illustrated in Figure 3.