MCC092 Integrated Circuit Design: Lab 2

Hierarchical Schematic Design for 8-bit Ripple Carry Circuit

Version 2018-09-24

2012-2018 (65 nm CMOS): Kasyab Subramaniyan, Stavros Giannakopoulos,
Per Larsson-Edefors and Lena Peterson
2008-2011 (130 nm CMOS): Goran Čengić, Lars Svensson,
Simon Kristiansson and Lena Peterson
Chalmers University of Technology

Contents

1 Lab Purpose **2**
2 Pre-Lab Assignments **2**
3 Lab Goal **2**
4 Introduction **2**
5 Drawing the Schematic for a Carry Gate **4**
6 Creating a Symbol for the Carry Gate Cell **6**
7 Creating a 1-bit Ripple-Carry Cell **6**
8 Creating a Test Bench for the 1-bit Ripple-Carry Cell **7**
9 Simulating the 1-bit Ripple-Carry Cell **9**
10 Investigating an 8-bit Ripple-Carry Circuit **11**
11 Extra for Those Who Have Time **14**
12 The End **15**

1 Lab Purpose

The main purpose of this second lab is to reinforce the schematic design phase and to give you an understanding of how complex circuits can be designed using repeated use of simple leaf cells (standard cells) in a hierarchical fashion. In the next lab, lab 3, you will carry out layout work on the cell considered in this second lab.

2 Pre-Lab Assignments

You will find the pre-lab assignment in a separate document available from the course home page in PingPong. You can find it most easily under the assignment called **Pre-lab preparation for lab 2**. This is also where you are to submit your solution to the pre-lab assignment. Note that even though you work in pairs in the lab you are to submit you own individual solution.

At the beginning of the lab you will get your pre-lab solution back with feedback from the teachers. Before you go ahead with the in-lab tasks, you should correct any mistakes you have made in the pre-lab assignment.

3 Lab Goal

During the lab you will carry out the instructions in this memo and show the circuit schematics (on screen) and simulation results to the lab TA. In case you do not finish on time, you will have to do the remaining tasks on your own and report your results to the lab TA. You should show the lab TA these results during, or after, the in-lab session:

- Schematic for both the basic carry gate and the 1-bit ripple-carry cell that you have entered in Cadence Virtuoso.
- Schematic for the test bench for the 1-bit ripple-carry cell.
- Results from a circuit simulation of the 1-bit ripple-carry cell.
- Schematic for the 8-bit ripple-carry circuit.
- Schematic for the test bench for the 8-bit ripple-carry circuit.
- Results from a circuit simulation of the 8-bit ripple-carry circuit.
- The table on the last page.

4 Introduction

In this lab, you will learn more about how to use the Cadence Virtuoso design platform (CV for short). The focus of this lab is on hierarchical composition and simulation, creating a complex

implementation making use of simple schematic cells that you have developed. The other parts of the design cycle, consisting of physical layout and verification, will be performed in the next lab, lab 3.

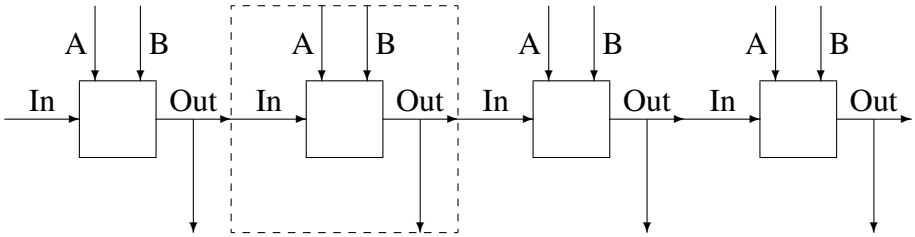

Figure 1: The 8-bit ripple-carry circuit is constructed from eight 1-bit ripple-carry cells. The figure shows only four of these 1-bit cells.

You will perform all the steps in the design flow necessary to implement the 8-bit ripple-carry circuit (see Fig. 1 for an overview) that you have worked on as the preparation assignment. *Note that this is not a complete 8-bit adder since the sum logic is not included.* Here, in lab 2, you will create the *schematic* for the 1-bit ripple-carry cell, which in turn is made up of the compound carry gate that you will design as well as some inverter(s). Then, you will create a *symbol* representing the 1-bit ripple-carry cell, which will be used to represent this cell at higher levels of hierarchy. Then, you will create a *test bench* for your 1-bit ripple-carry cell and investigate its properties through circuit simulations. In fact, you have encountered these steps in the previous lab, for the inverter. What makes lab 2 different is that, based on a basic 1-bit ripple-carry cell, a schematic and a test bench for an 8-bit ripple-carry circuit will be hierarchically created and simulated.

All actions which you should perform are written in bullets. Since you should have read through this instruction before the in-lab session, the different tasks should not take too much time. This requires however that you follow the instructions carefully!

First, start CV in the same directory that you used in the inverter lab.

- Log in on one of the designated computers and start a terminal window.
- Move to the working directory you used last time and initiate and start CV:

```
> cd $HOME/MCC092/cadence
> /usr/local/cad/course/MCC092/Y2018/stm065
```

Open the library you created in the inverter lab using the **Library Manager** and create your basic carry gate. This is the compound carry-generating gate, without any inverters, that you prepared in the pre-lab assignment. You will add the inverters later, to create the 1-bit ripple-carry cell which should have a non-inverting output.

- In **Library Manager**, select the name of your library in the list of libraries below the field `'Library'`.
- Choose **File→New→Cell View...** so the form **Create New File** appears.

- Type in a cell name, e.g., `carry_gate`, in field `'Cell'`. Make sure that field `'View'` is `schematic`. We will next create the circuit schematic of the carry gate.
- Click on **OK**.

5 Drawing the Schematic for a Carry Gate

As for the inverter in lab 1, we will implement a carry gate using transistors from the library `cmos065` and ideal components from the library `analogLib`. This phase, the schematic phase, is very similar to the work done in lab 1. The only difference is that here you are to use the X2 widths: 0.2 μm for nMOS transistors and 0.4 μm for pMOS transistors.

First place the transistors needed for the carry gate in the schematic editing window.

- Press `[i]` so the form **Create Instance** appears.
- Place all `nsvtlp` transistors first:

  ```
  'Library'  cmos065
  'Cell'     nsvtlp
  'View'     symbol
  'Width'    0.2
  'Length'   0.06
  ```

  Click in the workspace and place the transistors according to the schematic you drew as preparation for the lab. Leave plenty of space between the transistors.
- Place all the `psvtlp` transistors and set the widths to `0.4`.
- Finish with `[ESC]` (with the mouse pointer in the schematic window).

As in lab 1, symbols for supply and ground are available in the library `analogLib`; look for `vdd` and `gnd`.

- Insert `vdd` and `gnd` at appropriate places in your schematic.

Now we will connect the different transistors together and with the supply voltages. You can use the command **Create→Wire(narrow)** or, rather, use a shortcut:

- Press `[w]`.
- Connect the different components according to your sketch. Do not forget to connect the bulk nodes of the transistors to `vdd` or `gnd`.

Assign some appropriate names for the signals in your schematic.

- Choose **Create→Wire Name...** or press `[l]`.

- Give names to the signals in your schematic. You may want to, in one form, give several signal names separated by spaces to speed up this process.

Now when the schematic is done we add pins to the schematic, to allow the carry gate to interface to the outside world.

- Press `[p]`. The form **Add Pin** is shown.
- Make sure that `'Attach Net Expression'` is selected as `No`.
- Write the pin names for all inputs, select `'Direction'` to be input and place the pins in the schematic. Do not forget that at this level some of the inputs are inverted and the names should reflect that.
- Open the form again, write the pin name for the output, select `'Direction'` to be output and place the pin in the schematic.
- Connect the pins to the rest of the schematic with wires.

Save the schematic and correct any errors.

- Save the schematic with **File→Check and Save** or press `[shift-X]`.

The schematic capture for the 1-bit carry gate is now finished and it is now time to export the schematic.

- In the schematic window, select **File→Export Image ...**.
- In the new window that appears type in the name that you want of the image and select the appropriate file type. It is strongly suggested to select `.png` as they have good quality and are easy to view in other operating systems.
- Click the folder button and navigate to the directory where the image will be stored and press **Open**.
- It is also suggested you swap the background colors to make the image more clearly readable. White is more readable than black.
- Select any other options as appropriate and press **Save**.

At the moment it is unfortunately not possible to print or export files from `heffalump`. To view the file you just saved, use the utility program `gthumb` to view the resulting png file. Run the command in the terminal window on `heffalump` to view your saved file, here assumed to called `cell.png`:

```
> gthumb cell.png
```

A screen shot can of course save the image to your lab PC if you want to keep it for later. But it is better to take a screen shot of the image shown in `gthump` than of the schematic inside Cadence, due to the black background used in Cadence. You can use the Windows 10 Snipping tool to take a screen shot of a rectangular area on the screen.

6 Creating a Symbol for the Carry Gate Cell

Similar to the procedure for the inverter in lab 1, we will soon use our carry-gate schematic design as a building block in other designs. We thus will create a symbol that represents our new gate as it becomes available as a cell.

- Choose **Create→Cellview→From Cellview...** in the schematic editing window. The form **Cellview From Cellview** appears. Here you specify the type of cell view to create. Make sure that `'To View Name'` is specified to `symbol`.
- Click on **OK**.

The form **Symbol Generation Options** appears next. Here we specify how the pins should be placed along the perimeter of the symbol. The symbol that is created will be rectangular with the pins placed as we have specified. We choose to place the pins for the input signals, `A` and `B`, on the top side. The third input `CIn` is placed on the left-hand side, and the output on the right-hand side.

- Give the positions of the pins as specified above.
- Click on **OK**.

Now a new Schematic window appears showing the newly created symbol. One can change the graphical representation of the symbol if one so desires. We will leave it as it is and carry on with the lab. We can now close the window with the symbol, and the window with the schematic.

7 Creating a 1-bit Ripple-Carry Cell

To prepare the construction of the 8-bit ripple-carry circuit, you now have to create a non-inverting 1-bit carry cell, called `ripple1`, that includes your compound carry gate `carry_gate` and one or several instances of the inverter cell which you added to your library in lab 1. Again, open the library you created in the inverter lab using the **Library Manager** and now create your 1-bit ripple-carry cell.

- In **Library Manager**, select the name of your library in the list of libraries below the field `'Library'`.
- Choose **File→New→Cell View...** so the form **Create New File** appears.
- Type in a cell name, e.g., `ripple1`, in field `'Cell'`. Make sure that field `'View'` is `schematic`.
- Click on **OK**.

- Place one instance of `carry_gate` and the instances of `inverter` you need. Connect them as needed using wires and add pins so that all the pins are non-inverted signals, e.g., `CIn`, `COut`, `A`, and `B`.
- Save your 1-bit ripple-carry cell by pressing `[shift-X]`.

Finally, we have to create a symbol for this cell.

- Like we have done several times before, choose **Create→Cellview→From Cellview...** in the schematic editing window. Ensure that `'To View Name'` is specified to `symbol`.
- Click on **OK**.
- In the **Symbol Generation Options** window, arrange the signals so that `A` and `B` are at the top, `CIn` on the left, and `COut` on the right.

8 Creating a Test Bench for the 1-bit Ripple-Carry Cell

We will now create a test bench to be used when investigating the properties of the 1-bit ripple-carry cell. The test bench will contain one instance of the 1-bit ripple-carry cell, i.e., `ripple1`, a few voltage sources for the signals and the power supplies, and a load capacitor connected to the output.

- After choosing **File→New→Cell View...** in the **Library Manager**, type in a name for the new cell, e.g., `ripple1_tb`.
- Click on **OK**.

In the schematic window that appears, we first place one instance of our ripple-carry cell.

- Press `[i]` or use the command **Add→Instance...** to place the ripple-carry cell `ripple1` into the schematic.

For power supplies, we use the DC source called `vdc` which is available in the library `analogLib`.

- Place an instance each of `vdc`, `vdd`, and `gnd` in the schematic.
- Connect the `vdd` component and the `gnd` component to the `vdc` source's positive and negative pin, respectively.

We will later perform a transient simulation of a digital circuit. For this purpose, the input signals are most easily generated with `vpulse` voltage sources.

- Add three `vpulse` sources to the schematic.

- Connect the positive pin of each `vpulse` source to one input each on the ripple-carry cell.
- Connect the negative pins of the sources to ground, either with the ground component already placed in the schematic, or through additional grounds.

The output of the ripple-carry cell will be loaded by a capacitor. The capacitor is called `cap` and can be found in the `analogLib` library.

- Add a capacitor into your schematic.
- Connect the capacitor between the output of the ripple-carry cell and ground.

It will be easier to identify the different simulation curves in the simulated results if the signal wires are given appropriate names.

- Assign some appropriate names for the input and output wires of the ripple-carry cell, by pressing `[l]` and attach the signal names to the wires.

Check and save as before, and correct any errors. Save the schematic image and show it to the lab TA.

We will now assign values to the different parameters in the schematic. We begin with the `vpulse` components, which are very common in a test bench. As in the inverter lab, we will choose parameters such that all possible input combinations are tested in the simulation. Since our ripple-carry cell has three input signals, this means that we have eight different binary combinations. We also want to be able to measure the propagation delay from `CIn` to `COut` for different combinations of `A` and `B`.

- Open the form **Edit Object Properties** by pressing `[q]`.
- Select one `vpulse` component at a time. Type in the parameter values and press **Apply**. For all these sources we have:

  ```
  'Voltage 1'  0 V
  'Voltage 2'  1.2 V
  'Rise time'  20p s
  'Fall time'  20p s
  ```

  However, the width of the pulses and the period of the signals differ:

  ```
                  CIn   A     B

  'Period'        8n s  4n s  2n s
  'Delay time'    4n s  2n s  1n s
  'Pulse width'   4n s  2n s  1n s
  ```

What remains is to give a value of the supply voltage and a parameter name for the capacitor.

- Select the `vdc` component and set `'DC voltage'` to `1.2 V`.

- Select the capacitor and write for `'Capacitance'` a symbolic value, e.g., `Cload`.
- Save with `[shift-X]`.

The value `Cload` for the capacitor is a *simulation parameter* which we will give a numeric value before we simulate the ripple-carry cell. The name for the parameter is arbitrary. (In a succeeding lab, we will vary the value of the load capacitor; if not we could have just as well given a fixed value.)

9 Simulating the 1-bit Ripple-Carry Cell

In order to investigate the performance of our ripple-carry design, we will now perform circuit simulations. Commence circuit simulation as in lab 1:

- Choose **Launch→ADE-L** in the schematic editing window.

Then we select transient simulation mode and define a stop time.

- Issue **Analyses→Choose...** in the **Virtuoso Analog Design Environment** (ADE-L) window. The form **Choosing Analyses** appears.
- Choose `tran`.
- As `'Stop Time'`, type in `9n`.
- Click on **OK**.

Any symbolic circuit parameters in the schematic must be assigned numerical values. In our case, there is only one such parameter, namely the load capacitor. A value of 1-2 fF is comparable to the input capacitance of a small gate in the 65-nm process technology that we use here.

- Choose **Variables→Copy From Cellview** in the ADE-L window. The capacitor parameter should appear in the field `'Design Variables'`.
- Double click on the capacitor parameter in the field `'Design Variables'`. The form **Editing Design Variables** appears.
- Type `1.5f` in the field `'Value (Expr)'`
- Click on **OK**. Make sure that the capacitance value has appeared in the field `'Design Variables'`.

Now, you have to specify the signals you want to view after the simulation. Like for the inverter in lab 1, you want to study the input and output voltages of the ripple-carry cell, and the current drawn by this cell. We select these in the circuit schematic:

- Choose **Outputs→To Be Plotted→Select On Design** in the ADE-L window.
- In the schematic, select the wires corresponding to the voltages you are interested in.
- In the schematic, also select the positive pin on the supply voltage source to monitor the current flowing.
- Press `[ESC]` when finished.

The selected signals are shown in the field `'Outputs'` in ADE-L.

Finally, you must select the corner to use. If this is not done, no simulation models are loaded since there is no default value and the simulation will abort.

- Issue **ArtistKit→Setup Corners...** in the ADE-L window. A form called **Setup Corners** appears.
- Under Scenario, select TT (the top option) for GLOBAL VARIATIONS.
- Click on **Save Model File** (*Do not confuse it with Save Scenario!*)
- Close the window by selecting **Session→Close**

Before we move on to the simulation, it is wise to save the information entered so far. This way, we can save some time if later on we want to perform the same or similar simulations.

- Choose **Session→Save State...** in the ADE-L window.
- select `Cellview` as the `Save State Option` at the top. With this option the state will be saved inside your testbench schematic rather than in an arbitrary folder somewhere.
- Click on **OK** to save.

Finally, we can start the simulation.

- Choose **Simulation→Netlist and Run** or the corresponding short cut button in ADE-L.

The simulation starts and a separate window appears showing the simulation progresses.

When the simulation has completed a graph window appears, containing the signals we chose to display. Now there are a number of in-lab tasks that you need to perform and later show to the lab TA:

- Verify that the ripple-carry cell implements the correct logic function.
- Measure the largest current drawn by the ripple-carry cell.
- Measure the propagation delay for the ripple-carry cell. The propagation delay will vary with what inputs are changing. Which number should one use?

- Measure the rise and fall times of `COut` in a few cases.

Save an image of the result as you did in the inverter lab. Then show your results to the lab TA.

Before we move on to the next task, it is good to close all simulation windows to avoid unnecessary confusion.

10 Investigating an 8-bit Ripple-Carry Circuit

As promised, we will now use our 1-bit ripple-carry cell as the building block in an 8-bit ripple-carry circuit. Since we have designed our 1-bit ripple-carry cell with hierarchy in mind, this design will be easy: Eight instances of `ripple1` will be connected in series, and the remaining input and output signals will be connected to pins to make an interface to the outside.

First, create a new schematic cell in the `labs` library for the 8-bit ripple-carry circuit and add some instances.

- Choose **File→New→Cell View...** in the **Library Manager**.
- Type in a name, e.g., `ripple8`.
- Click on **OK**.
- Add eight instances of your ripple-carry cell in the schematic. Place them in one row with some space between, so you can easily connect each cell output to the next input, using `[w]`.

We must also add pins for the `ripple8` circuit, reflecting the carry signal from less significant ripple-carry circuits and two 8-bit buses for `A` and `B`. For the bus signals we will use pin names with a *bus syntax* (*xxx*<*m*:*n*>) in order to decrease the number of pins. Bus syntax is briefly discussed in Section 7.5 of the Cadence Crib Sheet.

- Add one input pin, preferably close to the input of the left-most ripple-carry cell. Name it `CIn`.
- Add one output pin, preferably close to the output of the right-most ripple-carry cell. Name it `COut`.
- Add two input pins with names in bus syntax: `A<7:0>` and `B<7:0>`. Make sure that bus expansion is set to off for the pins to work as intended.

The connection of the components is simplified by the naming convention which states that nets with the same name are assumed to be connected, even if they are not connected explicitly using wires in the schematic. Consequently, a large net does not have to be drawn completely; it is sufficient to draw a short wire to each component pin and to let the other end, so to speak "hang in the air" (which is accomplished by a double click). The wires are then connected through the naming of the wires.

- Press [w] and add short wires to each of the bus pins and to each of the A and B inputs of the ripple-carry cells. Double click to end a wire "in the air".
- Name the input signals of the ripple-carry cells A<0>, A<1>, A<2>, etc. You can do that easily by putting A<7:0> as the wire name and then checking the box Bus expansion. That will allow you to click multiple times (8) and name each wire in order. When you name the wires make sure the least significant bit (LSB) is numbered 0 and the most significant bit (MSB) is numbered 7.
- Save the schematic with **File→Check and Save**. Correct any errors before continuing.
- Save the image of the schematic as before. Show it to the TA.

Next we will create a symbol for the 8-bit ripple-carry circuit.

- Choose **Create→Cellview→From Cellview...** in the schematic editing window.
- Accept the library name, cell name, and the view name as before.
- Move A<7:0> and B<7:0> to 'Top Pins' in the **Symbol Generation Options** form.
- Click on **OK**.
- Inspect the symbol and then close the window.
- In addition, close the window with the 8-bit ripple-carry circuit schematic.

Now, we will create a test bench for the 8-bit ripple-carry circuit.

- Choose **File→New→Cell View...** in the **Library Manager**.
- Give a name for the test bench, e.g., ripple8_tb.
- Click on **OK**.

Add components to the test bench the same way as before.

- Add one instance of the 8-bit ripple-carry circuit.
- Add one vdc, one vdd, and one gnd to the schematic and connect them as before.
- Add three vpulse sources and connect their negative pins to ground.
- Connect the positive pin of one of the vpulse sources to CIn of the 8-bit ripple-carry circuit. Give the signal the name CIn.
- Add a load capacitor (cap) and connect it between ground and COut of the 8-bit ripple-carry circuit. Name the signal COut.

The connections from the pulse sources to the ripple-carry cell inputs A and B are a little bit different in this case, since these input signals are buses. We again exploit that signals can be connected by giving them the same name in the schematic.

- Connect short wires to the positive terminals of the two remaining `vpulse` sources and the `A<7:0>` and `B<7:0>` inputs of the ripple-carry cells.
- Choose **Create→Wire Name...** to name the wires connected to the `vpulse` sources. Use the names `In0` and `In7`, where 0 and 7 refer to the bit positions in the ripple-carry cells.

The bus signals connected to the inputs of the ripple-carry cells are given names that are *lists* of simple signal names.

- Name the `B` input bus `vdd!,vdd!,vdd!,vdd!,vdd!,vdd!,vdd!,vdd!` (no spaces!).
- Name the `A` input bus `In7,gnd!,gnd!,gnd!,gnd!,gnd!,gnd!,In0` (no spaces!).

The most significant and the least significant bits of the `A` input have in this way been connected to one `vpulse` source each. The remaining bits are connected to the global signal `gnd!`. (Read more about global signals in the Cadence Crib Sheet, Section 7.5.)

Finally, we assign parameter values to the load capacitor and the sources and save our work.

- Select the load capacitor, open **Object Properties** (with `[q]`), and name it `Cload`.
- Set the `vdc` source voltage to `1.2 V`.

Using our three `vpulse` sources, we want to achieve that the signal is propagated from the different inputs to the output.

- Assign the following parameter values to the sources:

'Voltage 1'	0 V	
'Voltage 2'	1.2 V	
'Period'	60n s	
'Rise time'	40p s	
'Fall time'	40p s	
'Pulse width'	10n s	
'Delay time'	10n s	(in7)
	30n s	(in0)
	50n s	(cin)

- Save the cell with `[X]`. Correct any errors and save again.
- Save the image of the schematic and show it to the lab TA.

The test-bench design is now ready for simulation. Basically the same way as for the 1-bit ripple-carry cell test bench, we outline the procedure below:

- Open the window **Virtuoso Analog Design Environment** with **Launch→ADE-L**.
- Issue **Analyses→Choose...** and choose `tran`. Set the stop time to `100n`.

- Issue **Variables→Copy From Cellview**. Set the parameter value for the load capacitance to `1.5f` as before. (Make sure there is no space between `1.5` and `f`).
- Issue **Outputs→To Be Plotted→Select On Design**. Choose to plot the three inputs and the single output of the ripple-carry cell, and the current going through the DC supply source.
- Select the `TT` corner for the analysis.
- Using a descriptive name for this state, issue **Session→Save State**.
- Start the simulation with **Simulation→Netlist and Run**.
- Take a look at the power supply current and how it varies over time. Zoom in (press `[z]`) so you can study the current profile in detail. If the profile varies with the input that is switching, what is the relation between input transition and current profile?
- Measure the propagation delay (t_{pd}) from the inputs to the output and fill in the table below. How does the propagation delays from the different inputs relate to each other? Can you explain the differences in propagation delays? Compare your measured data with your delay calculations from the pre-lab. Do they match? If there is a large difference try to figure out why.

Rising inputs			Falling inputs		
At time	Input signal	Propagation delay t_{pd}	At time	Input signal	Propagation Delay t_{pd}
10 ns			20 ns		
30 ns			40 ns		
50 ns			60 ns		

Save an image of the simulation result and show to the lab TA.

- Save the simulation results with **Results→Save...** in the ADE-L window. Make sure you choose an appropriate name here; the result will be used in a later lab.

You are now finished and can close down CV, unless you want to continue to do some of the extra experiments listed in section 11.

- Choose **File→Exit ...** in the CIW.
- Save the simulation state for future use.

11 Extra for Those Who Have Time

The 1-bit ripple-carry cell that you designed is not speed optimal, since there is an unnecessary inverter inserted between the carry generation gates. (We used this scheme because it makes all ripple-carry cells and their connections identical.) You can try a couple of things to see how much you can improve the delay.

First, you have not optimized the size of the inverter w.r.t the Cin input. Calculate the optimal inverter size, change the sizes in your X4 inverter to this size and rerun the delay measurement for the 8-bit carry cell. Does the improvement match your calculation? Change the size of the X4 inverter back to what it should be after this experiment.

Another possibility is to use the speed-optimal full-adder cell from Weste and Harris. That schematic, shown in Figure 11.4(c) of Weste and Harris (red book), is available for you to copy. See instructions on the course PingPong page for how to do this. The only difference in our schematic from the one in the book, is that we have included the inverters for the carry output so that one does not have to invert the A and B inputs in every other cell. We have used the size (1) in the figure to mean 0.2 μm.

Repeat the experiments above but instead use the speed-optimal carry circuit in Figure 11.4(c) of Weste and Harris. Fill in the table below for this design. How do the delays compare with the ones for your ripple-carry cell?

Rising inputs			Falling inputs		
At time	Input signal	Propagation delay t_{pd}	At time	Input signal	Propagation Delay t_{pd}
10 ns			20 ns		
30 ns			40 ns		
50 ns			60 ns		

Another experiment you can try is to use the slow (SS) and fast (FF) corners instead of the typical model values (TT) and see how much the delay for the 8-bit carry chain changes.

Finally, you can change the order of the transistors in the n- and p-nets that are in the critical path in your compound gate to see how much the delay for the 8-bit carry chain changes.

12 The End

We have now completed the schematic design and simulation cycle for a hierarchical design. We have investigated the performance of our hierarchical design using circuit simulations at several levels. There is some overhead in handling all these small cells for a limited design like a 8-bit ripple-carry circuit. However, for larger designs, this hierarchical way of working is indispensable.

In the next lab you will learn to make the physical layout and how to convince yourself that the layout matches the circuit schematic.