Power-gated MOS current-mode logic

Master semester project, spring 2009

Date: 27.05.2009
Author: Michael Schwander
Assistants: Stéphane Badel
           Alessandro Cevrero
Professor: Yusuf Leblebici
Laboratory: LSM, Microelectronic systems laboratory
Power-gated MOS current-mode logic
Michael Schwander, EPFL 2009

Abstract
Abstract

In today’s mixed signals applications the noise transmission from the noisy digital part to the mostly sensitive analog part becomes a problem, because of loss in accuracy of the analog circuits or in a reduction of the dynamic range. In CMOS the noise is basically due to current peaks and high voltage variations during the switching of logic states. To solve this problem several solutions have been proposed. One of these is a different logic style, called MCML. Being fully differential and using a constant current consumption, the voltage swing is reduced and the current peaks got eliminated. So less noise will be propagated to the next stage.

Other advantages of MCML logic gates compared with CMOS are that they are also faster and show a higher robustness against process variations, but on the other hand, more silicon area is used. The constant current consumption of MCML gates leads to a higher power consumption, when not operated in high speed applications.

The goal of this project is to introduce a sleep mechanism in a MOS current mode logic (MCML) library to reduce the constant power consumption, which is the main disadvantage of this logic style. In a first step different sleep methods are shown and analyzed. The best method will be tested and used to design a cell library. Solutions to save the state in a flip-flop are shown and evaluated. Finally a test circuit on schematic level will be done and simulated.

The schematics simulation of this circuit shows that for this application 40% of the average power consumption can be saved, assuming that the whole circuit is running all the time. If it can be turned off, even more power can be saved, due to the overall sleep mechanism. The following table shows the results for the whole circuit without any sleep mechanism and the circuit with the gates including the sleep mechanism developed in this project.

<table>
<thead>
<tr>
<th>Average values</th>
<th>MCML without Sleep</th>
<th>MCML with Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ion [A]</td>
<td>40.68m</td>
<td>24.4m</td>
</tr>
<tr>
<td>Ioff [A]</td>
<td>40.68m</td>
<td>1.86m</td>
</tr>
<tr>
<td>Power on [W]</td>
<td>34.58m</td>
<td>20.74m</td>
</tr>
<tr>
<td>Power off [W]</td>
<td>34.58m</td>
<td>1.58m</td>
</tr>
</tbody>
</table>

Table 1: Average current and power consumption when the circuit is running (on) and when the circuit is unused (off)
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>5</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2 Design of MOS CML gates</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Principle</td>
<td>8</td>
</tr>
<tr>
<td>2.2 One stage design</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Two or more stages</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Control circuit</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Sizing of a gate</td>
<td>12</td>
</tr>
<tr>
<td>2.4.1 The current source</td>
<td>13</td>
</tr>
<tr>
<td>2.4.2 The load resistors</td>
<td>13</td>
</tr>
<tr>
<td>2.4.3 The differential pair network</td>
<td>14</td>
</tr>
<tr>
<td>2.4.4 Buffer size</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Results</td>
<td>14</td>
</tr>
<tr>
<td>3 Various sleep methods</td>
<td>16</td>
</tr>
<tr>
<td>3.1 Bulk voltage control</td>
<td>17</td>
</tr>
<tr>
<td>3.1.1 Functionality</td>
<td>17</td>
</tr>
<tr>
<td>3.1.2 Schematics simulation</td>
<td>17</td>
</tr>
<tr>
<td>3.1.3 Integration in the layout of standard cells</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Sleep transistor</td>
<td>19</td>
</tr>
<tr>
<td>3.2.1 Functionality</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2 Schematics simulation</td>
<td>19</td>
</tr>
<tr>
<td>3.2.3 Integration in the layout of standard cells</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Comparison and choice</td>
<td>22</td>
</tr>
<tr>
<td>4 D flip-flops</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Problem</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Different solutions</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Comparison and choice</td>
<td>27</td>
</tr>
<tr>
<td>4.4 Sizing and simulation of one D-Latch</td>
<td>27</td>
</tr>
<tr>
<td>5 Test circuit</td>
<td>29</td>
</tr>
<tr>
<td>5.1 Description</td>
<td>29</td>
</tr>
</tbody>
</table>
1 Introduction

In today’s mixed signals applications the noise transmission from the noisy digital part to the mostly sensitive analog part becomes a problem, because of loss in accuracy of the analog circuits or in a reduction of the dynamic range. In CMOS the noise is basically due to current peaks and high voltage variations during the switching of logic states. To solve this problem several solutions have been proposed. One of these is a different logic style, called MCML. Being fully differential and using a constant current consumption, the voltage swing is reduced and the current peaks got eliminated. So less noise will be propagated to the next stage.

Other advantages of MCML logic gates compared with CMOS are that they are also faster and show a higher robustness against process variations. But on the other hand, more silicon area is used. The constant current consumption of MCML gates leads to a higher power consumption, when not operated in high speed applications.

A new logic family, called Dynamic CML (DyCML)[3], was introduced, to eliminate the static power consumption. This was achieved by replacing the current source of a CML circuit, by a dynamic current source and by adding a latch to save the output stage. But due to the noise sensibility and the difficulty to implement such kind of gates, this method is not used anymore.

The goal of this project is to elaborate another method to solve the power problems on a higher level. Trying to add some kind of a sleep mechanism to every cell the goal is to temporarily shut down whole blocks, which are not used in the next few clock cycles. In this case, just the blocks, that are really used, consume power while the others are switched off. Different methods will be tested and analyzed. The best method will be used after to develop a cell library on schematics level. Using this library a test circuit will be created, simulated and analyzed.
2 Design of MOS CML gates

2.1 Principle

A MCML gate consists of three different blocks. The first block is the current source, which determines the current consumed by the gate and is therefore in charge of the static power consumption of the gate. The second block defines the logic function using one or more differential pairs to switch the current. Depending on the complexity of the logic function, one or more stages are needed. The last block is the load of the gate.

![Principle of a MCML gate](image)

The current switched by the second bloc, will pass one of the two resistors and pull this level down to logic ‘0’, while no current is flowing in the other resistor and so the level stays at logic ‘1’. So the output voltage swing depends ideally just on the current delivered by the current source and the resistor R:

\[ V_{SW} = I \cdot R \]

To reduce the area and for an improvement in noise, the load resistors are replaced by two PMOS transistors, which are controlled by a gate voltage \( V_P \). The current source is implemented using a NMOS transistor, which is also controlled by a gate voltage \( V_N \). The highest speed and also the lowest supply voltage, for which the gate is still working, can be obtained by using just low-Vth transistors. But in this case the leakage current will be
maximized. Therefore a high-Vth transistor is used for the current source, because we will try to turn this source off, when the gate is not active. The drain voltage of this source is constant, when the gate is on, so there is no loss during operation due to this high-Vth transistor.

<table>
<thead>
<tr>
<th>Block</th>
<th>Vth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current source</td>
<td>High-Vth</td>
</tr>
<tr>
<td>Logic function</td>
<td>Low-Vth</td>
</tr>
<tr>
<td>Load</td>
<td>Low-Vth</td>
</tr>
</tbody>
</table>

Table 1: Vth used for different transistors

### 2.2 One stage design

For the better understanding of MCML gates, the buffer / inverter gate is shown in this chapter. As MCML logic is differential the inverse of the function is always automatically given by just inverting the outputs.

When the input is changing from low to high, the differential pair will switch the current from the right to the left side. The current now flowing through the left resistor will pull down \( V_{out} \) to logic ‘0’, while \( V_{out+} \) will go up to \( V_{dd} \).

### 2.3 Two or more stages

Gates using two stages consist either of three differential pairs or of two differential pairs and a dummy transistor, which is used to get the same number of transistors in every possible path.

In figure 3 the AND gate is shown, which uses a dummy transistor and two differential pairs. To get a logic ‘1’ at the output, both inputs A and B have to be equal to ‘1’, otherwise the output will be at logic ‘0’. So this gate is really implementing an AND function.
An example for a two stage gate with three differential pair is the multiplexer 2to1 (figure 4, right side). In this case, the three Inputs are present. In the first stage on the bottom the select signal and on the second stage the two input signals, which can be selected. As the example of the XOR gate shows, there are also two stage gates with three differential pairs using just two inputs (figure 4, left side).

Figure 3: AND gate

Figure 4: XOR gate (left) and MUX2to1 (right)
For more complex gates like XOR3 (figure 5, left side), multiplexer 4to1 and majority function 3to2 (figure 5, right side) three stages have to be used. Where for the multiplexer just a stage with four differential pairs is added, the other functions are little bit more special. In general N stages can be used, when Vdd is high enough. During this project just gates with up to three stages are looked at, because every standard function can be done with maximal three stages.

![Figure 5: MAJ32 gate (left) and XOR3 gate (right)](image)

### 2.3 Control circuit
For MCML standard cells a control circuit is needed, to generate the VN and VP voltages, which will be connected directly to the gates of the load and the current source in every gate. If these voltages are generated depending on a reference gate and compared to the ideal values of Vsw or respectively the delivered current, process variations can be corrected and are in this case less important. As a reference cell a simple buffer is used, with a logic ‘1’ at the input. The negative output of this gate is then compared to the desired logic ‘0’ voltage (Vdd-Vsw). The voltage generated by this feedback loop is the VP voltage for all cells.
The VN voltage is generated such that, the current passing through the active path is equal to the reference current, which can be set in the VN generation block. The amplifiers are modeled by a voltage controlled voltage source (VCVS). To limit the useful frequency domain, a RC filter is added at the output.

2.4 Sizing of a gate

In this chapter a buffer will be sized and analyzed with following specifications:

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vdd</td>
<td>0.85 V</td>
</tr>
<tr>
<td>Vsw</td>
<td>0.4 V</td>
</tr>
<tr>
<td>Noise Margin (NM)</td>
<td>&gt; 50mV</td>
</tr>
<tr>
<td>Iref</td>
<td>50 uA</td>
</tr>
</tbody>
</table>

Table 2: Specifications

Vdd is set to the minimal working voltage 0.85 V, which was determined after several simulations to assure a proper functionality also for corner cases and for up to three stage gates. For every simulation the following corner cases are checked additionally to the typical case:
The static noise margin is defined as the maximal distance between the input and the output signal as shown in figure 8. For every gate the noise margin has also to be elevated for all corner cases to assure, that there is never a noise margin less than given in the specifications.

The reference current defines the speed and also the static power consumed by a gate. Therefore the trade-off between power consumption and speed determines the value of this current. It can easily be changed and adapted for a given circuit.

### 2.4.1 The current source

The current source should be operated in saturation to achieve the lowest voltage drop possible, which will let the Vdd as scalable as possible. The larger the W/L ratio, the smaller is the saturation voltage, what is exactly what is needed. This ratio is limited by the area used for a too large with. The length of the current source is limited by the minimal length of the technology and by the fact that a better matching is achieved for non minimal length. If the current is better matched, there will be less variation in speed and power consumption and therefore fewer margins have to be put in the final specifications of the gate.

### 2.4.2 The load resistors

Because the load transistors should do the work of two resistors, there are likely used in the linear region, where their transfer function can be approximated by a linear resistor. The sizing
of these transistors is directly given by the desired voltage swing $V_{sw}$ for the given current by the current source. Also here a non minimal length is desired to get the resistors matched. Once the length is set, the width can be obtained, using the following formulas:

$$V_{sw} = R \cdot I \quad R \sim \frac{L}{W}$$

### 2.4.3 The differential pair network

The highest speed can be obtained by using minimal size transistors, which clearly also minimizes the silicon area used. But the noise margin of the gate is also depending on the W/L ratio and therefore the lower limit of the width is set by the noise margin, while for the length the minimum can be used. Eventually used dummy transistors are sized the same way, because the load should be the same for every path. Regarding the layout it can be desirable that all transistors in this block have the same width in order that they could be merged together. This solution will be very place efficient, when not too complex gates are used.

### 2.4.4 Buffer size

Using the design criteria’s shown above the following values are used for the buffer:

<table>
<thead>
<tr>
<th>Block</th>
<th>Type</th>
<th>Width [m]</th>
<th>Length [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current source</td>
<td>NMOS</td>
<td>1u</td>
<td>240n</td>
</tr>
<tr>
<td>Load resistors</td>
<td>PMOS</td>
<td>500n</td>
<td>240n</td>
</tr>
<tr>
<td>Differential pair</td>
<td>NMOS</td>
<td>320n</td>
<td>80n</td>
</tr>
</tbody>
</table>

Table 4: Buffer size

All bulk connections were made to Vss or to Vdd according the type of transistor used. A very small advantage could be taken from the change of the bulk voltage in schematic simulations, but regarding to the separate wells needed to lay it out and the associated increase of used area, there is not really a benefit to take.

### 2.5 Results

The simulation result plots of the buffer cell can be found in chapter 9.C. The most important results are shown in the following table. If nothing is mentioned the value corresponds to all transistors in typical configuration.

<table>
<thead>
<tr>
<th></th>
<th>Typical</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$ [A]</td>
<td>49.24u</td>
<td>49.24u</td>
<td>54.24u (lvt_slow)</td>
</tr>
<tr>
<td>$V_{out_high}$ [V]</td>
<td>847.2m</td>
<td>812.1m (lvt_slow)</td>
<td>847.2m</td>
</tr>
<tr>
<td>$V_{out_low}$ [V]</td>
<td>456.6m</td>
<td>275.1m (lvt_slow)</td>
<td>456.6m</td>
</tr>
<tr>
<td>$V_{out_swing}$ [V]</td>
<td>390.6m</td>
<td>390.6m</td>
<td>537m (lvt_slow)</td>
</tr>
<tr>
<td>$VP$ [V]</td>
<td>63.55m</td>
<td>381.6p (lvt_slow)</td>
<td>197.7m (lvt_fast)</td>
</tr>
<tr>
<td>$VN$ [V]</td>
<td>651.7m</td>
<td>607m (hvt_fast)</td>
<td>782.1m (lvt_slow)</td>
</tr>
<tr>
<td>$NM$ [V]</td>
<td>66.75m</td>
<td>51.31m (hvt_slow)</td>
<td>171.6m (lvt_slow)</td>
</tr>
</tbody>
</table>

Table 5: Simulation results of a buffer cell
The current and the output swing is close to the given specifications. Further we can observe that the critical value for the noise margin is the corner case with the high Vth transistors slow, but that also this case fulfils the specifications.
3 Various sleep methods

In this chapter various sleep methods are presented and two of them further developed and evaluated. The idea behind all sleep mechanisms is to shut down the current source or to simply cut off the current. If it will be possible to simply shut down the current source, the sleep mechanism will be for free, because the current source is already present in the design. The high-Vth used for the current source is also in favors such a solution. Important is the shut down and the restart speed of the gate, because finally whole circuits have to be shut down or powered on in just a few or even better in less than one clock cycle.

Figure 10: First (left) and second (right) sleep method

The first method consists of an analog amplifier in voltage follower configuration and a NMOS transistor put between the gate of the current source and ground to put the gate voltage down to ground. In this case the gate can be turned off very fast, but because off the big capacitance that have to be charged by the analog voltage a lot of time will be needed to turn it on again.

A better approach uses two switches, one to pull down the gate voltage of the current source to ground and the other one to separate the VN voltage from the gate. But also here the restart time is expected to be too high.
3 Various sleep methods

3.1 Bulk voltage control

The current delivered by the current source assuming that the source is connected to ground is defined by:

\[ I_D = \frac{\mu}{2} \cdot (V_G - V_{th})^2 \quad \text{where} \quad V_{th} = V_{th0} + \gamma\left(\sqrt{V_{SB}} + 2\phi_F - \sqrt{2\phi_F}\right) \]

So for a fixed transistor size we have two possibilities to change the current. Either we can change the gate voltage as we did for the normal gates, or we can change the threshold voltage by changing the source-bulk voltage. The range of the bulk voltage ensuring a correct functionality of the transistor is limited and therefore it has to be checked, that this method is able to correct all possible process variations. To generate the new bulk voltage the same VN voltage generation circuit can be used and therefore no further transistors are needed.

3.1.1 Functionality

An external rail to rail sleep signal will be applied at the gate. The gate voltage will depend on the sleep signal therefore either be Vdd or Vss. So the current source is either providing current or is blocked.

3.1.2 Schematics simulation

Two different gates are simulated to show the performances of the bulk voltage control mechanism. The sizing of the transistors can be found in the following tables (Buffer: table 6, And: table 8). The plots of the simulation results are annexed in chapter 9.C.

Another method will be to use the bulk voltage of the current source to correct the process variations, while the gate of the transistor is just set to Vdd or Vss according the sleep signal. This method seems to be very interesting and is described in 3.1.

Also the approach of simply adding a sleep transistor in series will be described (3.2) and compared to the bulk control method. This method seems not to be as advantageous as the other one, because an extra transistor has to be added between the current source and the other blocks, while the transistors added in the other approaches don’t affect the gate itself in on mode.

Figure 11: The bulk voltage control method (left) and the simple sleep transistor (right)
### 3.1.2.1 Buffer:

<table>
<thead>
<tr>
<th>Transistors</th>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Source</td>
<td>240n</td>
<td>440n</td>
</tr>
<tr>
<td>Logic Block</td>
<td>80n</td>
<td>320n</td>
</tr>
<tr>
<td>Load</td>
<td>240n</td>
<td>600n</td>
</tr>
</tbody>
</table>

Table 6: Sizing of a bulk voltage controlled buffer cell

<table>
<thead>
<tr>
<th>Typical</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>I [A]</td>
<td>50.01u (lvt_slow)</td>
<td>47.73u (lvt_slow)</td>
</tr>
<tr>
<td>Vout_high [V]</td>
<td>846.5m (lvt_slow)</td>
<td>835.2m (lvt_slow)</td>
</tr>
<tr>
<td>Vout_low [V]</td>
<td>455.9m (lvt_slow)</td>
<td>364.7m (lvt_slow)</td>
</tr>
<tr>
<td>Vout_swing [V]</td>
<td>390.6m (hvt_slow)</td>
<td>383.6m (hvt_slow)</td>
</tr>
<tr>
<td>VP [V]</td>
<td>56.76m (lvt_slow)</td>
<td>458.3p (lvt_slow)</td>
</tr>
<tr>
<td>VN [V]</td>
<td>198.9m -418.3m (hvt_fast)</td>
<td>849.8m (lvt_slow)</td>
</tr>
<tr>
<td>NM [V]</td>
<td>92.61m 52.05m (hvt_slow)</td>
<td>172.4m (lvt_slow)</td>
</tr>
</tbody>
</table>

Table 7: Simulation result of a bulk voltage controlled buffer cell

### 3.1.2.2 AND gate:

<table>
<thead>
<tr>
<th>Transistors</th>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Source</td>
<td>240n</td>
<td>530n</td>
</tr>
<tr>
<td>Logic Block</td>
<td>80n</td>
<td>550n (stage 1) / 360n (stage 2)</td>
</tr>
<tr>
<td>Load</td>
<td>240n</td>
<td>550n</td>
</tr>
</tbody>
</table>

Table 8: Sizing of a bulk voltage controlled AND gate

<table>
<thead>
<tr>
<th>Typical</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>I [A]</td>
<td>49.8u (lvt_slow)</td>
<td>43.04u (lvt_slow)</td>
</tr>
<tr>
<td>Vout_high [V]</td>
<td>837.1m (lvt_slow)</td>
<td>807.3m (lvt_slow)</td>
</tr>
<tr>
<td>Vout_low [V]</td>
<td>450m (lvt_slow)</td>
<td>330.9m (lvt_slow)</td>
</tr>
<tr>
<td>Vout_swing [V]</td>
<td>387.1m</td>
<td>387.1m</td>
</tr>
<tr>
<td>VP [V]</td>
<td>69.31n 68.48n (lvt_slow)</td>
<td>150.1m (lvt_fast)</td>
</tr>
<tr>
<td>VN [V]</td>
<td>84.67m -504.7m (hvt_fast)</td>
<td>849.8m (lvt_slow)</td>
</tr>
<tr>
<td>NM</td>
<td>68.31m 62.34m (hvt_slow)</td>
<td>120.1m (lvt_fast)</td>
</tr>
</tbody>
</table>

Table 9: Simulation results of a bulk voltage controlled AND gate

We observe more or less all values in the expected range close to the specifications. Just for the AND gate the current drop due to a slow low Vth transistor is more than 10%. But also the control voltage of the current source has a very big range starting at about -0.5V and ending at about Vdd. These limits can be moved down with bigger sizes of the transistors, but the range won’t decrease. The negative limit is given by the breakdown voltage of the junction diodes, which is at about -1.2V. The upper limit is given by the increasing current in the well due to the
forward bias junction. So it will be difficult to get a correctly working current source in every case, especially when we expect even an increasing range for more complex gates.

### 3.1.3 Integration in the layout of standard cells
As there is no additional transistor in this method the implementation regarding layout is easy. But because the bulk is not connected to ground a separate well, which consumes a lot of area, has to be placed for the current sources.

### 3.2 Sleep transistor
In this chapter a simple sleep transistor is added on top of the current source as shown in figure 8. The current source stays untouched and will do the task of the process variation correction.

#### 3.2.1 Functionality
Also here an external rail to rail sleep signal will be applied to either switch on or off the sleep transistor. The cut off effect in this case is even larger, because applying a zero voltage at the gate of the sleep transistor will result even in a negative gate to source voltage. Although this transistor is added inside every gate, the speed of the gate in on mode is not affected, because the voltage on top of the sleep transistor is not switching and therefore represents an AC ground.

#### 3.2.2 Schematics simulation
Two different gates are simulated to show the performances of the bulk voltage control mechanism. The sizing of the transistors can be found in the following tables (Buffer: table 10, And: table 12). The plots of the simulation results are annexed in chapter 9.C.

#### 3.2.2.1 Buffer:

<table>
<thead>
<tr>
<th>Transistors</th>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Source</td>
<td>240n</td>
<td>1.2u</td>
</tr>
<tr>
<td>Sleep Transistor</td>
<td>80n</td>
<td>1.2u</td>
</tr>
<tr>
<td>Logic Block</td>
<td>80n</td>
<td>350n</td>
</tr>
<tr>
<td>Load</td>
<td>240n</td>
<td>600n</td>
</tr>
</tbody>
</table>

Table 10: Sizing of a buffer cell including a sleep transistor

<table>
<thead>
<tr>
<th></th>
<th>Typical</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>I [A]</td>
<td>49.29u</td>
<td>49.18u (lvt_slow)</td>
<td>51.5u (lvt_slow)</td>
</tr>
<tr>
<td>Vout_high [V]</td>
<td>847.2m</td>
<td>824.4m (lvt_slow)</td>
<td>848.6 (lvt_fast)</td>
</tr>
<tr>
<td>Vout_low [V]</td>
<td>456.2m</td>
<td>307.9m (lvt_slow)</td>
<td>459.4m (hvt_slow)</td>
</tr>
<tr>
<td>Vout_swing [V]</td>
<td>391m</td>
<td>384.2m (lvt_slow)</td>
<td>516.5m (lvt_slow)</td>
</tr>
<tr>
<td>VP [V]</td>
<td>63.8m</td>
<td>386.5p (lvt_slow)</td>
<td>197.8m (lvt_fast)</td>
</tr>
<tr>
<td>VN [V]</td>
<td>669.4m</td>
<td>613.3m (hvt_fast)</td>
<td>850m (lvt_slow)</td>
</tr>
<tr>
<td>NM</td>
<td>68.94m</td>
<td>53.73m (hvt_slow)</td>
<td>136m (lvt_slow)</td>
</tr>
</tbody>
</table>

Table 11: Simulation results of a buffer cell with sleep transistor
### 3.2.2.2 AND gate:

<table>
<thead>
<tr>
<th>Transistors</th>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Source</td>
<td>240n</td>
<td>1.2u</td>
</tr>
<tr>
<td>Sleep Transistor</td>
<td>80n</td>
<td>1.2u</td>
</tr>
<tr>
<td>Logic Block</td>
<td>80n</td>
<td>620n</td>
</tr>
<tr>
<td>Load</td>
<td>240n</td>
<td>600n</td>
</tr>
</tbody>
</table>

Table 12: Sizing of an AND gate with sleep transistor

<table>
<thead>
<tr>
<th></th>
<th>Typical</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I [\text{A}]$</td>
<td>48.63u</td>
<td>47.7u (lvt_slow)</td>
<td>49.21u (lvt_fast)</td>
</tr>
<tr>
<td>$V_{\text{out_high}} [\text{V}]$</td>
<td>843m</td>
<td>833m (hvt_slow)</td>
<td>847.8m (hvt_fast)</td>
</tr>
<tr>
<td>$V_{\text{out_low}} [\text{V}]$</td>
<td>458.1m</td>
<td>308.8m (lvt_slow)</td>
<td>458.8m (lvt_slow)</td>
</tr>
<tr>
<td>$V_{\text{out_swing}} [\text{V}]$</td>
<td>384.9m</td>
<td>374.2m (hvt_slow)</td>
<td>535m (lvt_slow)</td>
</tr>
<tr>
<td>$V_P [\text{V}]$</td>
<td>63.8m</td>
<td>898.8p (lvt_slow)</td>
<td>197.8p (lvt_fast)</td>
</tr>
<tr>
<td>$V_N [\text{V}]$</td>
<td>669.4m</td>
<td>613.3m (hvt_fast)</td>
<td>850m (lvt_slow)</td>
</tr>
<tr>
<td>NM</td>
<td>76.44m</td>
<td>58.99m (hvt_slow)</td>
<td>174.5m (lvt_slow)</td>
</tr>
</tbody>
</table>

Table 13: Simulation results of an AND gate with sleep transistor

It can be observed, that all specifications are fulfilled and there are less variations. But for slow low $V_{\text{th}}$ transistors the control voltage $V_N$ is saturated and therefore the limit of possible process variation correction is reached. But as the current and voltage variations aren’t important the correction is strong enough.

### 3.2.3 Integration in the layout of standard cells

Using high $V_{\text{th}}$ and the same width as for the current source the sleep transistor can easily be merged inside the current source, so just the price of the bigger width have to be paid. Some test layouts are drawn to estimate the impact of this additional transistor (9.2 Appendix B). The experience shows that the loss in area is not significant, because the width of the cell is normally dictated by the logic block. Just in the case of a simple buffer / inverter the cell area would be bigger, because in this case the width of the current source with the sleep transistor included will be larger than the logic block. In the following table some area estimations are given for standard cells with the sleep mechanism included.
Figure 12: Sleep transistor merged into a buffer cell

<table>
<thead>
<tr>
<th>Gate</th>
<th>Width</th>
<th>Area</th>
<th>Loss due to sleep mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer / Inverter</td>
<td>2.3u</td>
<td>11.132</td>
<td>1.839</td>
</tr>
<tr>
<td>AND / NAND</td>
<td>2.42u</td>
<td>11.713</td>
<td>0</td>
</tr>
<tr>
<td>OR / NOR</td>
<td>2.42u</td>
<td>11.713</td>
<td>0</td>
</tr>
<tr>
<td>MUX</td>
<td>2.78u</td>
<td>13.068</td>
<td>0</td>
</tr>
<tr>
<td>D-Latch (state saving)</td>
<td>3.84u</td>
<td>18.586</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 14: Estimated size and overhead of layout including sleep transistor
### 3.3 Comparison and choice

<table>
<thead>
<tr>
<th></th>
<th>Bulk correction method</th>
<th>Sleep transistor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional transistors</strong></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Expected area</strong></td>
<td>Bigger due to separate well</td>
<td>Smaller, can be merged into the current source</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>No affection</td>
<td>No affection</td>
</tr>
</tbody>
</table>
| **Problems**           | - High bulk voltage variations
                        | - Negative voltages needed
                        | - More than 10% current drop possible
                        | - Sleep signal rail to rail |
|                       | - VN can get saturated |

| Speed                  | No affection           |
|                       | No affection           |

Table 15: Comparison of the two methods

Due to the occurring problems with the bulk controlled correction and also the smaller expected area for sleep transistors, this method will be used and integrated in the gates in the remaining parts of this project.
4 D flip-flops

4.1 Problem
For several applications it can be favorable that flip-flops save their state also during the shutdown mode. But with the actual gate structure every output node will go up to logic ‘1’, when the gate enters the sleep mode and so the internal state of the gate is lost. Different solutions will be presented and evaluated in this chapter.

4.2 Different solutions
A. Adding a multiplexer in front
The simplest way to solve this problem is by adding a multiplexer in front of the gate with the output reconnected to one input. With the differential sleep signal the input can be switched between the real input in normal state and the output of the gate itself in sleep mode. The internal state will be saved. The disadvantage of this solution is the fact, that during off mode no power can be saved. If no speed will be lost, even additional power will be wasted in the multiplexer.
B. Adding a transistor to short the clock stage
Also an additional transistor between the current source and the second stage can be imagined, which will short the clock input to the internal node. So no new data can be loaded in the flip-flop and the internal stage would be kept. The problem with this method is the fact, that both clock inputs will go high, when the cell before is also turned off. In this case it will be very difficult to force the current to pass through the new transistor and not through the two of the differential pair, which have already $V_{dd}$ at the gate. During simulations no solution was found to solve this problem and anyway the power consumption problem is not solved.
C. Adding a sleep transistor and a second current source

Therefore a sleep transistor like in every other cell combined with a second current source connected directly to the second stage will be tested. Where due to the second current source the internal status can be saved and because speed isn’t an issue in sleep mode, the current can be as low as it doesn’t losses the saved state. With this mechanism power can be saved during the sleep mode of the block, without losing the state. Unfortunately due to the switching off of the cell before, both nodes of the clock input will go high and form a new undesired path to Vdd, which can conduct to the loss of the saved internal state.
D. Improvement of C

To improve the previous described method the sleep transistor has to be split up and added in series with the clock stage to cut the possibility that the current can use this path in sleep mode. Another issue to solve is the activation of the second current source active when the cell is sleeping. This can be done by an inverted sleep signal, which is not a very nice solution, because adding an inverter inside every flip flop is too power and area consuming, but otherwise an additional signal to distribute will be added. Simulations of this gate shows that for a non ideal sleep signal with finite slopes the sleep transistor in the right path have to be faster than the other one in order to correctly save the internal state resulting in a width, which is about six to eight times larger than initially. This large area overhead can be saved, if different Vth’s are used. Using a low Vth for the right sleep transistor will make it faster, but will result in higher leakage. Fortunately this doesn’t matter, because current is needed in this branch anyway to save the state and therefore the second current source can be design a little bit smaller.
4.3 Comparison and choice

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power savings</td>
<td>additional power</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Functional</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Area</td>
<td>~39u</td>
<td>-</td>
<td>-</td>
<td>37.17u</td>
</tr>
<tr>
<td>Off-Current</td>
<td>~150u</td>
<td>-</td>
<td>-</td>
<td>57.7u</td>
</tr>
</tbody>
</table>

Table 16: Comparison of the different methods

Due to the fact that just two methods are actually working and just one of them can provide power savings, version D was chosen and implemented in the final test circuit described in the next chapter.

4.4 Sizing and simulation of one D-Latch

Figure 18: Schematics of the state saving D-Latch to implement
Different important observations can be made for the flip-flop simulations, which are annexed in chapter 9.C. The most important thing is the current variation in sleep mode. For the slow low Vth case, the current savings are more or less zero, while for most of the other cases, including the typical case, just about half of the current is consumed in sleep mode.

Also here the control voltage VN is saturated in the case of the slow low Vth transistors. To try if the problem is there, the limitation of VN to Vdd can be eliminated. In this case it can be observed, that VN goes up to about 1V and the current in sleep mode is even higher for a slow low Vth transistor.

This gate is working and also saving power during sleep modes, but the goal is not really reached, because there’s not that much power saved. In the worst case there’s just a power saving of about 2% with the big price of an additional inverted sleep signal and additional area to pay. In a further project new methods have to be explored to save this problem and to get a more efficient gate.
5 Test circuit

5.1 Description
To show the performance and the utility of MCML gates with integrated sleep mechanism a test circuit - very similar the LEON 3 processor[5] - has be drawn on the schematic level, simulated and analyzed. A circuit, where not all parts are active at the same time would be perfect to show the advantages of this logic style. Therefore a 32-bit arithmetic logic unit (ALU) has been chosen. To buffer the in- and outputs, flip-flops are insert. Using two control signals the choice between a barrel shifter, a logic block, an adder or a subtractor has to be done. More control signals are used to choose between the different possibilities per block. Depending on all this control signals an internal logic block will calculate the different sleep signals to activate just the blocks used actually or in the next clock cycle. When the whole ALU is not used, it can be shut down by an external sleep signal. Because it doesn’t make sense to save the input/output signals, normal flip-flops are used there, which don’t save their states. To show although the functionality of state saving flip-flops a program counter is also included, which will save the actual state in a program also after the whole circuit was shut down.

Figure 19: Block schematics of the test circuit
With a first two bit signal (called MUX_Ctrl<1:0> in the figure above) the output multiplexer is addressed to choose between the three main blocks. A second four bit control signal is used for different operation modes of the main blocks. The different modes of the logic block are described in table 20. The barrel shifter will just shift right by the Ctrl<3:0> number, while the adder block doesn’t need any control signals. There are two additional inputs, one to reset the program counter and one to set the whole circuit in sleep mode.

<table>
<thead>
<tr>
<th>Ctrl_Mux</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>Barrel shifter</td>
</tr>
<tr>
<td>0 1</td>
<td>Logic block</td>
</tr>
<tr>
<td>1 0</td>
<td>Adder</td>
</tr>
<tr>
<td>1 1</td>
<td>Subtractor</td>
</tr>
</tbody>
</table>

Table 19: Control signals for the multiplexer to choose between blocks

<table>
<thead>
<tr>
<th>Ctrl</th>
<th>Logic function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>XNOR</td>
</tr>
<tr>
<td>0 1</td>
<td>XOR</td>
</tr>
<tr>
<td>0 0</td>
<td>NOR</td>
</tr>
<tr>
<td>0 1</td>
<td>OR</td>
</tr>
<tr>
<td>1 0</td>
<td>NAND</td>
</tr>
<tr>
<td>1 0</td>
<td>AND</td>
</tr>
<tr>
<td>1 1</td>
<td>Not used</td>
</tr>
<tr>
<td>1 1</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Table 20: Different mode of operations for the logic block
5.2 Sleep logic

Three levels of sleep hierarchy are present in the circuit. The highest priority has the external sleep signal, which will shut down all blocks, including sleep logic and the input and output stages. Just the internal program counter will save his stage can’t be turned off completely. All other sleep signals are generated automatically inside the circuit. The first block will calculate the sleep signals for the three different main blocks and just activate the block really used for the operation. The lowest priority has the sleep logic inside the logic block, which will shut down the block internal logic and multiplexer gates not used. Actually the sleep logic (figure 21) inside these two blocks is exactly the same. Inside the logic block the sleep signals have just to be connected correctly (Barrel->XOR, Logic->OR, Adder->AND).
Because the data comes with one clock cycle delay, due to the register access, the blocks can get activated one clock cycle before operation and are therefore ready, when they are used. The control signals to switch the multiplexers are delayed by one clock cycle inside the sleep block, to be in time with the delayed data input.
5.3 Simulation results

Figure 22: Comparison of MCML current with and without sleep logic during operation mode

Figure 23: Current consumption of the main blocks compared to the circuit
Power-gated MOS current-mode logic
Michael Schwander, EPFL 2009

Table 21: Comparison with normal MCML

<table>
<thead>
<tr>
<th></th>
<th>MCML without Sleep</th>
<th>MCML with Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average(I_{on}) [A]</td>
<td>40.68m</td>
<td>24.4m</td>
</tr>
<tr>
<td>Average(I_{off}) [A]</td>
<td>40.68m</td>
<td>1.86m</td>
</tr>
<tr>
<td>Power on [W]</td>
<td>34.58m</td>
<td>20.74m</td>
</tr>
<tr>
<td>Power off [W]</td>
<td>34.58m</td>
<td>1.58m</td>
</tr>
</tbody>
</table>

Figure 24: The worst case delay of the circuit

Table 22: Sleep delays compared to the worst case delay

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst case delay</td>
<td>1.9 ns</td>
</tr>
<tr>
<td>Power on</td>
<td>0.6 ns</td>
</tr>
<tr>
<td>Power off</td>
<td>0.3 ns</td>
</tr>
</tbody>
</table>
5.4 Discussion

Comparing the normal MCML logic with the MCML logic with integrated sleep mechanism there is a huge gain in power consumption can be achieved in several applications. It’s evident, that the circuit has to offer inactive gates or even better, inactive blocks to be able to take profit from the sleep mechanism. As the turn on time is smaller than the critical path, the whole circuit can be turned off or on during one clock cycle and therefore whole blocks can be turned off, if the used blocks for the next cycle are known one cycle before.

The higher current consumption during one clock cycle is due to the fact that two blocks are active the same time. Because when a block will be used, it is activated one clock cycle before to guarantee the all cells are working, when they are used. Also therefore the same spikes are observable during the logic block is active, because it includes also different blocks, which will turned off and on.

With the ripple carry adder a clock of about 500 Mhz can be used. But with a faster adder design faster clocks can be used as far as the clock period is bigger than the power switching delay.
7 Conclusion

In this project it has been shown, that by adding a simple sleep transistor in series with the current source, a sleep mechanism can be introduced easily in MCML gates. To estimate the resulting area overhead some layouts are drawn. It can be observed, that the additional sleep transistor can be merged together with the current source and therefore for most of the cells not a big area overhead is resulting.

It seems that MCML including a sleep mechanism is very interesting for future applications, because depending on the application a lot of power can be saved compared to normal MCML logic. In this case for a small area overhead all advantages of MCML logic can be used, without consuming too much of constant power.

7.1 Further work

To be sure, that this library really works in the given conditions, all layouts have to be drawn and post layout simulations have to be done and evaluated. Also different models used during this project, in example for the amplifier, have to be replaced by real components.

Further the solution for state saving flip-flops is not satisfying, because the some problems occurring during corner simulations and also due to the inverted sleep signal needed additionally.

7.2 Acknowledgements

I would like to thank my advisors, Stéphane Badel and Alessandro Cevrero for their competent help, their good suggestions and ideas. Further I am really grateful for their patience explaining me features and helping me solving problems with the Cadence Design Kit and other tools used.
8 References

8.1 Bibliography

8.2 Software
[a] Cadence Design Kit, www.cadence.com

8.3 Abbreviations
- ALU Arithmetic Logic Unit
- CML Current-Mode Logic
- CMOS Complementary MOS
- MCML MOS Current-Mode Logic
- MOS Metal Oxide Semiconductor
- NM Noise Margin
- VCVS Voltage Controlled Voltage Source
- Vsw Voltage Swing
- Vth Threshold Voltage

8.3 List of figures
Figure 1: Principle of a MCML gate.................................................................8
Figure 2: Buffer schematic............................................................................9
Figure 3: AND gate .......................................................................................10
Figure 4: XOR gate (left) and MUX2to1 (right) .............................................10
Figure 5: MAJ32 gate (left) and XOR3 gate (right) .......................................11
Figure 6: Generation of VP and VN using the negative output of the reference gate...........12
Figure 7: Model of the amplifier................................................................12
Figure 8: Definition of the static noise margin .............................................13
Figure 9: Noise margins of a buffer cell ......................................................15
Figure 10: First (left) and second (right) sleep method ................................16
Figure 11: The bulk voltage control method (left) and the simple sleep transistor (right)........17
Figure 12: Sleep transistor merged into a buffer cell ....................................21
Figure 13: Comparison of current [μA] and NM [mV] of the different sleep mechanism ........................................22
Figure 14: Flip-Flop (left) and the schematics of a D-Latch (right) ..............................................................23
Figure 15: Saving the data by adding a multiplexer in front ............................................................................24
Figure 16: Adding a transistor to short the clock stage in sleep mode .........................................................25
Figure 17: Flip-Flop with second current source and the undesired additional path due to high clock inputs (arrow) ........................................................................................................................................26
Figure 18: Schematics of the state saving D-Latch to implement .................................................................27
Figure 19: Schematics of the test circuit ........................................................................................................29
Figure 20: Block schematics of the Logic block ........................................................................................31
Figure 21: Sleep logic to generate the sleep signals for the main blocks ...................................................32
Figure 22: Comparison of MCML current with and without sleep logic during operation mode ................33
Figure 23: Current consumption of the main blocks compared to the circuit ........................................33
Figure 24: The worst case delay of the circuit ...............................................................................................34

8.4 List of tables
Table 1: Vth used for different transistors ........................................................................................................9
Table 2: Specifications ........................................................................................................................................12
Table 3: Corner cases .......................................................................................................................................13
Table 4: Buffer size ..........................................................................................................................................14
Table 5: Simulation results of a buffer cell .....................................................................................................14
Table 6: Sizing of a bulk voltage controlled buffer cell ................................................................................18
Table 7: Simulation result of a bulk voltage controlled buffer cell .............................................................18
Table 8: Sizing of a bulk voltage controlled AND gate ................................................................................18
Table 9: Simulation results of a bulk voltage controlled AND gate .............................................................18
Table 10: Sizing of a buffer cell including a sleep transistor .........................................................................19
Table 11: Simulation results of a buffer cell with sleep transistor ...............................................................19
Table 12: Sizing of an AND gate with sleep transistor ..................................................................................20
Table 13: Simulation results of an AND gate with sleep transistor .............................................................20
Table 14: Estimated size and overhead of layout including sleep transistor ............................................21
Table 15: Comparison of the two methods ....................................................................................................22
Table 16: Comparison of the different methods ...........................................................................................27
Table 17: Sizing of a state saving D-Latch with sleep transistors ...............................................................28
Table 18: Simulation results of a state saving D-Latch (if not specified: ON-value) ..................................28
Table 19: Control signals for the multiplexer to choose between blocks ..................................................30
Table 20: Different mode of operations for the logic block .......................................................................30
Table 21: Comparison with normal MCML ....................................................................................................34
Table 22: Sleep delays compared to the worst case delay ...........................................................................34
9.1 Appendix A: Cell library

All schematics of the sleep transistor cell library and the also of the tested circuit are presented in this appendix:

Figure A. 1: Buffer schematics ................................................................. 40
Figure A. 2: AND schematics ................................................................. 41
Figure A. 3: OR schematics ................................................................. 42
Figure A. 4: XOR schematics ................................................................. 43
Figure A. 5: MUX2to1 schematics ........................................................ 44
Figure A. 6: MUX4to1 schematics ......................................................... 45
Figure A. 7: Majority3to2 schematics .................................................... 46
Figure A. 8: XOR3 schematics .............................................................. 47
Figure A. 9: D-Latch schematics .......................................................... 48
Figure A. 10: State saving D-Latch schematics ..................................... 49
Figure A. 11: Flip-Flop schematics ....................................................... 50
Figure A. 12: VP generation circuit schematics ...................................... 51
Figure A. 13: VN generation circuit schematics ...................................... 51
Figure A. 14: Test circuit schematics ..................................................... 52
Figure A. 15: Sleep control schematics ................................................ 53
Figure A. 16: Decoder schematics ........................................................ 54
Figure A. 17: Level restore schematics ................................................ 55
Figure A. 18: Programm counter schematics ........................................ 55
Figure A. 19: Logic block ..................................................................... 56
Figure A. 1: Buffer schematics
Figure A. 2: AND schematics
Figure A. 3: OR schematics
Figure A. 4: XOR schematics
Figure A. 5: MUX2to1 schematics
Figure A. 6: MUX4to1 schematics
Figure A. 7: Majority3to2 schematics
Figure A.8: XOR3 schematics
Figure A. 9: D-Latch schematics
Figure A. 10: State saving D-Latch schematics
Figure A. 11: Flip-Flop schematics
Figure A. 12: VP generation circuit schematics

Figure A. 13: VN generation circuit schematics
Figure A. 14: Test circuit schematics
Figure A.15: Sleep control schematics
Figure A.16: Decoder schematics
9.1 Appendix A: Cell library

Figure A. 18: Programm counter schematics

Figure A. 17: Level restore schematics
Figure A. 19: Logic block
9.2 Appendix B: Layouts

In this section, the test layouts drawn to estimate more or less the used area of the gates are represented.

Figure B. 1: Layout of a buffer cell ................................................................. 58
Figure B. 2: Layout of an AND gate ................................................................. 59
Figure B. 3: Layout of an OR gate ................................................................. 60
Figure B. 4: Layout of a 2to1 multiplexer ...................................................... 61
Figure B. 5: Layout of a state saving D-latch .............................................. 62
Figure B. 1: Layout of a buffer cell
Figure B. 2: Layout of an AND gate
Figure B. 3: Layout of an OR gate
Figure B. 4: Layout of a 2to1 multiplexer
Figure B. 5: Layout of a state saving D-latch
9.3 Appendix C: Figures

Figure C. 1: Current plot of a buffer ..................................64
Figure C. 2: Output levels of a buffer ..................................64
Figure C. 3: VN voltages of a buffer ...................................65
Figure C. 4: VP voltage of a buffer .......................................65
Figure C. 5: Noise margins of a buffer ...................................66
Figure C. 6: Current plot of a bulk voltage controlled buffer ..........66
Figure C. 7: Output levels of a bulk controlled buffer ..................67
Figure C. 8: VN voltage of a bulk controlled buffer .....................67
Figure C. 9: VP voltage of a bulk controlled buffer .....................68
Figure C. 10: Noise margins of a bulk controlled buffer .................68
Figure C. 11: Current plot of a bulk controlled AND gate with off times 69
Figure C. 12: Output levels of a bulk controlled AND gate with off times 69
Figure C. 13: VN voltage of a bulk controlled AND gate ................70
Figure C. 14: VP voltage of a bulk controlled AND gate .................70
Figure C. 15: Noise margins of a bulk controlled AND gate .............71
Figure C. 16: Current plot of a buffer cell with sleep transistor ........71
Figure C. 17: Output levels of a buffer cell with sleep transistor ..........72
Figure C. 18: VN voltage of a buffer cell with sleep transistor ...........72
Figure C. 19: VP voltage of a buffer cell with sleep transistor ...........73
Figure C. 20: Noise margins of a buffer cell with sleep transistor ........73
Figure C. 21: Current plot of a AND gate with sleep transistor ..........74
Figure C. 22: Output levels of a AND gate with sleep transistor ..........74
Figure C. 23: VN voltage of a AND gate with sleep transistor ..........75
Figure C. 24: VP voltage of a AND gate with sleep transistor ...........75
Figure C. 25: Noise margin of a AND gate with sleep transistor ........76
Figure C. 26: Current plot of the D-Latch with sleep transistor ...........76
Figure C. 27: Sleep signal of the D-Latch with sleep transistor ..........77
Figure C. 28: Input clock of the D-Latch with sleep transistor ..........77
Figure C. 29: Output levels of the D-Latch with sleep transistor ..........78
Figure C. 30: VN voltage of the D-Latch with sleep transistor ..........78
Figure C. 31: VP voltage of the D-Latch with sleep transistor ..........79
Figure C. 32: Changing time of the node with the biggest load ............79
Figure C. 33: Sleep signals for the main blocks ..........................80
Figure C. 34: Sleep signals of the main blocks (Zoom) ......................81
Current plot of a buffer cell

Output of a buffer cell
 VN of a buffer cell

Figure C. 3: VN voltages of a buffer

 VP of a buffer cell

Figure C. 4: VP voltage of a buffer
9.3 Appendix C: Figures

Figure C. 5: Noise margins of a buffer cell

Figure C. 6: Current plot of a bulk voltage controlled buffer cell
9.3 Appendix C: Figures

Figure C.7: Output levels of a bulk controlled buffer

Figure C.8: VN voltage of a bulk controlled buffer
9.3 Appendix C: Figures

Figure C. 9: VP voltage of a bulk controlled buffer

Figure C. 10: Noise margins of a bulk controlled buffer
Current plot of a bulk controlled AND gate

Figure C. 11: Current plot of a bulk controlled AND gate with off times

Output of a bulk controlled AND gate

Figure C. 12: Output levels of a bulk controlled AND gate with off times
Power-gated MOS current-mode logic
Michael Schwander, EPFL 2009

9.3 Appendix C: Figures

Figure C. 13: VN voltage of a bulk controlled AND gate

Figure C. 14: VP voltage of a bulk controlled AND gate
Figure C. 15: Noise margins of a bulk controlled AND gate

Figure C. 16: Current plot of a buffer cell with sleep transistor
Figure C. 17: Output levels of a buffer cell with sleep transistor

Figure C. 18: VN voltage of a buffer cell with sleep transistor
**Appendix C: Figures**

**Figure C. 19:** Vp voltage of a buffer cell with sleep transistor

**Figure C. 20:** Noise margins of a buffer cell with sleep transistor
Current plot of a AND gate with sleep transistor

Figure C. 21: Current plot of a AND gate with sleep transistor

Output level of a AND gate with sleep transistor

Figure C. 22: Output levels of a AND gate with sleep transistor
Figure C. 23: VN voltage of a AND gate with sleep transistor

Figure C. 24: VP voltage of a AND gate with sleep transistor
Power-gated MOS current-mode logic
Michael Schwander, EPFL 2009

9.3 Appendix C: Figures

76

Figure C. 25: Noixse margin of a AND gate with sleep transistor

Figure C. 26: Current plot of the D-Latch with sleep transistors
**Sleep signal**

Figure C. 27: Sleep signal of the D-Latch with sleep transistor

**Input clock of the D-Latch with sleep transistors**

Figure C. 28: Input clock of the D-Latch with sleep transistor
Output Q of a D-Latch with sleep transistors

Figure C. 29: Output levels of the D-Latch with sleep transistor

VN of a D-Latch with sleep transistors

Figure C. 30: VN voltage of the D-Latch with sleep transistor
Figure C. 31: VP voltage of the D-Latch with sleep transistor

Figure C. 32: Changing time of the node with the biggest load
Figure C. 33: Sleep signals for the main blocks
Figure C. 34: Sleep signals of the main blocks (Zoom)
9.4 Appendix D: Scripts

Script 1: Noise margin calculation for cadence .................................................................82
Script 2: PCF file for corner simulations with cadence .......................................................83
Script 3: VHDL file to generate pseudo random inputs (cont. next page) .........................83
Script 4: Script to save the VHDL outputs in ModelSim in a VCD file .............................83
Script 5: Matlab scrip to plot the noise margin boxes .........................................................83
Script 6: Matlab script to plot the comparison plots ..........................................................83
Script 7: Matlab file to generate the plots for the standard cells .......................................83
Script 8: Matlab script to generate the bulk controlled plots ...........................................83
Script 9: Matlab script to generate the plots for the sleep transistor cells ........................83
Script 10: Matlab script to plot the sleep signals for the test circuit ................................83
Script 11: Matlab script to generate the different current plots for the test circuit ............83
Script 12: Matlab script to genearte the worst case delay plot .......................................83
Script 13: Matlab script to generate the plot for the node with the biggest load ..............83

\[ \text{NM}(\text{waveVsWave}(\text{?y (VS("/Vo+") \text{– VS("/Vo-")}) \text{?x (VS("/IN+") \text{– VS("/IN-")}))}) ) \]

Script 1: Noise margin calculation for cadence
-- randomInputGenerator.vhd
-- generates up to 2^NBITS random signal values
-- with the propriety that the same value is never repeated.
--
-- Michael Schwander

library ieee;
use ieee.std_logic_1164.all;
use ieee.math_real.all;
use ieee.numeric_std.all;
use work.lfsr_pkg.all;

entity lfsr is
  generic (
    NBITS : positive := 32
  );
  port (
    clk   : in std_logic;
    rst_b : in std_logic;
    seed  : in std_logic_vector (NBITS-1 downto 0);
    dout  : out std_logic_vector (NBITS-1 downto 0)
  );
end entity lfsr;

architecture o2m of lfsr is
begin
  REG: process (clk, rst_b)
  begin
    if rst_b = '0' then
      state_reg <= seed;
    elsif clk'event and clk='1' then
      state_reg <= state_next;
    end if;
  end process REG;

  LFSR: process (state_reg)
  begin
    feedback := state_reg(NBITS-1);
    for i in NBITS-1 downto 1 loop
      if TAPS(i-1) = '1' then
        -- bit shifts and xors at this index (tap point)
        state_next(i) <= state_reg(i-1) xor feedback;
      else
        state_next(i) <= state_reg(i-1);
      end if;
    end loop;
    state_next(0) <= feedback;
  end process LFSR;

  OUTPUT: dout <= state_reg;
end architecture o2m;

library ieee;
use ieee.std_logic_1164.all;
use ieee.math_real.all;
use ieee.numeric_std.all;

entity randomInputGenerator is
  generic ( 
    NBITS   : positive := 32;
    NOUTPUT : positive := 10**5;
    CLKPERIOD : time   := 20 ns 
  );
end entity randomInputGenerator;

architecture bench of randomInputGenerator is
begin
  LFSR1: entity work.lfsr(o2m)
    generic map ( 
      NBITS => NBITS 
    )
    port map ( 
      clk   => clk,
      rst_b => rst_b,
      dout  => tinap,
      seed  => seeda 
    );

  LFSR2: entity work.lfsr(o2m)
    generic map ( 
      NBITS => NBITS 
    )
    port map ( 
      clk   => clk,
      rst_b => rst_b,
      dout  => tinbp,
      seed  => seedb 
    );

end architecture bench;

library ieee;
use ieee.std_logic_1164.all;
use ieee.math_real.all;
use ieee.numeric_std.all;

entity randomInputGenerator is
  generic ( 
    NBITS   : positive := 32;
    NOUTPUT : positive := 10**5;
    CLKPERIOD : time   := 20 ns 
  );
end entity randomInputGenerator;

architecture bench of randomInputGenerator is
begin
  LFSR1: entity work.lfsr(o2m)
    generic map ( 
      NBITS => NBITS 
    )
    port map ( 
      clk   => clk,
      rst_b => rst_b,
      dout  => tinap,
      seed  => seeda 
    );

  LFSR2: entity work.lfsr(o2m)
    generic map ( 
      NBITS => NBITS 
    )
    port map ( 
      clk   => clk,
      rst_b => rst_b,
      dout  => tinbp,
      seed  => seedb 
    );

end architecture bench;
CLKSTIM: clk <= not clk after CLKPERIOD/2;

STIMULI: process
begin
-- set seeds
seeda <= std_logic_vector(to_unsigned(1, NBITS));
seedb <= std_logic_vector(to_unsigned(2**16-1, NBITS));

-- reset LFSRs
wait for 5 ns;
rst_b <= '0';
wait for 5 ns;
rst_b <= '1';
wait;
end process STIMULI;

RAND: process (tinap, tinbp)
procedure check (rand1, rand2 : in std_logic_vector(NBITS-1 downto 0)) is
begin
assert (to_integer(signed(rand1)) /= to_integer(signed(rand2)))
report LF & "Random numbers are equal for : " & integer'image(to_integer(signed(rand1)))
severity warning;
end procedure check;
begin
-- generate random signed numbers
check (tinap, tinbp);
randa <= to_integer (signed(tinap));
randb <= to_integer (signed(tinbp));
end process;

NEG: process (tinap, tinbp)
begin
  tinan <= not tinap;
  tinbn <= not tinbp;
end process;
end architecture bench;

-- vcdcreator.do
vcom randomInputGenerator.vhd

vsim -sdfnoerror -t ns work.randomInputGenerator
vcd file vectors.vcd
vcd add tinap
vcd add tinbp
vcd add tinan
vcd add tinbn
run 1ms
quit

Script 4: Script to save the VHDL outputs in ModelSim in a VCD file
% Matlab script
% 26.05.2009
% Michael Schwander, EPFL
% Figure generated by Matlab

% Buffer NM:
y1=[66.75; 171.6; 86.24; 51.31; 80.31];

% Buffer bulk voltage controlled NM:
y1=[92.61; 172.4; 62.06; 52.05; 167.6];

% AND bulk voltage controlled NM:
y1=[68.31; 104; 120.1; 61.52; 96.35];

% Buffer with ST NM:
y1=[68.94; 136; 85.1; 53.73; 86.39];

% AND with ST NM:
y1=[76.44; 174.5; 79.52; 58.99; 95.84];

%% Create figure
figure1 = figure('PaperPosition',[0.6345 6.345 20.3 15.23],'PaperSize',[20.98 29.68]);

%% Create axes
axes1 = axes(...
    'XTick',[1 2 3 4 5],...
    'XTickLabel',{'','','','',''},...
    'Parent',figure1);

%title(axes1,'Noise margins of a buffer cell','FontSize',18);
title(axes1,'Noise margins of an AND gate','FontSize',18);
xlabel(axes1,'typical      lvt slow      lvt fast      hvt slow      hvt fast','FontSize',14);
ylabel(axes1,'NM [mV]','FontSize',14);
hold(axes1,'all');

%% Create bar
bar1 = bar(y1);

Script 5: Matlab script to plot the noise margin boxes
% Matlab script for comparisons
% 26.05.2009
% Michael Schwander, EPFL
% Figure generated by Matlab

y1=[49.24 47.73 49.18; 51.31 52.05 53.73; 1 43.04 47.7; 1 62.34 58.99];

% Create figure
figure1 = figure('PaperPosition',[0.6345 6.345 20.3 15.23],'PaperSize',[20.98 29.68]);

% Create axes
axes1 = axes(...
    'FontSize',12,...
    'XTick',[1 2 3 4],...
    'XTickLabel',{'I min','NM min','I min','NM min'},...
    'Parent',figure1);
xlabel(axes1,'Buffer AND');
hold(axes1,'all');

% Create bar
bar1 = bar(y1,...
     'Parent',axes1,...
     'DisplayName','normal');

% Create bar
bar2 = bar(y2,...
    'Parent',axes1,...
    'DisplayName','bulk controlled');

% Create bar
bar3 = bar(y3,...
    'Parent',axes1,...
    'DisplayName','sleep transistor');

% Create legend
legend1 = legend(...
    axes1,{'normal','bulk controlled','sleep transistor'},...
    'FontSize',12,...
    'Position',[0.3392 0.7972 0.3339 0.1873]);

Script 6: Matlab script to plot the comparison plots
% File to load and plot *.csv files
% Michael Schwander, EPFL 2009

filename1 = 'Simulations/I_buffer.csv';
filename2 = 'Simulations/Buffer_all.csv';

% read file to matrix and skip the 15 first lines (due to different DC solutions)
M2 = csvread(filename2, 15, 0);

% Plot settings
VT1={'hvt slow' 'hvt fast' 'lvt fast' 'lvt slow' 'all typ'};
VT2={'hvt slow' 'hvt fast' 'all typ' 'lvt fast' 'lvt slow'};
CONF={'r' 'g' 'b' 'y' 'm'};
WIDTH1={1 1 1 2};
WIDTH2={1 2 1 1};
TITEL={'Input' 'VN of a buffer cell' 'VP of a buffer cell' 'Output of a buffer cell'};
Y={'Input [mV]' 'VN [mV]' 'VP [mV]' 'Input [mV]'};

% Current
hold on;
for i=1:5
    plot(M1(:,1)*10^6,M1(:,i+1)*10^6,CONF{i},'LineWidth',WIDTH1{i});
end;
title('Current plot of a buffer cell','FontSize',18);
xlabel('time [us]','FontSize',14), ylabel('I [uA]','FontSize',14);
legend(VT1);
hold off;
for j=1:4
    figure;
    hold on;
    for i=1:5
        plot(M2(:,1)*10^6,M2(:,j+i*4-3)*10^3,CONF{i},'LineWidth',WIDTH2{i});
    end;
title(TITEL{j},'FontSize',18);
xlabel('time [us]','FontSize',14), ylabel(Y{j},'FontSize',14);
legend(VT2);
hold off;
end;

Script 7: Matlab file to generate the plots for the standard cells
% File to load and plot *.csv files
% Michael Schwander, EPFL 2009
filename1 = 'Simulations/I_VTC_Buffer.csv';
filename2 = 'Simulations/VTC_Buffer_all.csv';
filename3 = 'Simulations/I_VTC_AND.csv';
filename4 = 'Simulations/VTC_AND_all.csv';

% read file to matrix and skip the 15 first lines (due to different DC solutions)
M1 = csvread(filename1, 15, 0);
M2 = csvread(filename2, 15, 0);
M3 = csvread(filename3, 15, 0);
M4 = csvread(filename4, 15, 0);

% Plot settings
VT1={'hvt slow' 'hvt fast' 'lvt fast' 'lvt slow' 'all typ'};
VT2={'hvt slow' 'hvt fast' 'all typ' 'lvt fast' 'lvt slow'};
CONF={'r-':'g-':'b-':'y-':'m-'};
WIDTH1=[1 1 1 1 2];
WIDTH2=[1 1 2 1 1];
TITEL1={'Input' 'VN of a bulk controlled buffer cell' 'VP of a bulk controlled buffer cell' 'Output of a bulk controlled buffer cell'};
TITEL2={'Input A' 'Input B' 'VN of a bulk controlled AND gate' 'VP of a bulk controlled AND gate' 'Output of a bulk controlled AND gate'};
Y1={'Input [mV]' 'VN [mV]' 'VP [mV]' 'Input [mV]'};
Y2={'Input A [mV]' 'Input B [mV]' 'VN [mV]' 'VP [mV]' 'Input [mV]'};

% Plots
hold on;
for i=1:5
    plot(M1(:,1)*10^6,M1(:,i+1)*10^6,CONF{i},'LineWidth',WIDTH1{i});
end;
title('Current plot of a bulk controlled buffer cell','FontSize',18);
xlabel('time [us]','FontSize',14), ylabel('I [uA]','FontSize',14);
legend(VT1);
hold off;

for j=1:4
    figure;
    hold on;
    for i=1:5
        plot(M2(:,1)*10^6,M2(:,j+i*4-3)*10^3,CONF{i},'LineWidth',WIDTH2{i});
    end;
title(TITEL1{j},'FontSize',18);
xlabel('time [us]','FontSize',14), ylabel(Y1{j},'FontSize',14);
legend(VT2);
hold off;
end;

figure;
hold on;
for i=1:5
    plot(M3(:,1)*10^6,M3(:,i+1)*10^6,CONF{i},'LineWidth',WIDTH1{i});
end;
title('Current plot of a bulk controlled AND gate','FontSize',18);
xlabel('time [us]','FontSize',14), ylabel('I [uA]','FontSize',14);
legend(VT1);
hold off;

for j=1:5
    figure;
    hold on;
    size(M4)
    for i=1:5
        plot(M4(:,1)*10^6,M4(:,j+i*5-4)*10^3,CONF{i},'LineWidth',WIDTH2{i});
    end;
title(TITEL2{j},'FontSize',18);
xlabel('time [us]','FontSize',14), ylabel(Y2{j},'FontSize',14);
legend(VT2);
hold off;
end;
% File to load and plot *.csv files
% Michael Schwander, EPFL 2009
filename1 = 'Simulations/I_ST_Buffer.csv';
filename2 = 'Simulations/ST_Buffer_all.csv';
filename3 = 'Simulations/I_ST_AND.csv';
filename4 = 'Simulations/ST_AND_all.csv';
filename5 = 'Simulations/I_ST_D.csv';
filename6 = 'Simulations/ST_D_all.csv';

% read file to matrix and skip the 15 first lines (due to different DC % solutions)
M1 = csvread(filename1, 15, 0);
M2 = csvread(filename2, 15, 0);
M3 = csvread(filename3, 15, 0);
M4 = csvread(filename4, 15, 0);
M5 = csvread(filename5, 15, 0);
M6 = csvread(filename6, 15, 0);

% Plot settings
VT1={'hvt slow' 'hvt fast' 'lvt fast' 'lvt slow' 'all typ'};
VT2={'hvt slow' 'hvt fast' 'all typ' 'lvt fast' 'lvt slow'};
CONF={'r' 'g' 'b' 'y' 'm'};
WIDTH1={1 1 1 1 2};
WIDTH2={1 1 1 1 1};
TITEL1={'Input' 'VN of a buffer cell with sleep transistor' 'VP of a buffer cell with sleep transistor' 'Output of a buffer cell with sleep transistor'};
TITEL2={'Input A' 'Input B' 'VN of a AND gate with sleep transistor' 'VP of a AND gate with sleep transistor' 'VP of a AND gate with sleep transistor'};
TITEL3={'Input Clk' 'Input D' 'Output Q of a D-Latch with sleep transistors' 'Sleep signal' 'VN of a D-Latch with sleep transistors' 'VP of a D-Latch with sleep transistors'};
Y1={'Input [mV]' 'VN [mV]' 'VP [mV]' 'Input [mV]'};
Y2={'Input A [mV]' 'Input B [mV]' 'VN [mV]' 'VP [mV]' 'Input [mV]'};
Y3={'Input Clk [mV]' 'Input D [mV]' 'Output Q [mV]' 'Sleep signal [mV]' 'VN [mV]' 'VP [mV]' 'Input Clk [mV]'};

% Plots
hold on;
for i=1:5
plot(M1(:,1)*10^6,M1(:,i+1)*10^6,CONF{i},'LineWidth',WIDTH1{i});
extend;
title('Current plot of a buffer cell with sleep transistor','FontSize',18);
xlabel('time [us]','FontSize',14), ylabel('I [uA]','FontSize',14);
legend(VT1);
hold off;
for j=1:4
figure;
hold on;
for i=1:5
plot(M2(:,1)*10^6,M2(:,j+i*4-3)*10^3,CONF{i},'LineWidth',WIDTH2{i});
extend;
title(TITEL1{j}, 'FontSize',18);
xlabel('time [us]','FontSize',14), ylabel(Y1{j},'FontSize',14);
legend(VT2);
hold off;
end;
figure;
hold on;
for i=1:5
plot(M3(:,1)*10^6,M3(:,i+1)*10^6,CONF{i},'LineWidth',WIDTH1{i});
extend;
title('Current plot of a AND gate with sleep transistor','FontSize',18);
xlabel('time [us]','FontSize',14), ylabel(Y1{i},'FontSize',14);
legend(VT1);
hold off;
end;
...

...
... 
for j=1:5 
if (j == 3) 
else 
figure; 
hold on; 
for i=1:5 
plot(M4(:,1)*10^6,M4(:,j+i*5-4)*10^3,CONF{i},'LineWidth',WIDTH2{i}); 
end; 
title(TITEL2{j},'FontSize',18); 
xlabel('time [us]','FontSize',14), ylabel(Y2{j},'FontSize',14); 
legend(VT2); 
hold off; 
end; 
end; 
figure; 
hold on; 
for i=1:5 
plot(M5(:,1)*10^6,M5(:,i+1)*10^6,CONF{i},'LineWidth',WIDTH1{i}); 
end; 
title('Current plot of a D-Latch with sleep transistors','FontSize',18); 
xlabel('time [us]','FontSize',14), ylabel('I [uA]','FontSize',14); 
legend(VT1); 
hold off; 
end; 
for j=1:7 
figure; 
hold on; 
for i=1:5 
plot(M6(:,1)*10^6,M6(:,j+i*7-6)*10^3,CONF{i},'LineWidth',WIDTH2{i}); 
end; 
title(TITEL3{j},'FontSize',18); 
xlabel('time [us]','FontSize',14), ylabel(Y3{j},'FontSize',14); 
legend(VT2); 
hold off; 
end; 
end;

Script 9: Matlab script to generate the plots for the sleep transistor cells
% File to load and plot *.csv files
%
% Michael Schwander, EPFL 2009
filename1 = 'Simulations/Circuit_Sleep/Adder_Barrel.Logic.csv';
filename2 = 'Simulations/Circuit_Sleep/Sleep.csv';
filename3 = 'Simulations/Circuit_Sleep/Clk.csv';

% read file to matrix and skip the 15 first lines (due to different DC
% solutions)
M1 = csvread(filename1, 15, 0);
M2 = csvread(filename2, 15, 0);
M3 = csvread(filename3, 15, 0);

% Plot settings
CONF={'r-','g-','b-'};
LEGEND={'Adder' 'Barrel' 'Logic'}

% Plot
subplot(3,1,1);
plot(M2(:,1)*10^6,M2(:,2)*10^3,CONF{1},'LineWidth',1);
title('Sleep signal of the circuit','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Sleep signal [mV]','FontSize',12);
%xlim([1.9 2.1]);
subplot(3,1,2);
hold on;
for i=1:3
plot(M1(:,1)*10^6,M1(:,i+1)*10^3,CONF{i},'LineWidth',1);
end;
legend(LEGEND);
title('Sleep signals of the main blocks','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Sleep signal [mV]','FontSize',12);
%xlim([1.9 2.1]);
hold off;
subplot(3,1,3);
plot(M3(:,1)*10^6,M3(:,2)*10^3,CONF{1},'LineWidth',1);
title('Clk of the circuit','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Clk [mV]','FontSize',12);
%xlim([1.9 2.1]);

Script 10: Matlab script to plot the sleep signals for the test circuit
% File to load and plot *.csv files
% Michael Schwander, EPFL 2009
filename1 = 'Simulations/Circuit_Sleep/I_circuit.csv';
filename2 = 'Simulations/Circuit_Sleep/I_Adder.csv';
filename3 = 'Simulations/Circuit_Sleep/I_Barrel.csv';
filename4 = 'Simulations/Circuit_Sleep/I_Logic.csv';

% read file to matrix and skip the 15 first lines (due to different DC solutions)
M1 = csvread(filename1, 15, 0);
M2 = csvread(filename2, 15, 0);
M3 = csvread(filename3, 15, 0);
M4 = csvread(filename4, 15, 0);

% Plot settings
CONF={'r-','g-','b-'};
LEGEND={'Adder','Barrel','Logic'};

% Plot
subplot(2,1,1);
plot(M1(:,1)*10^6,M1(:,2)*10^3,CONF{1},'LineWidth',1);
title('Current consumption of the circuit','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Current [mA]','FontSize',12);
%xlim([1.9 2.1]);
subplot(2,1,2);
hold on;
plot(M2(:,1)*10^6,M2(:,2)*10^3,CONF{1},'LineWidth',1);
plot(M3(:,1)*10^6,M3(:,2)*10^3,CONF{2},'LineWidth',1);
plot(M4(:,1)*10^6,M4(:,2)*10^3,CONF{3},'LineWidth',1);
legend(LEGEND);
title('Current consumption of the main blocks','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Current [mA]','FontSize',12);
%xlim([1.9 2.1]);
hold off;

Script 11: Matlab script to generate the different current plots for the test circuit
% File to load and plot *.csv files
% Michael Schwander, EPFL 2009
filename1 = 'Simulations/Circuit_Sleep/Adder_delay.csv';

% read file to matrix and skip the 15 first lines (due to different DC
% solutions)
M1 = csvread(filename1, 15, 0);

% Plot settings
VT={'hvt slow' 'hvt fast' 'all typ' 'lvt fast' 'lvt slow'};
CONF1={'r-' 'g-' 'b-' 'y-' 'm-'};
CONF2={'r--' 'g--' 'b--' 'y--' 'm--'};
WIDTH={1 1 2 1 1};

% Current
hold on;
xlim([0.097 0.104]);
ylim([250 900]);
for i=1:5
    plot(M1(:,1)*10^6,M1(:,2*i)*10^3,CONF1{i},'LineWidth',WIDTH{i});
end;
legend(VT);
for i=1:5
    plot(M1(:,1)*10^6,M1(:,2*i-1)*10^3,CONF2{i},'LineWidth',WIDTH{i});
end;
title('Worst case delay of the RCA','FontSize',18);
xlabel('time [us]','FontSize',14), ylabel('[mV]','FontSize',14);
hold off;

Script 12: Matlab script to generate the worst case delay plot
% File to load and plot *.csv files
%
% Michael Schwander, EPFL 2009
filename1 = 'Simulations/Circuit_Sleep/Internal_A0-5.csv';
filename2 = 'Simulations/Circuit_Sleep/Sleep.csv';

% read file to matrix and skip the 15 first lines (due to different DC
% solutions)
M1 = csvread(filename1, 15, 0);
M2 = csvread(filename2, 15, 0);

% Plot settings
CONF={'r-' 'g-' 'b-' 'y-' 'm-' 'b-'};
LEGEND={'A<0>' 'A<1>' 'A<2>' 'A<3>' 'A<4>' 'A<5>'};

% Plot
subplot(2,1,1);
plot(M2(:,1)*10^6,M2(:,2)*10^3,CONF{1},'LineWidth',1);
title('Sleep signal of the circuit','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Sleep [mV]','FontSize',12);
xlim([1.98 2.02]);

subplot(2,1,2);
hold on;
for i=1:6
plot(M1(:,1)*10^6,M1(:,1+i)*10^3,CONF{i},'LineWidth',1);
end;
legend(LEGEND);
title('Internal node with the biggest load','FontSize',14);
xlabel('time [us]','FontSize',12), ylabel('Voltage [mV]','FontSize',12);
xlim([1.995 2.005]);
hold off;

Script 13: Matlab script to generate the plot for the node with the biggest load