Homework 2 Fall 2017  TA: Joseph Aymond

Problem 1:
In figure 1.4 we can find the count of transistors in the processor increases by about ~10 times per seven years. In 2002 the count was about 0.1 Billion, so we can predict the count in 2020 to be about

\[ 0.1 \text{ billion} \times 10^{(18/7)} = 37.28 \text{ billion}. \]

Problem 2:
The transistor level schematic for a CMOS 3-input NAND gate can be made like this (other ways are possible)

\[ \text{Out} = A \cdot B \cdot \overline{C} \]

Problem 3:
Area of die = \( A_{dir} = (3 \text{mm})^2 = 9 \text{mm}^2 = 0.09 \text{cm}^2 \)

Area of wafer = \( A_{wafer} = \left(\frac{300}{2}\right)^2 \cdot \pi = 70685 \text{mm}^2 = 706.85 \text{ cm}^2 \)

Hard yield of die \( Y_H = e^{-Adie \cdot d} = e^{-0.09 \cdot 1.3} = e^{-0.117} = 0.890 \)

The total yield \( Y = Y_H \cdot Y_S = 0.890 \cdot 1 = 0.890 \)

The cost per good die \( C_{good} = \frac{C_{wafer}}{A_{wafer}/A_{die}} \cdot \frac{1}{Y} = \frac{\$3200}{\frac{70685}{9}} \cdot \frac{1}{0.890} = \$0.458 \)

Problem 4:
\[ Y = \frac{x - u}{g} = \frac{4mv - 0mv}{2mv} = \frac{4}{2} = 2 \]

\[ P_{\text{Soft,Amp}} = \int_{-2}^{2} f(x)dx = 2F_N(2) - 1 = 0.9545 \]

Four Op Amps per chip \( P_{\text{Soft,chip}} = 0.9545^4 = 0.830 \)
Problem 5:
Since the input offset voltage follow Gaussian distribution we have the soft yield \( \theta \):

\[
Y_s = P_{soft} = 2F_N(y) - 1 = 0.95 \Rightarrow F_N(y) = 0.975
\]
\[
\therefore y = 1.96 = \frac{x - u}{g} = \frac{2mv - 0}{(A_{VTO}/\sqrt{A})} \Rightarrow A = 600\mu m^2
\]

Problem 6:
The area of a die in new process = \( A_{die} = \left( \frac{7}{65} \right)^2 \times 0.5 cm^2 = 5.7988 \times 10^{-3} cm^2 \)
The hard yield = \( Y_H = e^{-A_{die}d} = e^{-0.00725} = 0.9928 \)
The yield \( Y = Y_H \times Y_s = 0.9928 \)

\[
C_{good} = \frac{C_{wafer}}{(A_{wafer}/A_{die})} \times \frac{1}{Y} = \frac{\$8000 \times 1}{1590.4 \times 0.9928} = \$0.0294
\]

Problem 7:
This switch level model assumes \( A = 0 \) and \( C = 0 \).
Problem 8:

When both inputs are 0V the output is VDD = 3V
When one input is 0V and the other is 3V the output is VDD = 3V
When both inputs are 3V the output is VSS = 0V

Problem 9:

a- The master is addressing a slave and requesting data. Data is transferred to the master.
b- The master is sending data to slave and receiving acknowledgement again just like part a.
c- Open drain refers to the exposed drain of a transistor typically used as the output.
d- Logic contention refers to data being driven at any time from several devices. I2C follows a protocol so that the contention doesn’t appear.
e- Due to the I2C open drain protocols the bus remains on a low threshold voltage. Devices can drive past the threshold so that the different logic levels are apparent.

Problem 10:

Here is one solution to this problem

```vhdl
<table>
<thead>
<tr>
<th>Ln#</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>`timescale 1ns/1ps</td>
</tr>
<tr>
<td>2</td>
<td>module NOR2 (i_A, i_B, o_F);</td>
</tr>
<tr>
<td>3</td>
<td>input i_A, i_B;</td>
</tr>
<tr>
<td>4</td>
<td>output o_F;</td>
</tr>
<tr>
<td>5</td>
<td>assign o_F = !(i_A</td>
</tr>
<tr>
<td>6</td>
<td>endmodule</td>
</tr>
</tbody>
</table>

//

<table>
<thead>
<tr>
<th>Ln#</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>`timescale 1ns/1ps</td>
</tr>
<tr>
<td>2</td>
<td>module OR2 (i_A, i_B, o_F);</td>
</tr>
<tr>
<td>3</td>
<td>input i_A, i_B;</td>
</tr>
<tr>
<td>4</td>
<td>output o_F;</td>
</tr>
<tr>
<td>5</td>
<td>wire A_nor_B;</td>
</tr>
<tr>
<td>6</td>
<td>NOR2 nor0(i_A(i_A), !i_B(i_B), .o_F(!A_nor_B));</td>
</tr>
<tr>
<td>7</td>
<td>NOR2 nor1(i_A(!A_nor_B), i_B(!A_nor_B), .o_F(!o_F));</td>
</tr>
<tr>
<td>8</td>
<td>endmodule</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
```
module OR3 (i_A, i_B, i_C, o_F);
    input i_A, i_B, i_C;
    output o_F;
    wire A_or_B;
    OR2 nor0(i_A, i_B, o_F(A_or_B));
    OR2 nor1(i_A, i_C, o_F(A_or_B));
endmodule

module AND2 (i_A, i_B, o_F);
    input i_A, i_B;
    output o_F;
    wire A_not, B_not;
    NOR2 nor0(i_A, i_B, o_F(A_not));
    NOR2 nor1(i_A, i_B, o_F(B_not));
    NOR2 nor2(i_A, i_B, o_F(A_not));
endmodule

module AND3 (i_A, i_B, i_C, o_F);
    input i_A, i_B, i_C;
    output o_F;
    wire A_and_B;
    AND2 and0(i_A, i_B, o_F(A_and_B));
    AND2 and1(i_A, i_B, o_F(A_and_B));
endmodule
module Function (i_A, i_B, i_C, o_F);
input i_A, i_B, i_C;
output o_F;
wire w_nA, w_nB, w_nC;
wire w_A1, w_A2, w_A3;
NOR2 nor0(i_A[i_A], i_B[i_B], o_F[w_nA]);
NOR2 nor1(i_A[i_B], i_B[i_B], o_F[w_nB]);
NOR2 nor2(i_A[i_C], i_B[i_C], o_F[w_nC]);
AND3 and1(i_A[w_nA], i_B[w_nB], i_C[w_nC], o_F[w_A1]);
AND3 and2(i_A[w_nA], i_B[w_nB], i_C[w_nC], o_F[w_A2]);
AND3 and3(i_A[w_nA], i_B[w_nB], i_C[w_nC], o_F[w_A3]);
OR3 or0(i_A[w_A1], i_B[w_A2], i_C[w_A3], o_F[o_F]);
endmodule

module Function_ME();
reg i_A, i_B, i_C;
wire w_F;
initial
begin
  r_A = 1'b0;
  r_B = 1'b0;
  r_C = 1'b0;
end
always
  #10 r_C = r_C;
always
  #20 r_B = r_B;
always
  #40 r_A = r_A;
Function U(i_A(r_A), i_B(r_B), i_C(r_C), o_F(w_F));
endmodule