Homework 2 Fall 2017 TA: Joseph Aymond

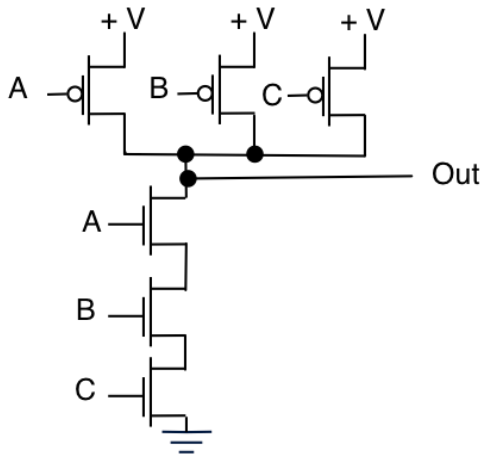
Problem 1:

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 billion * 10^(18/7) = 37.28 billion.

Problem 2:

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



A, B, and C are inputs

Out is the output

Out = $\overline{A * B * C}$

Problem 3:

Area of die = $A_{dir} = (3mm)^2 = 9mm^2 = 0.09cm^2$

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

Hard yield of die $Y_H = e^{-Adie*d} = e^{-.09*1.3} = e^{-0.117} = 0.890$

The total yield $Y = Y_H * Y_S = 0.890 * 1 = 0.890$

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

Problem 4:

$$Y = \frac{x-u}{g} = \frac{4mv - 0mv}{2mv} = \frac{4}{2} = 2$$

$P_{Soft\_Amp} = \int_{-2}^{2} f(x)dx = 2F_N(2) - 1 =$ 0.9545

Four Op Amps per chip $P_{Soft\_chip} = 0.9545^{\,4} =$ 0.830

Problem 5:

Since the input offset voltage follow Gaussian distribution we have the soft yield θ:

$Y_s = P_{soft} = 2F_N(y) - 1 = 0.95 => F_N(y) = 0.975$

$\therefore y = 1.96 = \frac{x-u}{g} = \frac{2mv-0}{\left(\frac{A_{VTO}}{\sqrt{A}}\right)} => A = 600\mu m^2$

Problem 6:

The area of a die in new process $= A_{die} = \left(\frac{7}{65}\right)^2 * 0.5cm^2 = 5.7988 * 10^{-3}cm^2$

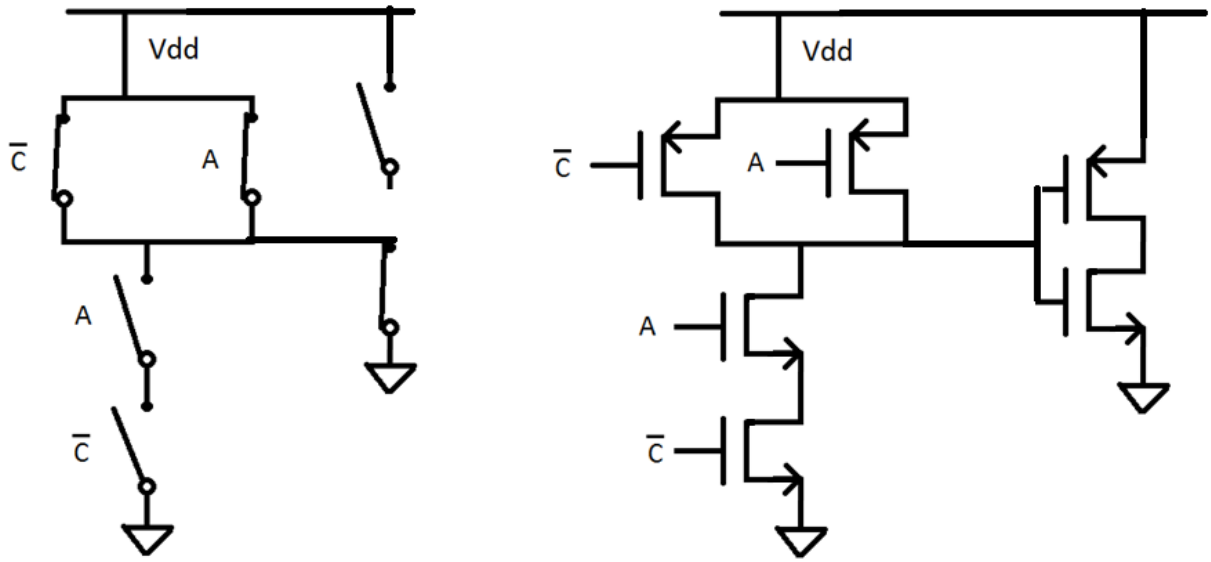
The hard yield = $Y_H = e^{-A_{die}*d} = e^{-0.00725} = 0.9928$

The yield $Y = Y_H * Y_S = 0.9928$

$C_{good} = \frac{C_{wafer}}{(A_{wafer}/A_{die})} * \frac{1}{Y} = \frac{\$8000}{\frac{1590.4}{0.005799}} * \frac{1}{0.9928} = \$0.0294$

Problem 7:

This switch level model assumes A =0 and $\bar{C}$ = 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

```
Ln#
 1  `timescale 1ns/1ps
 2  module NOR2 (i_A, i_B, o_F);
 3    input i_A, i_B;
 4    output o_F;
 5
 6    assign o_F = ~(i_A||i_B);
 7
 8  endmodule
 9
```

```
Ln#
 1
 2  `timescale 1ns/1ps
 3  module OR2 (i_A, i_B, o_F);
 4    input i_A, i_B;
 5    output o_F;
 6    wire A_nor_B;
 7
 8    NOR2 nor0(.i_A(i_A), .i_B(i_B), .o_F(A_nor_B));
 9    NOR2 nor1(.i_A(A_nor_B), .i_B(A_nor_B), .o_F(o_F));
10
11  endmodule
12
13
```

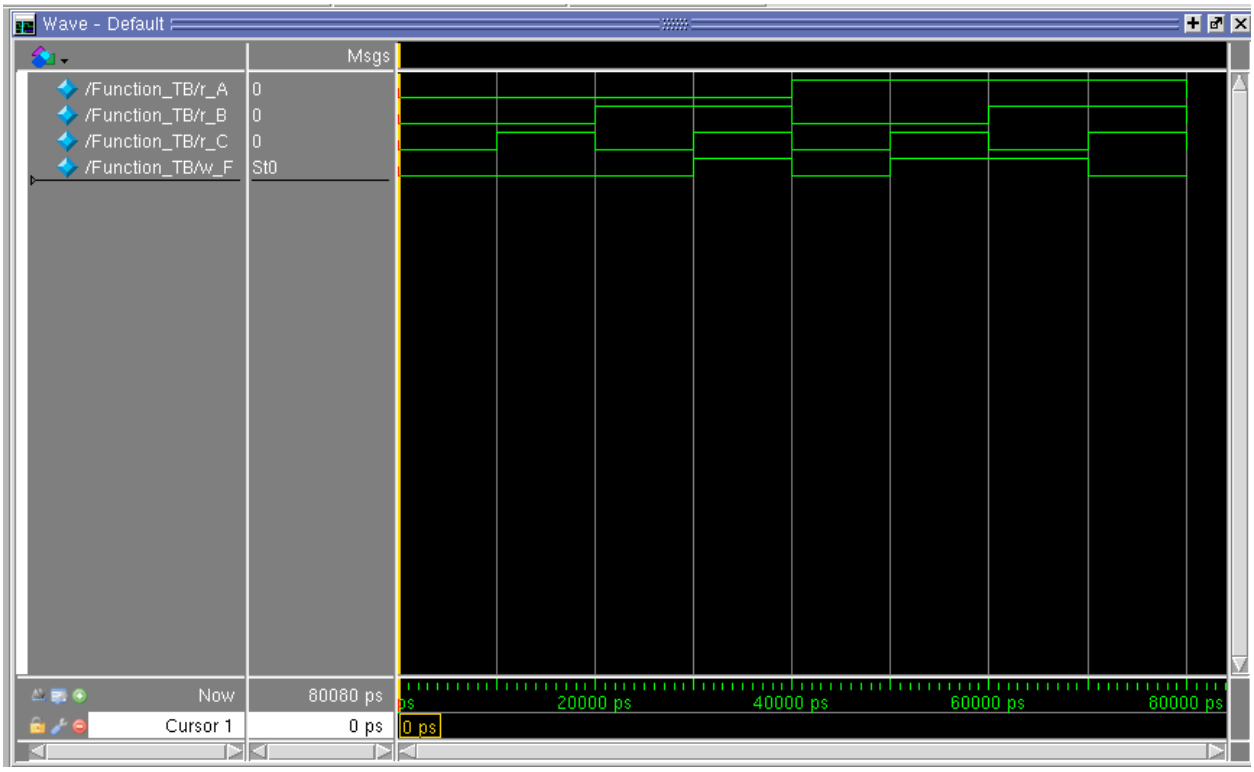
```
Ln#
 1
 2   `timescale 1ns/1ps
 3   module OR3 (i_A, i_B, i_C, o_F);
 4     input i_A, i_B, i_C;
 5     output o_F;
 6     wire A_or_B;
 7
 8     OR2 nor0(.i_A(i_A), .i_B(i_B), .o_F(A_or_B));
 9     OR2 nor1(.i_A(i_C), .i_B(A_or_B), .o_F(o_F));
10
11   endmodule
12
13
14
```

```
Ln#
 1
 2   `timescale 1ns/1ps
 3   module AND2 (i_A, i_B, o_F);
 4     input i_A, i_B;
 5     output o_F;
 6     wire A_not, B_not;
 7
 8     NOR2 nor0(.i_A(i_A), .i_B(i_A), .o_F(A_not));
 9     NOR2 nor1(.i_A(i_B), .i_B(i_B), .o_F(B_not));
10     NOR2 nor2(.i_A(A_not), .i_B(B_not), .o_F(o_F));
11
12   endmodule
```

```
Ln#
 1
 2   `timescale 1ns/1ps
 3   module AND3 (i_A, i_B, i_C, o_F);
 4     input i_A, i_B, i_C;
 5     output o_F;
 6     wire A_and_B;
 7
 8     AND2 and0(.i_A(i_A),     .i_B(i_B), .o_F(A_and_B));
 9     AND2 and1(.i_A(A_and_B), .i_B(i_C), .o_F(o_F));
10
11   endmodule
```

```
Ln#
 1
 2
 3   `timescale 1ns/1ps
 4   module Function (i_A, i_B, i_C, o_F);
 5     input i_A, i_B, i_C;
 6     output o_F;
 7     wire w_nA, w_nB, w_nC;
 8     wire w_A1, w_A2, w_A3;
 9
10     NOR2  nor0(.i_A(i_A), .i_B(i_A), .o_F(w_nA));
11     NOR2  nor1(.i_A(i_B), .i_B(i_B), .o_F(w_nB));
12     NOR2  nor2(.i_A(i_C), .i_B(i_C), .o_F(w_nC));
13     AND3 and1(.i_A(w_nA),.i_B(i_B), .i_C(i_C), .o_F(w_A1));
14     AND3 and2(.i_A(i_A), .i_B(w_nB),.i_C(i_C), .o_F(w_A2));
15     AND3 and3(.i_A(i_A), .i_B(i_B), .i_C(w_nC), .o_F(w_A3));
16     OR3   or0(.i_A(w_A1), .i_B(w_A2),.i_C(w_A3), .o_F(o_F));
17
18   endmodule
19
20
21
```

```
Ln#
 1   `timescale 1ns/1ps
 2   module Function_TB();
 3     reg r_A, r_B, r_C;
 4     wire w_F;
 5
 6     initial
 7     begin
 8       r_A = 1'b0;
 9       r_B = 1'b0;
10       r_C = 1'b0;
11     end
12
13     always
14       #10 r_C=~r_C;
15     always
16       #20 r_B=~r_B;
17     always
18       #40 r_A=~r_A;
19
20     Function U(.i_A(r_A),.i_B(r_B),.i_C(r_C),.o_F(w_F));
21
22   endmodule
23
24
25
```


Wave - Default
Msgs
/Function_TB/r_A
/Function_TB/r_B
/Function_TB/r_C
/Function_TB/w_F
0
0
0
St0
Now
80080 ps
Cursor 1
0 ps
0 ps
20000 ps
40000 ps
60000 ps
80000 ps