	Normal Cost
$F_A = A + C + \bar{B}\bar{D} + BD$	2 AND gates + 3 OR gates
$F_B = \bar{B} + CD + \bar{C}\bar{D}$	2 AND gates + 2 OR gates
$F_C = B + \bar{C} + D$	2 OR gates
$F_D = A + \bar{B}C + \bar{B}\bar{D} + C\bar{D} + B\bar{C}D$	5 AND gates + 4 OR gates
$F_E = \bar{B}\bar{D} + C\bar{D}$	2 AND gates + 1 OR gate
$F_F = A + B\bar{C} + B\bar{D} + \bar{C}\bar{D}$	3 AND gates + 3 OR gates
$F_g = A + B\bar{C} + \bar{B}C + B\bar{D}$	3 AND gates + 3 OR gates
Total	17 AND gates + 18 OR gates + 3 NOT gates
	5 AND ICs 5 OR ICs 1 NOT IC

Simplification of cost: Step 1
Remove like terms:

$F_A = A + C + \underline{\bar{B}\bar{D}} + \boxed{BD}$	2 AND gates + 3 OR gates
$F_B = \bar{B} + CD + \underline{\bar{C}\bar{D}}$	2 AND gates + 2 OR gates
$F_C = B + \bar{C} + \bar{D}$	2 OR gates
$F_D = A + \underbrace{\bar{B}C}_{} + \underline{\bar{B}\bar{D}} + \boxed{C\bar{D}} + \boxed{(BD)}\bar{C}$	3 AND gates + 4 OR gates
$F_E = \underline{\bar{B}\bar{D}} + \boxed{C\bar{D}}$	1 OR gate
$F_F = \underbrace{A + B\bar{C} + B\bar{D}}_{} + \underline{\bar{C}\bar{D}}$	2 AND gate + 3 OR gate
$F_g = \underbrace{A + B\bar{C}}_{} + \underbrace{\bar{B}C}_{} + B\bar{D}$	1 OR gate
Total	9 AND gates + 16 OR gates + 3 NOT gates
	3 AND ICs + 4 OR ICs + 1 NOT IC

Notice how there are 9 AND gates, but 4 gates for one AND IC. Ideally, we want to get the number of gates to a multiple of 4, in this case, 8. Now look at the number of NOT gates. There are 3 when there could be 6 on an IC. So, we can use DeMorgan's to create AND gate outputs from existing OR gate results using a NOT gate. Keep in mind most simplifications of one gate create expenses in others

$F_A = A + C + \overline{B}\,\overline{D}^{1} + BD^{2}$ — 2 AND gates + 3 OR gates

$F_B = \overline{B} + CD + \overline{C}\,\overline{D}^{3}$ — 1 AND gate + 2 OR gates + 1 NOT gate

↑ DeMorgan's

$F_C = B + (\overline{D} + \overline{C})$ — 2 OR gates

$F_D = A + \overline{B}C^{4} + 1 + C\overline{D}^{5} + 2\overline{C}$ — 3 AND gates + 4 OR gates

$F_E = 1 + 5$ — 1 OR gate

$F_F = (A + B\overline{C} + B\overline{D})^{6} + 3$ — 2 AND gates + 3 OR gates

$F_g = 6 + 4$ — 1 OR gate

Total: 8 AND gates, 16 OR gates, and 4 NOT gates

Successfully dropped 1 IC $\longrightarrow$ 2 AND ICs, 4 OR ICs, 1 NOT IC

We can also use XOR simplification: $A \oplus B = A\overline{B} + \overline{A}B$ and $\overline{A \oplus B} = AB + \overline{A}\,\overline{B}$

$F_A = A + C + (\overline{B}\,\overline{D} + BD) \longrightarrow \overline{B \oplus D}$ — $F_A = A + C + \overline{B \oplus D}$

$F_B = \overline{B} + (CD + \overline{C}\,\overline{D}) \longrightarrow \overline{C \oplus D}$ — $F_B = B + \overline{C \oplus D}$

$F_C = B + \overline{C} + \overline{D}$ — $F_C = B + \overline{C} + \overline{D}$

$F_D = A + \overline{B}C + \overline{B}\,\overline{D} + C\overline{D} + B\overline{C}D$ — $F_D = A + \overline{B}C + \overline{B}\,\overline{D} + C\overline{D} + B\overline{C}D$

$F_E = \overline{B}\,\overline{D} + C\overline{D}$ — $F_E = \overline{B}\,\overline{D} + C\overline{D}$

$F_F = A + B\overline{C} + B\overline{D} + \overline{C}\,\overline{D}$ — $F_F = A + B\overline{C} + B\overline{D} + \overline{C}\,\overline{D}$

$F_g = A + \underbrace{B\overline{C} + \overline{B}C}_{\longrightarrow (B \oplus C)} + B\overline{D}$ — $F_g = A + B\overline{C} + \overline{B}C + B\overline{D}$

↑ This one actually doesn't help us

We can also simplify through expansion. This one is tricky and highly dependent on the function. In this case:

$$F_D = A + \bar{B}C + \bar{B}\bar{D} + \underline{C\bar{D}} + B\bar{C}D = A + \bar{B}C + \bar{B}\bar{D} + \underline{BC\bar{D}} + B\bar{C}D$$

Expanding the term $C\bar{D}$ to $BC\bar{D}$ results in the same equation in this case because of Don't Care's and other terms that overlap

Now we can combine $BC\bar{D}$ and $B\bar{C}D$ into $B(C \oplus D)$

We now have the following equations

$F_A = A + C + \overline{B \oplus D}$

$F_B = B + \overline{C \oplus D}$

$F_C = B + \bar{C} + \bar{D}$

$F_D = A + \bar{B}C + \bar{B}\bar{D} + B(C \oplus D)$

$F_E = \bar{B}\bar{D} + C\bar{D}$

$F_F = A + B\bar{C} + B\bar{D} + \bar{C}\bar{D}$

$F_g = A + B\bar{C} + B\bar{D} + \bar{B}C$

The last possible simplification is factoring and matching

$F_A = A + C + \overline{B \oplus D}$	2 OR gates, 1 XOR, 1 NOT gate
$F_B = B + \overline{C \oplus D}$	1 OR gate, 1 XOR gate, 1 NOT gate
$F_C = B + \underline{\bar{C} + \bar{D}}$	2 OR gate
$F_D = A + \bar{B}(C + \bar{D}) + B(C \oplus D)$	2 AND gate, 3 OR gate
$F_E = \bar{D}(\bar{B} + C)$	1 AND gate, 1 OR gate
$F_F = \underbrace{A + B(\bar{C} + \bar{D})} + \bar{C}\bar{D}$	2 AND gate, 2 OR gate
$F_g = A + B(\bar{C} + \bar{D}) + \bar{B}C$	1 AND gate, 1 OR gate

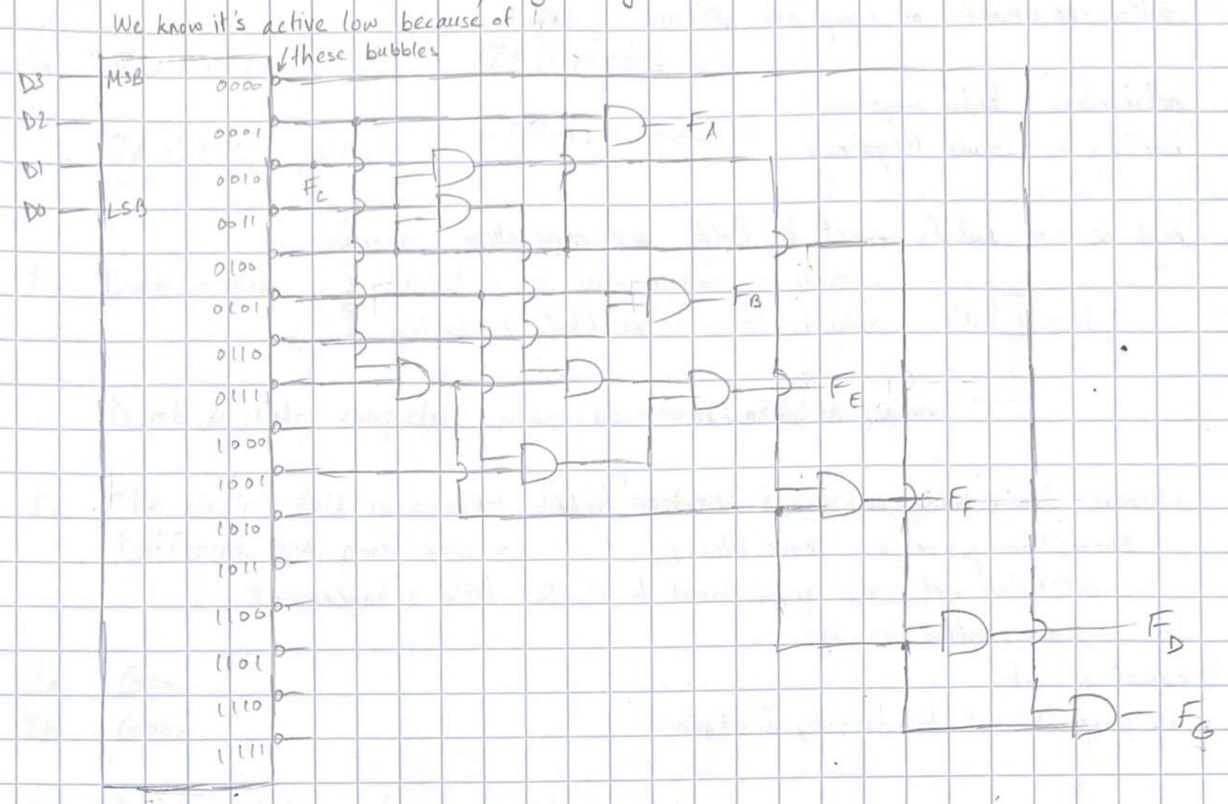
Factoring here and here removed an AND gate

Total: 12 OR gate, 6 AND gate, 2 XOR gate, 5 NOT gate

3 OR IC   2 AND IC   1 XOR IC   1 NOT IC

7 Total ICs. Same as not using XOR simplification
But total number of gates used drops from 28 to 25

Using a 4-to-16 demultiplexor/line decoder, we can make the same circuit. Remember, in an active low decoder, everything is high (5V) except the value corresponding to the input



Since the output is 0 when the corresponding input is high (5V), we want our maxterms to always be 1 for the output of the function to be 1

Total number of gates here = 11 AND gates + 1 4-to-16 demultiplexor

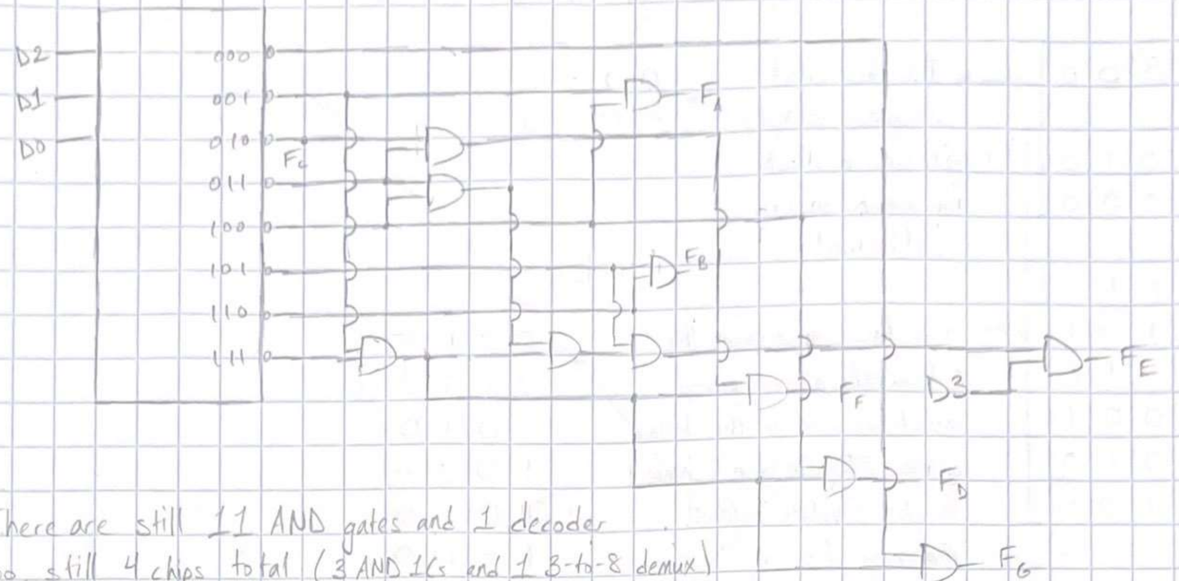
Total of 4 ICs

Since 4-to-16 decoders are uncommon, instead, 2 3-to-8 decoders can be used. However, only one output actually uses term 9, so we can remove 1 3-to-8 decoder and use another AND gate instead

Total number of gates = 11 AND gates + 1 3-to-8 demultiplexor

Total of 4 ICs still

This would be the implementation with a 3-to-8 decoder:



There are still 11 AND gates and 1 decoder
So, still 4 chips total (3 AND ICs and 1 3-to-8 demux)

Now what if we want to implement this function using an active high decoder? Previously we ANDed the terms we didn't want because if the line was a 0, the output should be 0. But now terms we don't want will be 1 when decoded and all other lines will be 0. So, the optimal logic function to create a 0 if there is a single 1 term would be OR followed by a NOT gate at the end.

Therefore, we can copy paste the above circuit, but replace all the AND gates with OR gates and the active low line decoder with an active high decoder. Then add NOT gates where each of the outputs are.
Because of the added NOT gates, it actually increases the number of chips used.
This would be one example of why you should consider using active high vs active low when building a circuit.
Another reason would be power consumption cost.