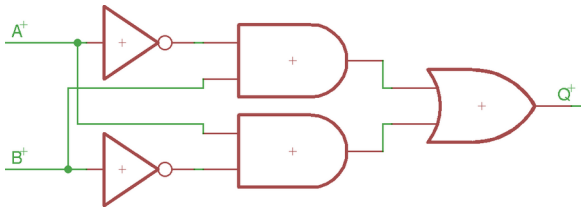
1. Podstawy logiki w automatyce

Bramki logiczne – ang. *gates* (nazywane także *funktorami logicznymi*) są najprostszymi układami cyfrowymi realizującymi elementarne funkcje logiczne. Służą one do budowy układów logicznych o większej złożoności, takich jak multipleksery, przerzutniki, pamięci, procesory.

Układy logiczne – układy przekaźnikowe, pneumatyczne, elektroniczne mające za zadanie informacje wejściowe (w postaci zer i jedynek) przetworzyć na informacje wyjściowe (również w postaci zer i jedynek) zgodnie z określonym algorytmem. Przykładem takiego układu będzie układ sterujący wymianą wody w basenie – w zależności od poziomu wody zamykane lub otwierane są zawory odpływu i dopływu wody (patrz zadanie 1.6).

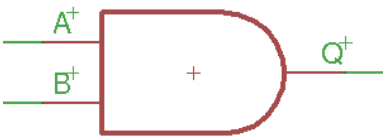
Przykładem takiego układu (bardzo prostego) może być bramka XOR (exclusive OR) – prawda jeśli jedno wejście jest prawdziwe a drugie fałszywe:



System funkcjonalnie pełny – zestaw funkcji logicznych umożliwiający algebraiczny zapis dowolnych funkcji logicznych. **Podstawowym systemem funkcjonalnie pełnym** jest zestaw bramek AND (oraz), OR (lub) oraz NOT (zaprzeczenie). Istnieją również inne systemy funkcjonalnie pełne, np. system składający się z jednej bramki NOR.

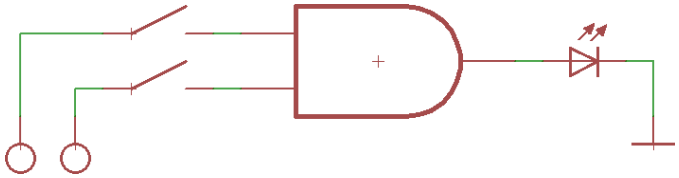
Zaprojektowanie układu można podzielić na trzy etapy: **przygotowanie tabeli prawdy, na jej podstawie stworzenie postaci kanonicznej i na tej podstawie narysowanie schematu.**

Bramka AND

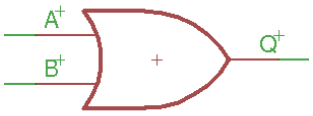
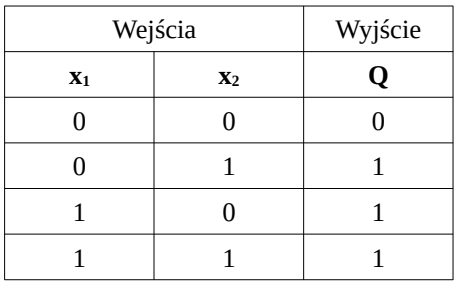
<table>
<tr><th>Funkcja logiczna</th><th>Symbol logiczny</th><th>Wyrażenie algebraiczne</th><th>Tabela prawdy</th></tr>
<tr>
<td>AND
(koniunkcja)</td>
<td> 
</td>
<td>$x_1 \bullet x_2 = Q$</td>
<td>
<table>
<tr><th colspan="2">Wejścia</th><th>Wyjście</th></tr>
<tr><th>x_1</th><th>x_2</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
</table>

Koniunkcja (i, ∧ , &) odpowiada działaniu bramki logicznej AND. Wynikiem *działania* bramki AND jest prawda, gdy wszystkie wejścia są prawdziwe. Bramka AND może mieć więcej niż dwa wejścia.

Przykład: jeśli do wejść A i B bramki AND podłączymy przyciski, zaś do wyjścia Q diodę, ta zapali się gdy oba przyciski zostaną wciśnięte.



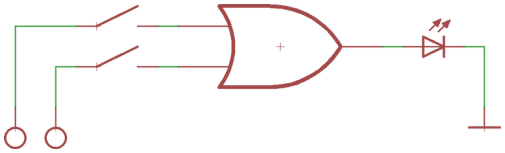
Bramka OR

Funkcja logiczna	Symbol logiczny	Wyrażenie algebraiczne	Tabela prawdy
OR (alternatywa)		$x_1+x_2=Q$	

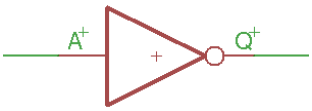
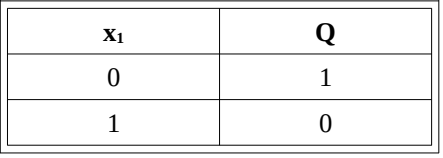
Wejścia		Wyjście
x_1	**x_2**	**Q**
0	0	0
0	1	1
1	0	1
1	1	1

Alternatywa (lub, ∨ , |) odpowiada działaniu bramki OR. Wynikiem *działania* bramki OR jest prawda wtedy, gdy przynajmniej jedno z wejść jest prawdziwe. Bramka OR może mieć więcej niż dwa wejścia.

Przykład: jeśli do wejść A i B bramki OR podłączymy przyciski, zaś do wyjścia Q diodę, ta zaświeci się gdy przynajmniej jeden z przycisków zostanie wciśnięty.



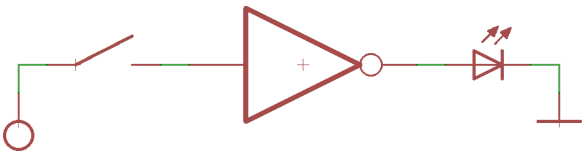
Bramka NOT

Funkcja logiczna	Symbol logiczny	Wyrażenie algebraiczne	Tabela prawdy
NOT (zaprzeczenie)		$\bar{x}_1=Q$	

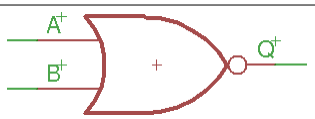
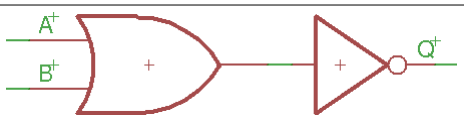
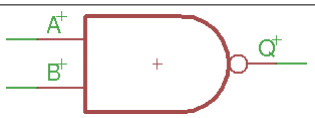
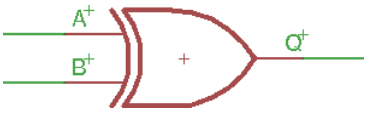
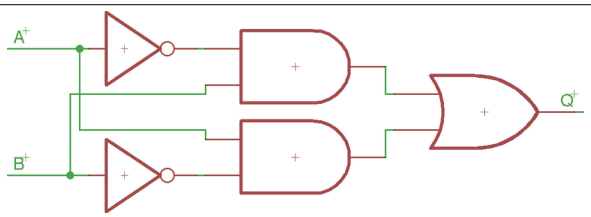
x_1	**Q**
0	1
1	0

Zaprzeczenie (nie, ¬, ~, !) odpowiada działaniu bramki NOT. Wynikiem *działania* bramki NOT jest prawda wtedy, gdy wejście jest fałszywe.

Przykład: jeśli do wejścia podłączymy przycisk, zaś do wyjścia diodę, ta ZGAŚNIE, gdy wciśniemy przycisk.



Inne bramki

	Symbol	Odpowiednik
NOR (NOT OR)		
NAND (NOT AND)		
XOR (exclusive OR)		

Wypełnianie tabeli prawdy

Aby przygotować tabelę prawdy należy ustalić ile układ ma lub ma mieć wejść oraz ile wyjść. Wejścia można opisać jako x_1, x_2, x_3 itd. lub według innej koncepcji (np. A, B, C itd.), wyjścia zazwyczaj oznacza się jako Q_1, Q_2, Q_3 itd. Nagłówek tabeli z trzema wejściami i dwoma wyjściami może wyglądać tak:

Wejścia			Wyjścia	
x_1	**x_2**	**x_3**	**Q_1**	**Q_2**

Liczba wierszy z *danymi* zależna jest <u>wyłącznie</u> od ilości <u>wejść</u> i będzie wynosić 2^n, gdzie n – ilość wejść.

Dla powyższego nagłówka, będzie to zatem $2^n = 2^3 = 8$ czyli:

	Wejścia			Wyjścia	
	x_1	**x_2**	**x_3**	**Q_1**	**Q_2**
0					
1					
2					
3					
4					
5					
6					
7					

Wypełnienie kolumn wejść jest <u>niezależne</u> od treści zadania i musi być wykonane w tak, by żaden z wierszy się nie powtórzył (w tym przykładzie – nie pojawia się taka sama kombinacja wejść x_1, x_2, x_3). Jedną z propozycji wypełnienia takiej tabeli jest – od lewej, od góry – wypełnienie połowy wierszy „0”, drugiej połowy „1”. Następnie, w kolejnej kolumnie, połowę wierszy z zerami z poprzedniej kolumny wypełnia się „0”, drugą połowę „1” itd.:

Wejścia			Wyjścia	
x_1	**x_2**	**x_3**	**Q_1**	**Q_2**
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Pozostałą część tabeli należy wypełnić zerami oraz jedynkami w zależności od treści zadania. Poza wartościami logicznymi w tabeli może pojawić się X, oznaczający „bez znaczenia”. Np. jeśli wyjścia oznaczają „włącz silnik”, „obroty prawe” oraz „obroty lewe”, to gdy silnik jest wyłączony w pozostałych kolumnach mogą pojawić się „X”.

Postać kanoniczna

Postać kanoniczna, to forma zapisu funkcji odpowiadającej *działaniu* tabeli prawdy. Jest to jeden ze sposobów pozwalających na łatwe przeniesienie tabeli prawdy na układ logiczny.

Przykład 1.1

Wejścia			Wyjścia	
x_1	x_2	x_3	Q_1	Q_2
0	0	0	**0**	0
0	0	1	1	0
0	1	0	1	**1**
0	1	1	**0**	0
1	0	0	1	**1**
1	0	1	1	0
1	1	0	**0**	0
1	1	1	1	**1**

Dla każdej tabeli prawdy można utworzyć **kanoniczną postać alternatywną** oraz **kanoniczną postać koniunkcyjną** dla <u>każdego wyjścia</u>. Wyjścia można interpretować jako oddzielne zadania. A więc:

Wejścia			Wyjście
x_1	x_2	x_3	Q_1
0	0	0	**0**
0	0	1	1
0	1	0	1
0	1	1	**0**
1	0	0	1
1	0	1	1
1	1	0	**0**
1	1	1	1

Wejścia			Wyjście
x_1	x_2	x_3	Q_2
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	0
1	0	0	**1**
1	0	1	0
1	1	0	0
1	1	1	**1**

Kanoniczna postać alternatywna to suma (alternatywa) odpowiednio zapisanych ilorazów (funkcji K_n) danej kombinacji wejść pomnożonej przez odpowiadające im wyjścia. Funkcje K_n to iloczyny wejść zaprzeczonych tam, gdzie wejście jest <u>fałszem</u>.

Kanoniczna postać koniunkcyjna to iloraz (koniunkcja) odpowiednio zapisanych sum (funkcji D_n) danej kombinacji wejść oraz odpowiadającego im wyjścia. Funkcje D_n to sumy wejść zaprzeczonych tam, gdzie wejście jest <u>prawdą</u>.

Uwaga: Na ogół, korzystając z praw algebry Boole’a, można przekształcać postacie kanoniczne w celu zmniejszenia liczby występujących w nich elementarnych operacji logicznych, co nazywamy minimalizacją funkcji logicznych. Inne metody minimalizacji, to metoda Quine’a-McCluskey’a, metoda tablic Karnaugha.

Wejścia			Wyjście	Postać alternatywna		Postać koniunkcyjna	
$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{Q_1}$	$\mathbf{K_n}$	$\mathbf{K_n \cdot Q_1}$ (1)	$\mathbf{D_n}$	$\mathbf{D_n + Q_1}$ (2)
0	0	0	**0**	$\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3$	0	$x_1 + x_2 + x_3$	
0	0	1	1	$\bar{x}_1 \cdot \bar{x}_2 \cdot x_3$		$x_1 + x_2 + \bar{x}_3$	1
0	1	0	1	$\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$		$x_1 + \bar{x}_2 + x_3$	1
0	1	1	**0**	$\bar{x}_1 \cdot x_2 \cdot x_3$	0	$x_1 + \bar{x}_2 + \bar{x}_3$	
1	0	0	1	$x_1 \cdot \bar{x}_2 \cdot \bar{x}_3$		$\bar{x}_1 + x_2 + x_3$	1
1	0	1	1	$x_1 \cdot \bar{x}_2 \cdot x_3$		$\bar{x}_1 + x_2 + \bar{x}_3$	1
1	1	0	**0**	$x_1 \cdot x_2 \cdot \bar{x}_3$	0	$\bar{x}_1 + \bar{x}_2 + x_3$	
1	1	1	1	$x_1 \cdot x_2 \cdot x_3$		$\bar{x}_1 + \bar{x}_2 + \bar{x}_3$	1

(1) FAŁSZ pomnożony przez *dowolne wyrażenie logiczne* daje 0.
(2) Wynik ≥ 1 w logice to PRAWDA, zatem 1 + *dowolne wyrażenie logiczne* daje 1.

Postać alternatywna to SUMA wartości w kolumnie $\mathbf{K_n \cdot Q_1}$, czyli:

$$f(x_1, x_2, x_3) = 0 + \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + 0 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 \cdot x_3 + 0 + x_1 \cdot x_2 \cdot x_3 \quad ,$$

co można uprościć do postaci:

$$f(x_1, x_2, x_3) = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot x_3 \quad .$$

Postać koniunkcyjna to ILOCZYN wartości w kolumnie $\mathbf{D_n + Q_1}$, czyli:

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \cdot 1 \cdot 1 \cdot (x_1 + \bar{x}_2 + \bar{x}_3) \cdot 1 \cdot 1 \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \cdot 1 \quad ,$$

co można uprościć do postaci:

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \cdot (x_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \quad .$$

Jak widać, postać koniunkcyjna jest mniej złożona niż postać alternatywna, ale obie funkcje są poprawne.

Aby z powyższej funkcji stworzyć układ logiczny, należy przyjąć jedną z dwóch *taktyk*:

- od ogółu do szczegółu lub
- od szczegółu do ogółu.

W pierwszym przypadku należy *nawiasy* rozpatrywać jako wejścia, czyli np.:

$$F_1 = g(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \quad , \quad F_2 = h(x_1, x_2, x_3) = (x_1 + \bar{x}_2 + \bar{x}_3) \quad , \quad F_3 = i(x_1, x_2, x_3) = (\bar{x}_1 + \bar{x}_2 + x_3) \quad .$$

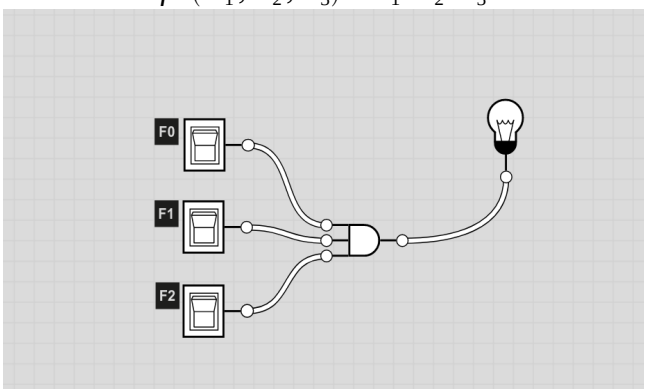
Co upraszcza (na chwilę) wzór do:

$$f'(F_1, F_2, F_3) = F_1 \cdot F_2 \cdot F_3 \quad ,$$

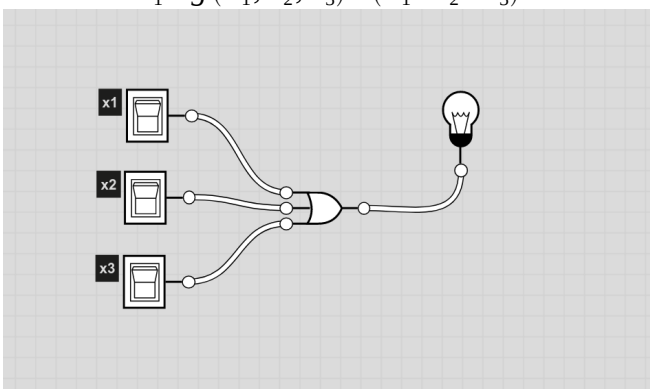
co jest niczym innym jak funkcją odpowiadającą działaniu bramki AND. Funkcjom F0, F1 oraz F2 odpowiada działanie bramki OR z bramkami NOT w przypadku wejść zaprzeczonych ($\bar{x}_n$).

Poniższe schematy wykonano na stronie logic.ly/demo.

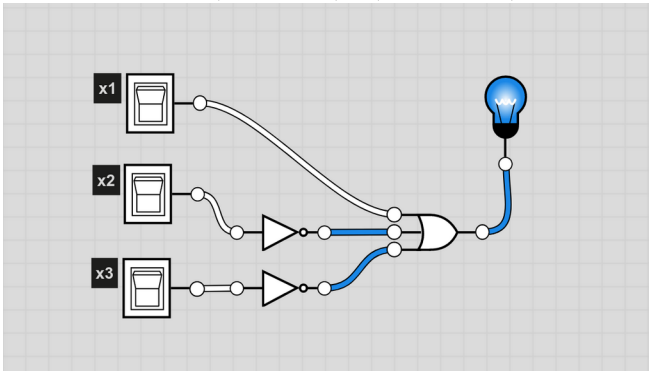
$f'(F_1, F_2, F_3) = F_1 \cdot F_2 \cdot F_3$



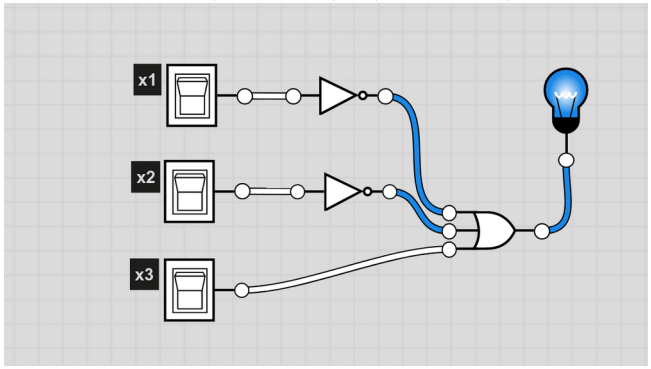
$F_1 = g(x_1, x_2, x_3) = (x_1 + x_2 + x_3)$



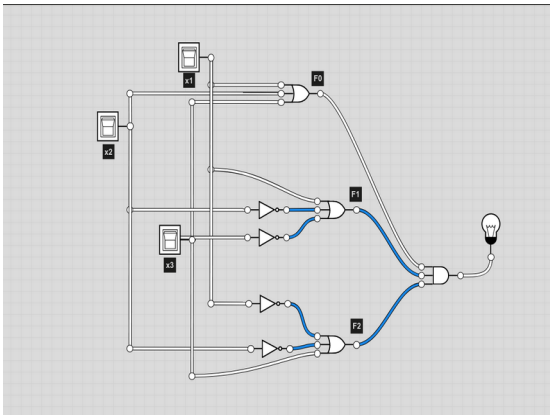
$F_2 = h(x_1, x_2, x_3) = (x_1 + \bar{x}_2 + \bar{x}_3)$

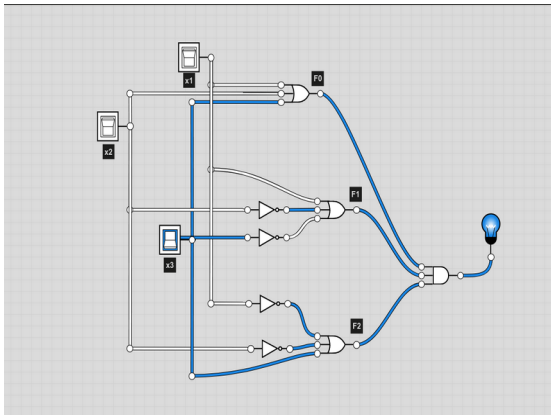


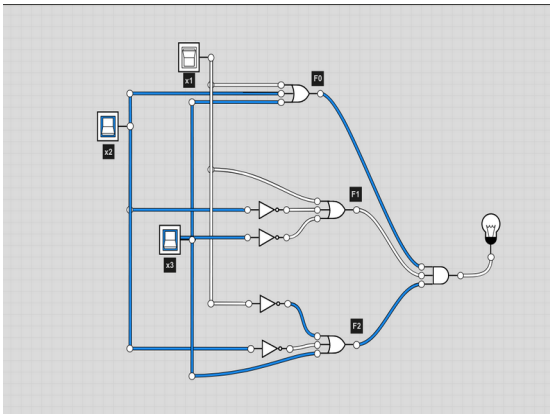
$F_3 = i(x_1, x_2, x_3) = (\bar{x}_1 + \bar{x}_2 + x_3)$

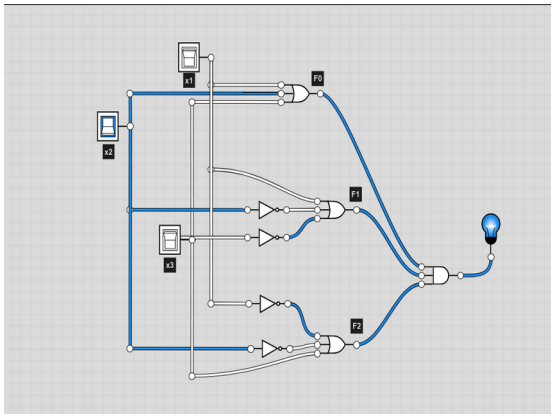


Aby otrzymać układ odpowiadający pierwotnej funkcji, należy połączyć wejścia x1, x2 oraz x3 oraz wyjścia układów F1, F2 oraz F3 podłączyć jako odpowiednie wejścia bramki AND i przetestować różne kombinacje wejść:









itd.

Zadanie 1.1. Dla wyjścia Q_2 przykładu 1.1 przeprowadzić cały proces projektowania układu: utworzyć tabelę z kolumnami K_n i D_n, wyprowadzić kanoniczną postać alternatywną oraz kanoniczną postać koniunkcyjną, wybrać funkcję prostszą i zrealizować ją na stronie logic.ly.

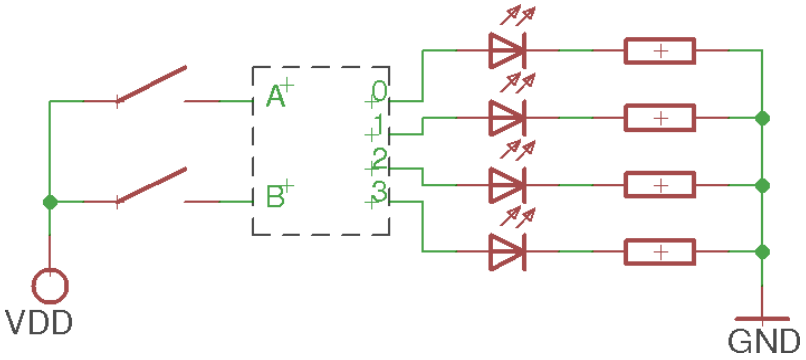
Zadanie 1.2. Za pomocą podstawowych bramek logicznych (AND, OR, NOT) zbudować bramkę XNOR (inaczej NXOR lub XAND). Czy da się ją zbudować z 5 bramek podstawowych, podobnie jak bramkę XOR (patrz przykłady *innych bramek*)?

Tablica prawdy XNOR

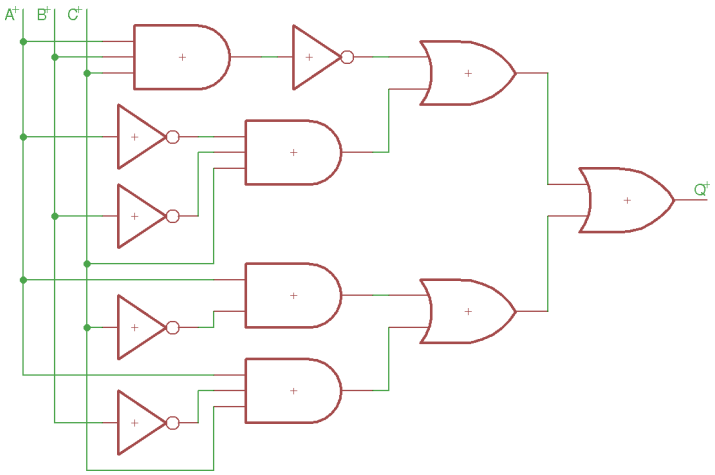
x_1	x_2	Q
0	0	1
0	1	0
1	0	0
1	1	1

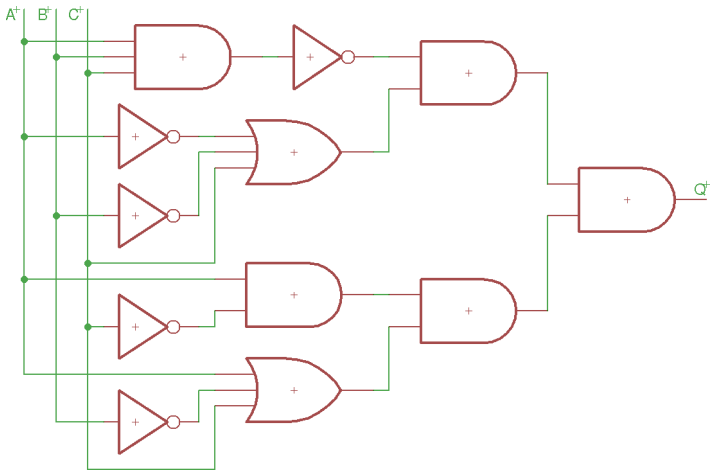
Zadanie 1.3. Zaprojektować demultiplekser dwu-bitowy, taki, że:

Wejścia		Wyjścia			
x_1	x_2	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Zadanie 1.4. Zapisać tabele prawdy dla poniższych schematów.



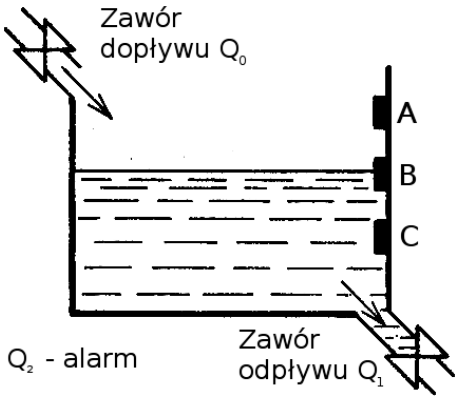


Zadanie 1.5. Zaprojektować układ kombinacyjny sterujący pracą dźwigu (suwnicy) pracującego wg następującego algorytmu: Dźwig przenosi bloczki betonowe z punktu A do punktu B (x_2). Zaczepienie bloczka (załadunek) oraz odczepienie bloczka (rozładunek) wykonywane są przez pracowników, co sygnalizowane jest odpowiednim stanem czujnika C (x_3=1 gdy bloczek zaczepiony, x_3=0 gdy bloczek odczepiony). Obecność dźwigu w okolicach punktu A sygnalizowana jest pojawieniem się na wejściu x_1 stanu „1”, podobnie jak z czujnikiem B, który sygnalizuje pojawienie się dźwigu w punkcie na wejściu x_2. Sygnały sterujące pracą dźwigu to Q_1 – ruch suwnicy w lewo i Q_2 – ruch suwnicy w prawo.

Zadanie 1.6. Na podstawie polecenia zbudować tablicę wartości, a następnie zaprojektować układ logiczny.

System sterowania napełniania basenu kąpielowego

Dopływ wody do basenu jest sterowny zaworem Q_0, a jego odpływ zaworem Q_1. Podanie wysokiego stanu logicznego do układu sterowania zaworu oznacza otwarcie zaworu, a stanu niskiego – zamknięcie. W basenie znajdują się trzy czujniki A, B, C podłączone kolejno do wejść x_1, x_2 i x_3, wyznaczające odpowiednio maksymalny, średni i minimalny poziom wody. Zadziałanie czujnika następuje po zanurzeniu go w wodzie i jest sygnalizowane pojawieniem się na jego wyjściu jedynki logicznej. Prędkości odpływu i dopływu wody do basenu w zależności od jej ciśnienia, mogą być różne. Lustro wody nie powinno obniżać się poniżej poziomu minimalnego. Dodatkowo jest włączony sygnałem Q_2 układ alarmu przy uszkodzeniu któregoś z czujników. Jednocześnie z sygnałem alarmu następuje zamknięcie zaworu dopływu i otwarcie zaworu odpływu wody. Między stanami wody średnim i maksymalnym powinny być otwarte oba zawory w celu ciągłej wymiany w basenie.



Zadanie dodatkowe: zaprojektować układ z użyciem wyłącznie bramek NOR.

Zadanie 1.7. Za pomocą bramek podstawowych zbudować układy realizujące poniższe tabele prawd. Uwaga: te układ 1 należy wykonać bez korzystania z postaci kanonicznych. Tabela 3 to zadanie z gwiazdką.

x_1	x_2	x_3	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

x_1	x_2	x_3	x_4	Q_1	Q_2	Q_3
1	0	0	0	0	0	0
0	1	0	0	1	0	0
0	0	1	0	0	1	0
0	0	0	1	1	1	0
0	0	0	0	X	X	1
Pozostałe				X	X	1

X – bez znaczenia

x_1	x_2	x_3	Q_1	Q_2	Q_3
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	0	X	X
1	1	0	0	0	X
1	1	1	X	1	1

Zadanie 1.8. Za pomocą wyłącznie bramek NAND a następnie bramek NOR zbudować bramki NOT, AND oraz OR.

Źródła:

http://home.agh.edu.pl/~byrska/ETI/7BramkiLogiczne.pdf
https://pl.wikibooks.org/wiki/Matematyka_dla_liceum/Logika
Biblioteki Eagle (Easily Applicable Graphical Layout Editor) V6.6.0 for Linux
Ćwirko R., Rusek M., Marciniak W.: Układy scalone w pytaniach i odpowiedziach. WNT 1987
logic.ly/demo