

**DSCH2** to program do edycji i symulacji układów logicznych. DSCH2 jest wykorzystywany do sprawdzenia architektury układu logicznego przed rozpoczęciem projektowania fizycznego. DSCH2 zapewnia ergonomiczne środowisko do projektowania hierarchicznych układów logicznych i symulacji złożonych układów razem z analizą opóźnień. Program ten umożliwia łączenie ze sobą na potrzeby testów, predefiniowanych komponentów bibliotecznych oraz własnych elementów opracowanych na poziomie bramek logicznych i pojedynczych tranzystorów MOS.

## **UWAGI**

### **Źródła informacji**

Informacje dotyczące struktury (schematów) oraz działania układów projektowanych w poniższych zadaniach można uzyskać z lektury tej instrukcji, materiałów wykładowych lub ze studiów własnych dostępnych źródeł, w tym zasobów sieci Internet.

O ile struktura analizowanych układów nie jest jednoznacznie podana w treści poniższych zadań, można wyprowadzić ją przy pomocy formuł boolowskich lub oprzeć się na istniejących rozwiązaniach opisanych w dostępnych źródłach. W tym drugim przypadku konieczne jest zamieszczenie w przygotowanym sprawozdaniu kompletnego opisu działania projektowanych układów.

Informacje dotyczące szczegółów działania oprogramowania DSCH można znaleźć w podręcznikach w sieci Internet oraz od prowadzącego, podczas zajęć lub w godzinach konsultacji.

### **Sprawozdania**

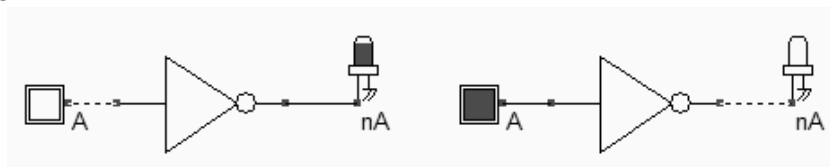
Dla zadań przedstawionych poniżej należy wykonać proste sprawozdanie. Dla każdego z zadań powinno ono zawierać opis działania lub wyprowadzenie struktury w formie przekształceń formuł boolowskich, zrzut ekranu ze schematem opracowanego układu, opis testów i ich wyników oraz przykładowe zrzuty przebiegu badanych sygnałów.

W przypadku zajęć prowadzonych zdalnie lub na polecenie osoby prowadzącej zajęcia, należy załączyć lub udostępnić pliki projektowe wykonanych zadań, w celu umożliwienia sprawdzenia działania zaprojektowanych układów przez osobę prowadzącą.

## ZADANIA – UKŁADY NA POZIOMIE BRAMEK LOGICZNYCH

### 1.1. Inwerter logiczny

Pierwszy przykład pokazuje prostą symulację układu logicznego inwertera. Należy wybrać z menu **File**→ **Open** i otworzyć plik z układem inwertera (inverter.sch). Ten układ zawiera jeden przycisk, układ inwertera oraz diodę. Aby rozpocząć symulację należy wybrać z menu **Simulate**→ **Start simulation**.



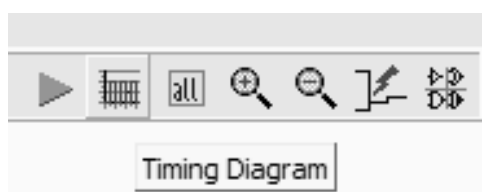
*Układ do symulacji inwertera (Inverter.SCH)*

Naciskając na przycisk po lewej stronie układu zmienia się sygnał wejściowy. Sygnał wyjściowy pokazuje dioda. Kolor czerwony oznacza logiczne 1, kolor czarny logiczne 0. Aby zatrzymać symulację i powrócić do edytora należy nacisnąć przycisk **Stop simulation**.










*Przycisk Stop Simulation*

Należy nacisnąć ikonę **chronogram**, aby uzyskać dostęp do wykresów czasowych symulacji. Jak widać na wykresach, wartość sygnału wyjściowego jest logicznym przeciwieństwem sygnału wejściowego.



### 1.2. Podstawowe bramki logiczne

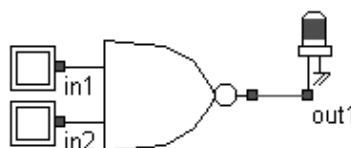
Program DSCH2 umożliwia budowę złożonych układów z podstawowych bramek logicznych. Poniższa tabela pokazuje symbole podstawowych bramek razem z opisem ich funkcji logicznych. Symbol & oznacza operację logiczną AND, | oznacza OR, ~ oznacza INVERT a ^ oznacza XOR.

Nazwa układu	Funkcja logiczna	Symbol
INWERTER	$Out = \sim in$	
AND	$Out = a \& b$	
NAND	$Out = \sim(a \cdot b)$	
OR	$Out = (a   b)$	
NOR	$Out = \sim(a   b)$	
XOR	$Out = a \wedge b$	
XNOR	$Out = \sim(a \wedge b)$	

### 1.3. Bramka NAND

Tablica prawdy bramki NAND i schemat układu do symulacji są pokazane poniżej. Aby zbudować ten układ w programie DSCH należy wybrać z palety poszczególne elementy układu: bramkę NAND, dwa przyciski oraz diodę. W razie konieczności można wykorzystać "interconnects" do połączenia elementów. Po zbudowaniu układu należy zweryfikować poprawność jego działania w symulacji.

in1	in2	Out
0	0	1
0	1	1
1	0	1
1	1	0

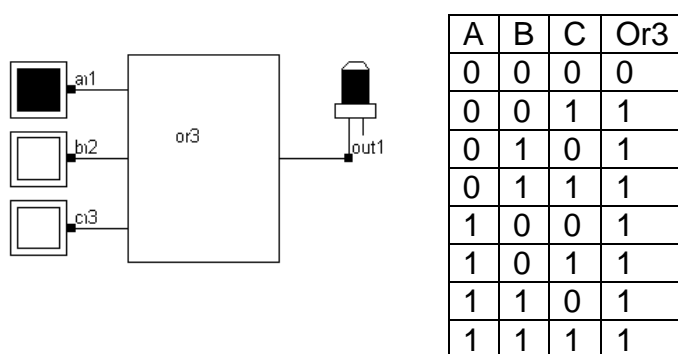


### 1.4. Bramki AND, OR, NOR, XOR

W podobny sposób jak opisany wcześniej, należy zbudować układy zawierające bramki AND, OR, NOR oraz XOR.

## 1.5. 3-wejściowa bramka OR

Tablica prawdy trójwejściowej bramki OR i schemat układu do symulacji są pokazane poniżej. Do zbudowania układu należy wykorzystać trójwejściową bramkę NOR oraz układ inwertera.



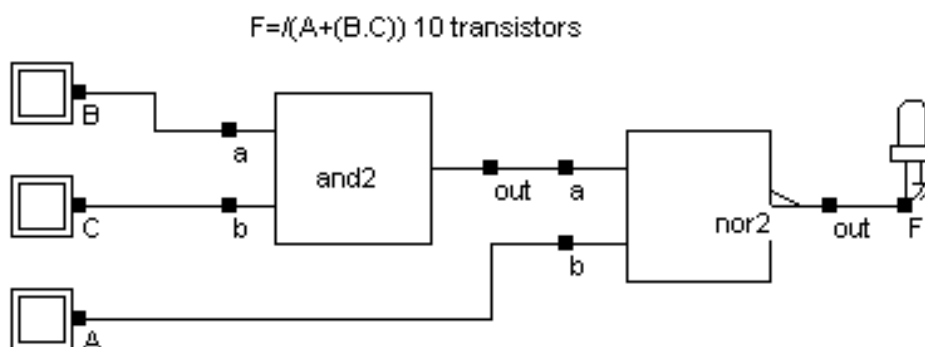
## 1.6. Realizacja złożonych funkcji logicznych

Jedną z możliwości realizacji złożonych funkcji logicznych jest zastosowanie kombinacji bramek AND i OR.

Ilustrację układu zbudowanego z kilku podstawowych bramek stanowi realizacja funkcji logicznej opisanej równaniem:

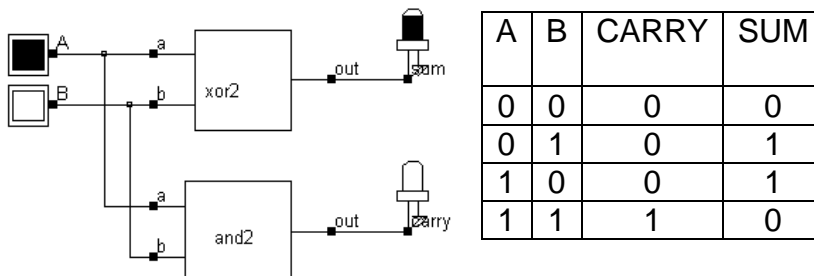
$$F = \neg(A + (B \cdot C))$$

Układ logiczny odpowiadający powyższej funkcji jest pokazany poniżej. Zbudowany jest on z bramki NOR oraz AND. Jego działanie należy zweryfikować w symulacji.



## 1.7. Półsumator

Układ sumatora i tablica prawdy są pokazane poniżej. Funkcja SUM jest zrealizowana za pomocą bramki XOR, funkcja CARRY z wykorzystaniem bramki AND. Poprawne działanie należy zweryfikować w symulacji.



## 1.8. Pełny sumator jednobitowy

Tablica wejść/wyjść sumatora bitowego przedstawiona jest poniżej. Warto zauważyć, że od strony logicznej, układ ten posiada trzy równoważne wejścia oraz jedno wyjście dwubitowe. Działanie tego układu polega na zliczaniu jedynek logicznych obecnych na wejściach oraz podawaniu ich liczby na wyjściu, w postaci liczby binarnej w kodzie naturalnym. Wyjście SUM to młodszy a CARRY to starszy bit tej liczby.

A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Proszę zbudować układ takiego sumatora i sprawdzić symulacyjnie jego działanie. Samodzielne wyprowadzenie i zaimplementowanie w postaci bramek logicznych funkcji SUM i CARRY, będzie dodatkowo punktowane.

## 1.9. Sumator 4-bitowy

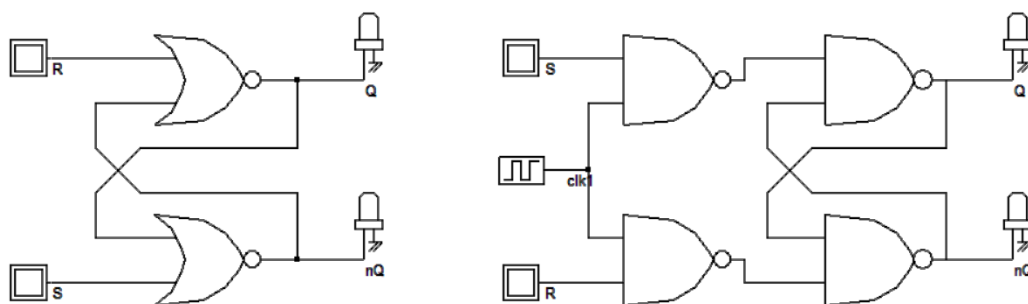
Proszę zbudować i sprawdzić działanie sumatora 4-bitowego złożonego z sumatorów jednobitowych, opracowanych w poprzednim zadaniu. Sumator 1-bitowy należy wyposażyć w widok symbolu i z jego wykorzystaniem hierarchicznie zbudować sumator 4-bitowy. Do sprawdzenia działania tego sumatora proszę wykorzystać elementy biblioteczne, takie jak zadajnik oraz wyświetlacz liczb binarnych.

## 2.1. Multiplexer i demultiplexer 4-bitowy

Proszę opracować zestaw składający się z multiplexera i demultiplexera 4-bitowego, wyłącznie z wykorzystaniem inwerterów i dwu- lub trzywejściowych bramek statycznych CMOS. Po opracowaniu widoku symboli dla układów, należy umieścić je w jednym schemacie i symulacyjnie prześledzić ich wspólne działanie, jako pełnego systemu MUX/DEMUX.

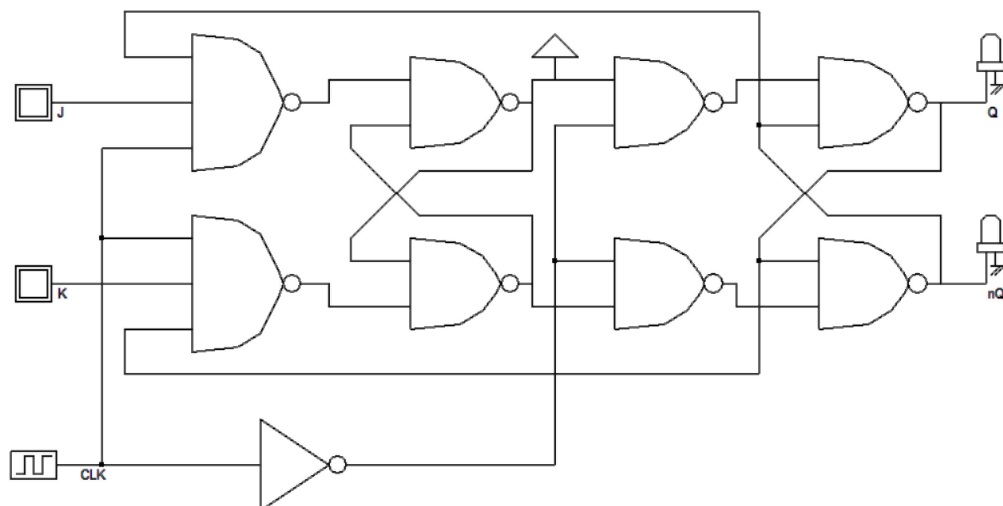
## 2.2. Przerzutniki RS

Proszę zbudować poniżej przedstawione przerzutniki RS w wersji asynchronicznej i synchronicznej, a następnie symulacyjnie zbadać poprawność ich działania.

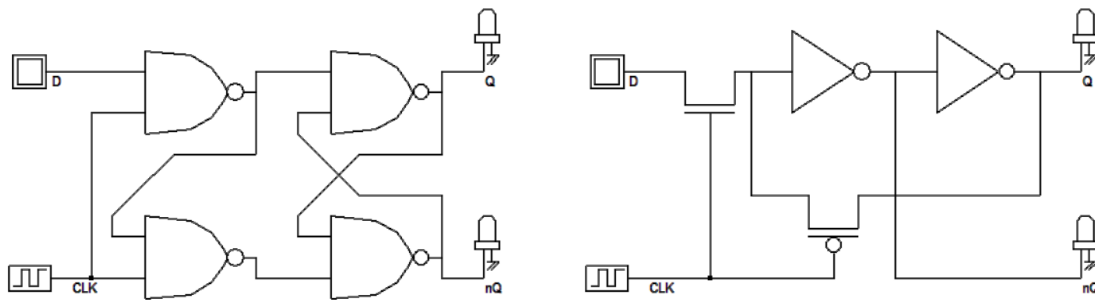


## 2.3. Przerzutnik synchroniczny JK w układzie Master-Slave

Proszę zbudować i symulacyjnie przebadać przedstawiony poniżej przerzutnik JK. Symbol źródła napięcia obecny w tym schemacie nie jest częścią składową tego komponentu. Zastosowanie go to jedynie zabieg formalny, umożliwiające prawidłowe zainicjalizowanie się przerzutnika po rozpoczęciu symulacji. Poza tym, element ten nie wpływa na działanie narysowanego przerzutnika.



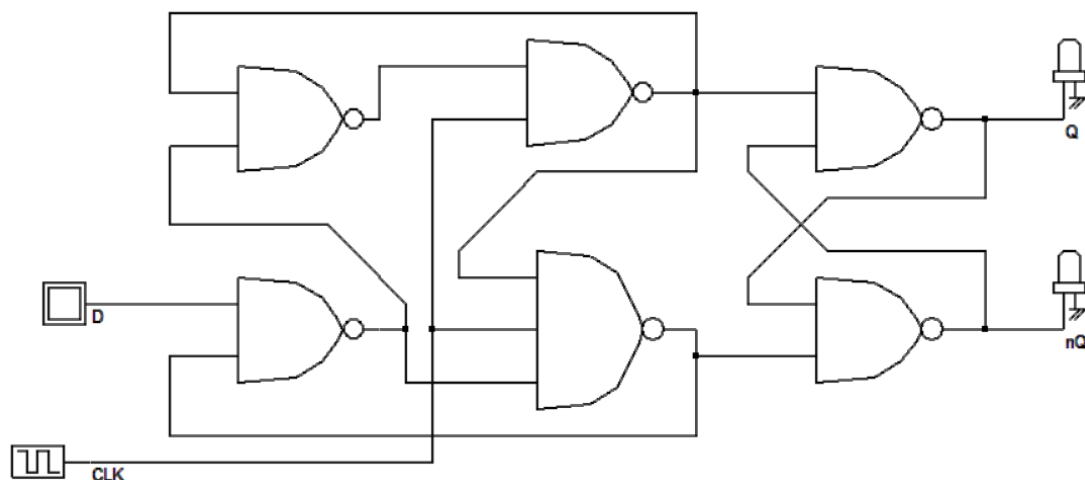
## 2.4. Przerzutnik D, w wersji statycznej oraz transmisyjnej



Proszę zaimplementować poniższe układy przerzutnika synchronicznego i symulacyjnie porównać ich działanie. Proszę przeanalizować działanie wersji transmisyjnej przerzutnika i wskazać potencjalne zagrożenie dla jego prawidłowego działania, wynikające ze sposobu działania zastosowanych bramek transmisyjnych.

## 2.5. Przerzutnik D wyzwalany zboczem

Proszę zbudować poniższy układ przerzutnika D a następnie symulacyjnie porównać jego działanie ze strukturami przebadanymi w poprzednim punkcie.



## 3.1. Trzybitowy licznik synchroniczny

Korzystając z przerzutników typu D proszę zaprojektować układ licznika synchronicznego z synchronicznym resetem o zadanej sekwencji cyfr. Należy zastosować komponent wyświetlacza siedmiosegmentowego dla czytelnej prezentacji działania opracowanego układu.

## ZADANIA – UKŁADY NA POZIOMIE TRANZYSTORÓW

### 4.1. Inwerter CMOS

Proszę zbudować standardowy statyczny inwerter CMOS składając go z pojedynczych tranzystorów MOS oraz symulacyjnie sprawdzić jego działanie.

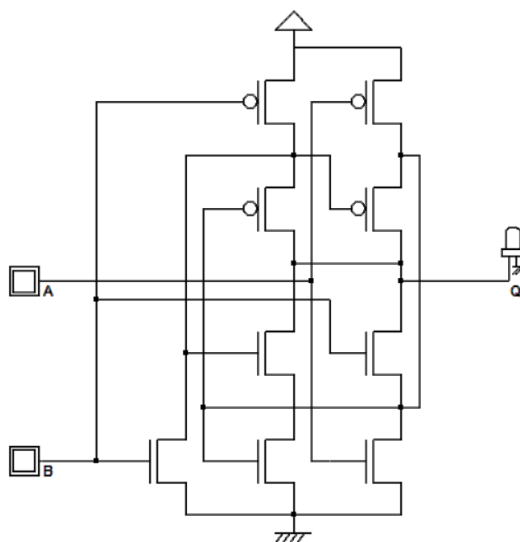
### 4.2. Dwuwejściowe statyczne bramki logiczne CMOS

Proszę zbudować z tranzystorów standardowe dwuwejściowe bramki NAND i NOR w technologii CMOS oraz symulacyjnie sprawdzić ich działanie. Opracowując struktury układów należy oprzeć się na materiale z wykładu oraz ogólnodostępnych źródłach wiedzy.

Po sprawdzeniu działania opracowanych bramek, należy każdą z nich wyposażyć w widok symbol, wstawić opracowany symbol do nowego schematu i ponownie sprawdzić działanie każdej z opracowanych bramek.

### 4.3. Bramka XOR, jako statyczna bramka logiczna CMOS

Proszę narysować i symulacyjnie przebadать działanie bramki XOR przedstawionej na poniższym rysunku a następnie przeanalizować bieg sygnałów wejściowych w tym układzie.



### 5.1. Transmisyjna bramka XOR

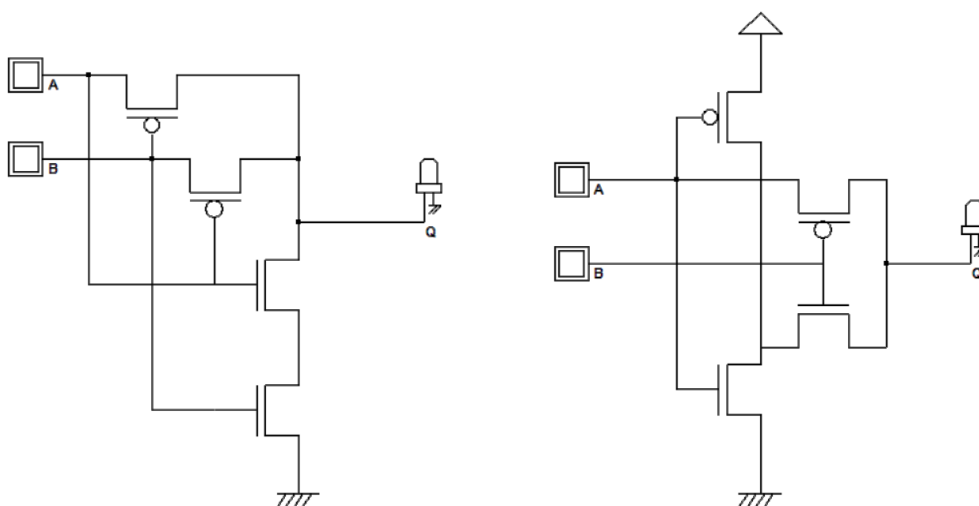
W przeciwieństwie do układów logiki statycznej CMOS, w których wyjście jest zwierane z masą lub zasilaniem, w bramkach transmisyjnych wyjście jest łączone z wyjściem lub od niego rozłączane.



Proszę narysować na poziomie tranzystorów bramkę XOR z elementem transmisyjnym oraz symulacyjnie sprawdzić jej działanie. Tablica wejść/wyjść bramki XOR jest następująca:

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Schematy bramek XOR do wykonania zamieszczone są poniżej.



Aby zapewnić prawidłowy przebieg symulacji układów transmisyjnych CMOS w programie DSCH, należy tranzystory NMOS i PMOS umieszczać tak, aby ich dreny znajdowały się po stronie wejściowej bramek transmisyjnych.

Analiza powyższych układów ma obejmować znalezienie potencjalnych punktów ryzyka dla ich poprawnego działania, wynikających ze specyfiki działania zastosowanych bramek transmisyjnych.

## 5.2. Multiplexer i demultiplexer 4-bit., wersja transmisyjna

Proszę opracować zestaw multiplexera i demultiplexera 4-bitowego. W projekcie można wykorzystać wyłącznie bramki transmisyjne CMOS oraz inwertery. Zestaw należy umieścić w jednym schemacie i prześledzić wspólne działanie elementów, jako pełnego systemu MUX/DEMUX.

**Koniec instrukcji**