Ćwiczenie 1

Stworzyć moduły, każdy w oddzielnych plikach, które będą opisywały następujące bramki:

NOT

AND / NAND 2-wejściowe

OR / NOR 2-wejściowe

XOR / XNOR 2-wejściowe

AND / NAND 3-wejściowe

OR / NOR 3-wejściowe

XOR / XNOR 3-wejściowe

Do każdego modułu stworzyć moduł testbenchu, który dokona symulacji stworzonego modułu. Należy sprawdzić wszystkie możliwe kombinacje.

Ćwiczenie 2

Stworzyć moduły, każdy w oddzielnych plikach, które będą opisywały następujące przerzutniki:

D master-slave wyzwalany zboczem (dowolne) z resetem asynchronicznym

T master-slave wyzwalany zboczem (dowolne) z resetem asynchronicznym

RS master-slave wyzwalany zboczem (dowolne) z resetem asynchronicznym

JK master-slave wyzwalany zboczem (dowolne) z resetem asynchronicznym

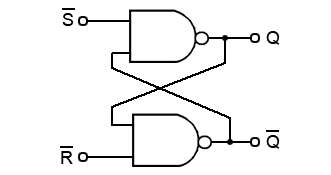
Przerzutniki powinny być wykonane z wykorzystaniem odpowiednich bramek (**opis strukturalny**). Niedozwolone jest wykorzystanie bezpośredniego opisu behawioralnego przerzutnika. Do każdego modułu stworzyć moduł testbenchu, który dokona symulacji stworzonego modułu. Należy sprawdzić wszystkie możliwe kombinacje.

W celu ułatwienia sprawdzania przerzutników wskazane jest etapowe tworzenie przerzutnika:

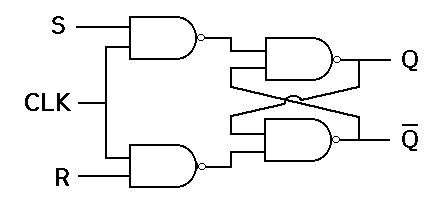
1. Przerzutnik asynchroniczny RS
2. Modyfikacja przerzutnika asynchronicznego RS na asynchroniczny D i T
3. Przerzutnik synchroniczny D, T i RS
4. Dodanie asynchronicznego resetu
5. Przerzutnik master-slave jako kaskadowe połączenie przerzutnika D/T/RS z RS

Przerzutnik JK należy wykonać oddzielnie z uwagi na sprzężenia zwrotne i brak łatwego łączenia kaskadowego przerzutników synchronicznych.

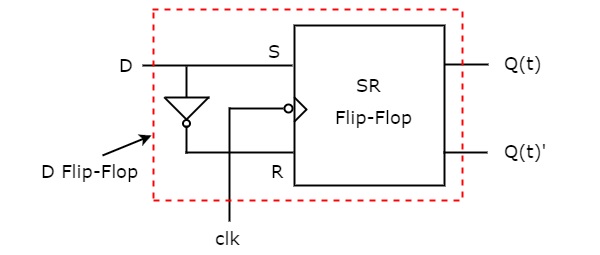
Przerzutnik RS asynchroniczny



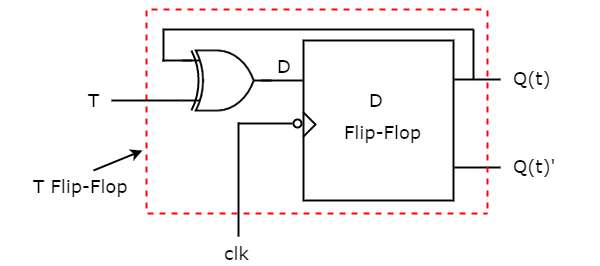
Przerzutnik RS synchroniczny



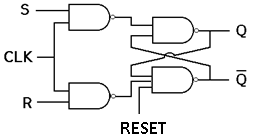
Przerzutnik D



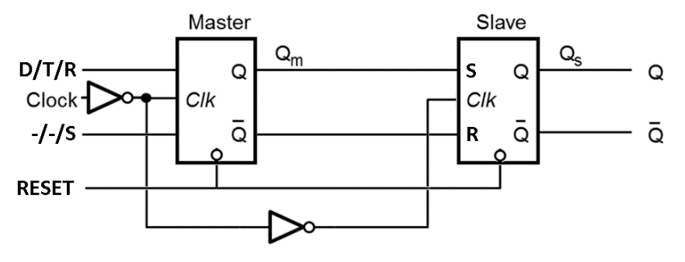
Przerzutnik T



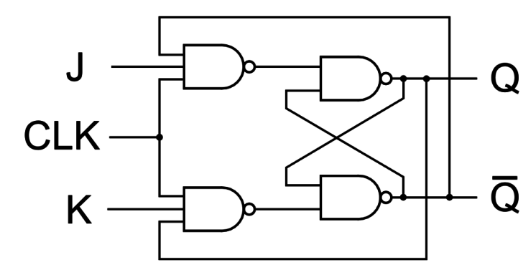
Asynchroniczny reset



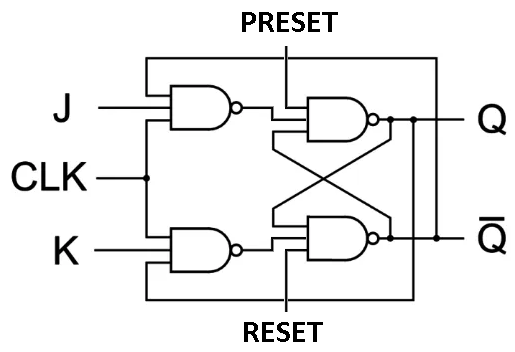
Przerzutnik master-slave z resetem asynchronicznym



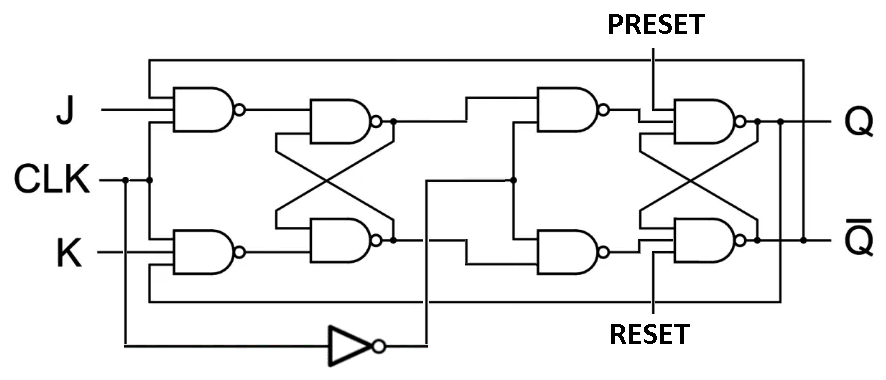
Przerzutnik JK synchroniczny



Przerzutnik JK synchroniczny z resetem asynchronicznym



Przerzutnik JK master-slave z resetem asynchronicznym



Ćwiczenie 3

Stworzyć moduł licznika 4-bitowego opartego na przerzutnikach D/T wyzwalanych zboczem (master-slave) z asynchronicznym resetem zliczającego w zadanej konfiguracji.

Do zaprojektowania należy stworzyć tablice przejścia dla każdego z bitów licznika (jeden przerzutnik odpowiada za jeden bit). Następnie należy zapisać funkcję przejścia minimalizując ją metodą tablicy Karnaugh.

Uwaga: Jeśli licznik nie wykorzystuje wszystkich stanów to należy uwzględnić przejście z tych stanów do pierwszego prawidłowego. Np. licznik 2-12 powinien posiadać możliwość przejścia ze stanów 0,1,13-15 do stanu 2.

Przykład:

Licznik na przerzutnikach D/T zliczający w konfiguracji 2-12.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (Q3Q2Q1Q0)t  stan poprzedni | | (Q3Q2Q1Q0)t+1  Stan następny | | T3T2T1T0 | D3D2D1D0 |
| Wymagane stany na wejściach przerzutników | |
| 0 | 0000 | 2 | 0010 | 0010 | 0010 |
| 1 | 0001 | 2 | 0010 | 0011 | 0010 |
| 2 | 0010 | 3 | 0011 | 0001 | 0011 |
| 3 | 0011 | 4 | 0100 | 0111 | 0100 |
| 4 | 0100 | 5 | 0101 | 0001 | 0101 |
| 5 | 0101 | 6 | 0110 | 0011 | 0110 |
| 6 | 0110 | 7 | 0111 | 0001 | 0111 |
| 7 | 0111 | 8 | 1000 | 1111 | 1000 |
| 8 | 1000 | 9 | 1001 | 0001 | 1001 |
| 9 | 1001 | 10 | 1010 | 0011 | 1010 |
| 10 | 1010 | 11 | 1011 | 0001 | 1011 |
| 11 | 1011 | 12 | 1100 | 0111 | 1100 |
| 12 | 1100 | 2 | 0010 | 1110 | 0010 |
| 13 | 1101 | 2 | 0010 | 1111 | 0010 |
| 14 | 1110 | 2 | 0010 | 1100 | 0010 |
| 15 | 1111 | 2 | 0010 | 1101 | 0010 |

Kolorem jasnoszarym zaznaczone są przejścia ze stanów, których licznik nie korzysta. Jak łatwo zauważyć, wartości wejść przerzutnika D są „kopią” stanu następnego. Wartości wejść przerzutnika T są natomiast funkcją XOR stanów poprzednich i następnych.

Minimalizacja funkcji przerzutnika T0:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (Q3Q2)t \ (Q1Q0)t | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 1 | 1 |

Minimalizację wykonujemy poprzez tworzenie możliwie jak największych sklejeń do momentu aż wszystkie jedynki zostaną uwzględnione. Każda jedynka może być użyta w wielu sklejeniach. Jest to nawet wskazane z uwagi na eliminację hazardu.

Można również wykonać minimalizację poprzez sklejanie zer, pamiętając o postaci funkcji, która wtedy będzie iloczynem sum. Wybór powinien zależeć od złożoności funkcji, im prostsza tym lepiej (w ćwiczeniu nie będzie to brane pod uwagę).

Ćwiczenie 4

Stworzyć moduł licznika 4-bitowego (identycznego jak w ćwiczeniu 3) za pomocą opisu behawioralnego. Zapoznać się z przypisaniami blokującymi oraz nieblokującymi. Można wykorzystać dowolne bloki instrukcji oraz pętle. Poniżej znajduje się przykładowy opis licznika 4-bitowego:

module behav\_counter( d, clk, clear, load, up\_down, qd);

// Port Declaration

input [7:0] d;

input clk;

input clear;

input load;

input up\_down;

output [7:0] qd;

reg [7:0] cnt;

always @ (posedge clk)

begin

if (!clear)

cnt <= 8'h00;

else if (load)

cnt <= d;

else if (up\_down)

cnt <= cnt + 1;

else

cnt <= cnt - 1;

end

assign qd = cnt;

endmodule