Motorola Science Cup²
Zadania II Etapu

Symulator układów logicznych

OMÓWIENIE

Symulator ma służyć jako pomoc naukowa lub narzędzie do projektowania układów dla hobbystów.

Jako rozwiązanie zadania prosimy o przesłanie gotowej do uruchomienia aplikacji oraz wszystkich plików źródłowych. Źródła możecie udostępnić jako archiwum załączone do maila, umieścić w serwisie chmurowym (Dysk Google, OneDrive, etc) lub w systemie kontroli wersji (GitHub, BitBucket, etc).

Rozwiązania desktopowe będą testowane w systemie Windows 10 i do pracy w takim systemie powinny być skompilowane. W przypadku aplikacji mobilnych testować będziemy w symulatorze Androida pod kontrolą systemu w wersji 10. Serwery aplikacji webowych będą uruchamiane pod kontrolą systemu Windows 10, a same aplikacje będą testowane w przeglądarce Firefox w wersji 94 lub nowszej.

CO OCENIAMY

1. Zgodność rozwiązania ze specyfikacją (zobacz opis etapów zadania)
2. Zaimplementowaną architekturę aplikacji (**0-10**)
3. Dokumentację projektu (**0-5**)
4. Zaproponowany interfejs aplikacji. Jego wygląd, czytelność i ogólne wrażenie użytkownika z użytkowania aplikacji (**0-20**)
5. Testy automatyczne (**0-5**)

SPECYFIKACJA

Po uruchomieniu aplikacja powinna pozwolić na utworzenie nowego projektu - musi zapytać o jego nazwę - lub otworzyć już istniejący projekt.

Ekran symulatora powinien składać się z pięciu obszarów.

1. U góry menu programu
    a. Zapis aktualnego układu jako gotowy klocek
    b. Opcje programu
    c. Powrót do okna wyboru projektu
2. Na dole zasobnik dostępnych klocków
3. W centrum obszar roboczy
4. Obszar wejścia po lewej a wyjścia po prawej stronie obszaru roboczego

Układ elementów na ekranie jest sugerowany, ale może być zmieniony. Najlepsze rozwiązania UX zostaną docenione dodatkowymi punktami (maksymalnie **10**)

ETAPY ZADANIA

Bramki logiczne

Za ten etap zadania uzyskacie do **30** punktów.

Nowy projekt powinien zawierać w sobie dwa startowe klocki - bramkę NOT i AND

Użytkownik dodaje pole w obszarze wejścia a jego stan logiczny (1 lub 0) ustala klikając w pole. Zmiana stanu wejścia powinna być możliwa w każdej chwili projektowania układu. Powinna również istnieć możliwość nadania nazwy pola wejściowego. Podobnie klikając w obszarze wyjścia, użytkownik może dodać pole wyjściowe którego stan (1 lub 0) jest zależny od układu klocków i ich połączeń w obszarze roboczym. Klocki różnią się kolorem i wielkością. Nie jest wymagane, aby klocki bramek logicznych miały kształt ich symbolu. Klocki umieszcza się na obszarze roboczym wybierając je z zasobnika.

Pola wejścia i wyjścia można łączyć z klockami za pomocą linii realizujących połączenia elektryczne.

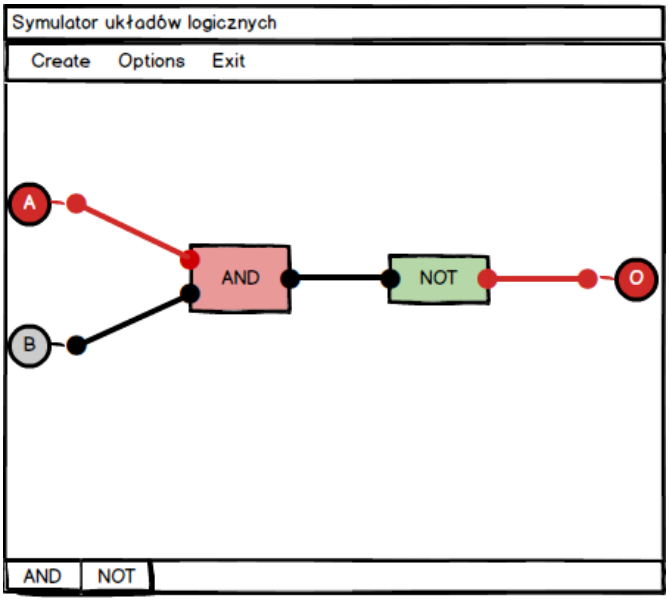
Wszystkie klocki można dowolnie przenosić w ramach obszaru roboczego i nie ma to wpływu na połączenia pomiędzy nimi. Klocki można również usuwać. Podobnie przenosić i usuwać można pola wejściowe i wyjściowe.

W opcjach programu powinna znaleźć się możliwość pokazania / ukrycia opisów przyłączeń.

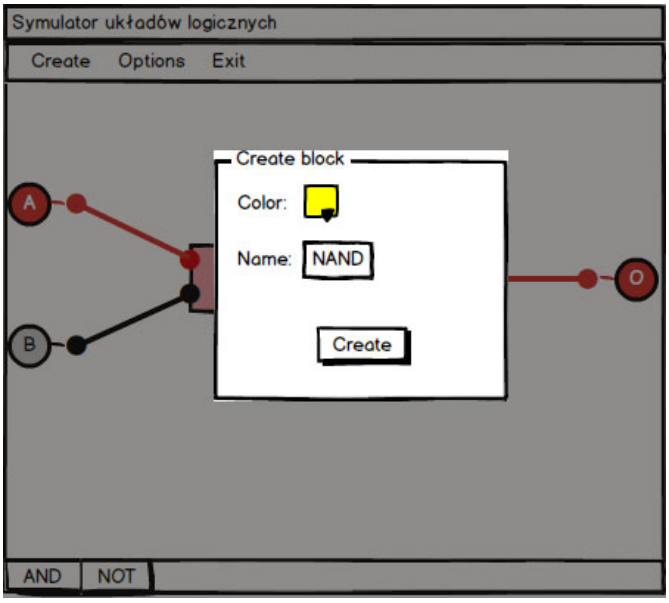
Połączenia można zaznaczać i usuwać. W jednym punkcie wejścia układu rozpocząć może się dowolna liczba połączeń. Do jednego punktu wejścia bramki, przyłączone może być tylko jedno połączenie. Połączenia można również rozgałęziać, ale nie mogą być łączone. Również skrzyżowanie połączeń nie spowoduje ich łączenia.

Przykłady prowadzenia połączeń zobaczycie na kolejnych rysunkach.

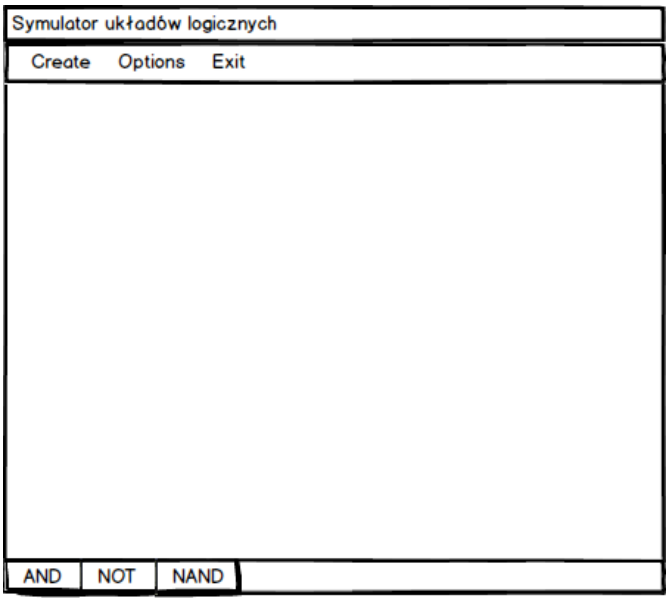
Oto przykład układu realizującego bramkę NAND



Stworzony układ można następnie zapisać jako kolejny klocek nadając mu nazwę, a program powinien wylosować dla niego kolor. Użytkownik powinien mieć możliwość wyboru innego koloru niż zaproponowany.

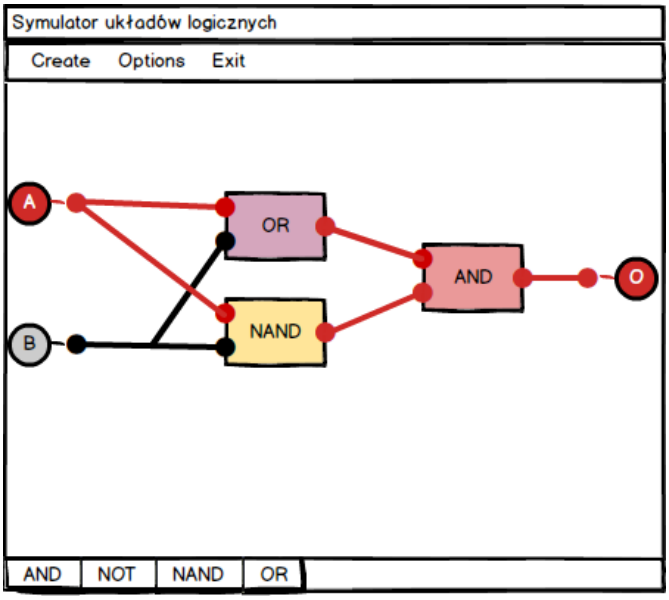


Utworzony klocek pojawia się w zasobniku, a obszar roboczy zostaje wyczyszczony.



Pozwala to budować coraz bardziej skomplikowane układy.

Poniżej układ realizujący bramkę XOR - w zasobniku znajdują się utworzone wcześniej bramki NAND i OR.



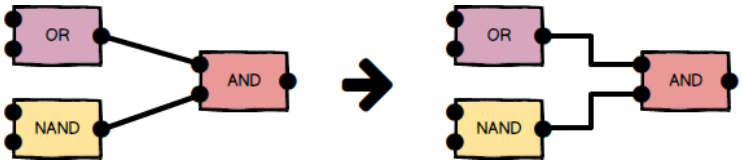
Zaawansowane sposoby wyświetlania połączeń

Ten etap wart jest do **20** punktów

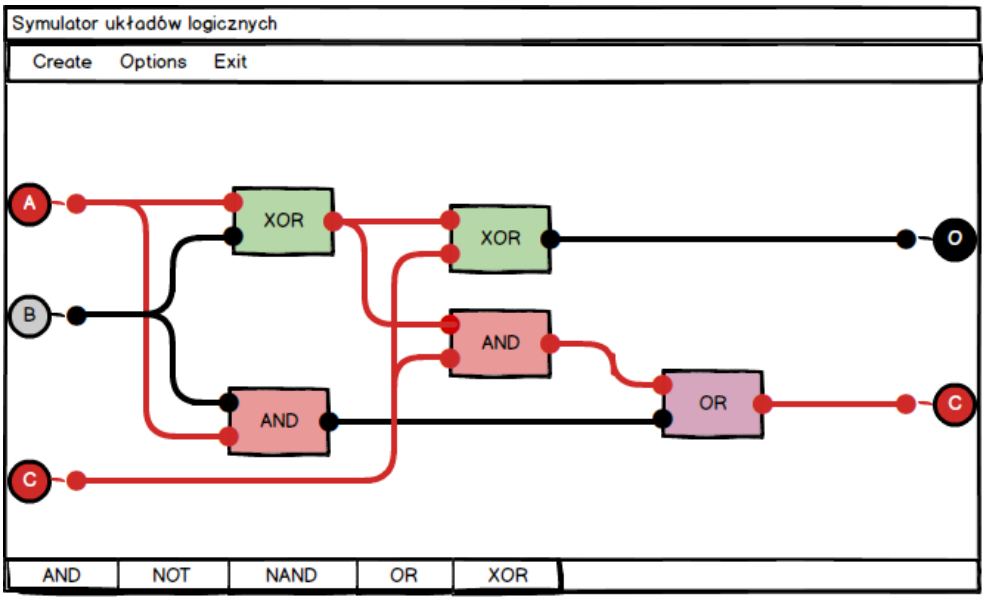
Klocki i pola przyłączeniowe powinny być rozkładane na wirtualnej siatce.

Połączenia - zwłaszcza rozgałęziające się - tworzą trudną do analizy plątanin. Symulator powinien rysować je w uporządkowany sposób.

Zamiast linii ukośnej program powinien rysować linię łamaną. Narożniki mogą być zaokrąglone - będą ładniej wyglądać.



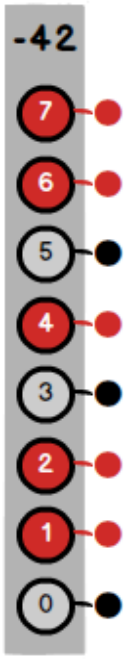
Oto układ sumatora bitowego realizujący powyższe założenia.



Edycja klocków i wielobitowe elementy wejścia / wyjścia

Za zrealizowanie poniższych wymagań możecie zyskać do **50** punktów

Symulator powinien pozwalać na edycję klocków. Pozycje poszczególnych elementów i połączeń mogą być odtworzone - nie muszą być identyczne jak przy tworzeniu klocka.



Zamiast pojedynczego, jednobitowego elementu symulator powinien udostępniać dwu, cztero i ośmiobitowe elementy wejścia/wyjścia. Powinny mieć one odpowiednią ilość punktów przyłączeniowych oraz wyświetlacz z aktualną wartością elementu. Dodatkowo powinny mieć możliwość przełączenia w tryb liczb ze znakiem, wtedy najstarszy bit staje się bitem znaku.