PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA
im. Stanisława Staszica w Pile
INSTYTUT POLITECHNICZNY
KIERUNEK ELEKTROTECHNIKA

Szymon Wierzchucki

Nr albumu 15006

ANALIZA SYMULACYJNA WYBRANYCH UKŁADÓW CYFROWYCH

SIMULATION ANALYSIS OF SELECTED DIGITAL CIRCUITS

Praca inżynierska
Napisana pod kierunkiem
prof. zw. dr hab. inż. Krzysztof Zawirski

Piła, 2019 r.

O Ś W I A D C Z E N I E

Ja, niżej podpisany……

(imię i nazwisko studenta)

student Państwowej Wyższej Szkoły Zawodowej im. Stanisława Staszica w Pile,

kierunek: ……

specjalność: ……

nr albumu: ……

oświadczam, że przedłożona praca dyplomowa pt.: …….

……

……

której jestem autorem:

1. Została/y przygotowana przeze mnie samodzielnie**, co oznacza, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.
2. Nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2018 r. poz. 1191 z późn. zm.) oraz dóbr osobistych chronionych prawem.
3. Nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony.
4. Nie była podstawą nadania dyplomu uczelni wyższej lub tytułu zawodowego ani mnie, ani innej osobie.

Ponadto oświadczam, że treść pracy przedstawionej przeze mnie do obrony, zawarta na przekazywanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu dyplomu zostanie cofnięta.

……, dn. …… …….

podpis studenta

**uwzględniając merytoryczny wkład promotora (w ramach prowadzonego seminarium dyplomowego)

Streszczenie pracy dyplomowej*

Temat pracy dyplomowej

Analiza symulacyjna wybranych układów cyfrowych

Imię i nazwisko autora pracy: Szymon Wierzchucki
Nr albumu: 15006
Imię i nazwisko promotora pracy: Krzysztof Zawirski
Słowa kluczowe (max. 10, charakterystyczne nazwy i określenia będące przedmiotem pracy):
symulacja, analiza, układy cyfrowe, Digital Works, konwerter kodu, rejestr przesuwny, licznik, multiplekser

Treść streszczenia (max. 1000 znaków; główne treści, tezy, wyniki i wnioski):

W niniejszej pracy dyplomowej przedstawiono analizę symulacyjną wybranych układów cyfrowych, takich jak: konwerter kodu BCD na kod 7 – segmentowy, rejestr przesuwający dwukierunkowy, licznik prosty, licznik rewersyjny, multiplekser – generator. Symulacja powyższych układów została wykonana w darmowym programie Digital Works, dla którego opisano szczegółową instrukcję obsługi, a także omówiono kilka darmowych programów alternatywnych, przeznaczonych do symulacji układów elektronicznych: CircuitMod, LogicCircuit, Logisim i Atanua. Przedstawione układy cyfrowe zostały zbudowane z podstawowych bramek logicznych i przerzutników, a ich działanie zademonstrowano na przebiegach sygnałów, tabelach oraz dodano komentarz słowny. Dodatkowo, pokazano rzeczywiste układy scalone znajdujące się w programie Digital Works, które są rozbudowanymi odpowiednikami symulowanych układów. Większość układów scalonych znajdujących się w programie posiada dołączoną dokumentację techniczną (*datasheet*).

* w przypadku prac dyplomowych w języku obcym należy dołączyć drugi egzemplarz dokumentu z tematem pracy dyplomowej, streszczeniem w języku pracy

PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA im. Stanisława Staszica w Pile

Piła, dnia 28.02.2019 r.

Instytut Politechniczny

Zakład Elektrotechniki i Elektroniki

Temat pracy dyplomowej inżynierskiej

Student: **Szymon Wierzchucki**

Kierunek: **Elektrotechnika** Studia: **Stacjonarne**

Specjalność: **Systemy Automatyki i Elektroniki**

Temat pracy: **Analiza symulacyjna wybranych układów cyfrowych**

1. Założenia i dane wyjściowe:

Celem pracy jest analiza symulacyjna i opracowanie zestawu materiałów pomocniczych do nauki i demonstracji multimedialnej działania wybranych złożonych układów cyfrowych, takich jak, liczniki proste i rewersyjne, rejestry przesuwające, multipleksery, konwertery kodów, itp.

2. Zadania szczegółowe:

- wybór programu symulacyjnego dostępnego darmowo w Internecie;
- zapoznanie się z rozwiązaniami (budową) wybranych układów cyfrowych na podstawie literatury;
- utworzenie i uruchomienie modeli zaprojektowanych układów;
- opracowanie dydaktyczne prezentowanych zagadnień;
- przygotowanie odpowiednich materiałów pomocniczych do dydaktyki służących do prezentacji na wykładzie (PowerPoint) i umieszczenia na stronie WWW.

3. Miejsce przeprowadzania prac laboratoryjnych, konstrukcyjnych:

   Laboratoria Instytutu Politechnicznego

4. Termin oddania pracy: 28.02.2019 r.
5. Kierujący pracą (promotor): prof. zw. dr hab. inż. Krzysztof Zawirski

Promotor Dyrektor Instytutu

SPIS TREŚCI

1. WSTĘP

W dzisiejszych czasach programy symulacyjne mają ogromne znaczenie m.in. w laboratoriach badawczych, biurach projektowych, uczelniach i szkołach. Pozwalają na przeprowadzanie badań lub eksperymentów bez specjalistycznych i często kosztownych urządzeń lub przyrządów. Rzeczywiste urządzenia i przyrządy są zastępowane odpowiednimi blokami w programie symulacyjnym, które symulują ich działanie na podstawie równań matematycznych, metod numerycznych i złożonych algorytmów. Przed skonstruowaniem układu elektronicznego często istnieje potrzeba sprawdzenia samej idei jego działania lub prototypu, dlatego znajomość programów przeznaczonych do symulowania układów elektronicznych jest bardzo pomocna w zawodowej pracy inżyniera elektronika.

Celem niniejszej pracy dyplomowej jest analiza symulacyjna wybranych złożonych układów cyfrowych takich jak: rejestry przesuwające, liczniki proste i rewersyjne, multipleksery, konwertery kodów itp. oraz przygotowanie materiałów pomocniczych do nauki i demonstracji multimedialnej działania wyżej wymienionych układów. Założeniem głównym jest wybór darmowego programu symulacyjnego, który umożliwi ich symulację. Podczas wyboru odpowiedniego programu kierowano się głównie funkcjonalnością i prostotą obsługi, a także zgodnością ze współczesnymi systemami operacyjnymi. Ostatecznie wybrano program Digital Works, który spełnia te założenia. Należy wspomnieć, że w Internecie istnieje wiele darmowych programów, które potrafią zasymulować działanie elektroniki cyfrowej, analogowej i mieszanej. Wybór odpowiedniego symulatora zależy od preferencji użytkownika, ponieważ nie istnieją darmowe programy, w których jest możliwość zasymulowania każdego układu.

Niniejsza praca dyplomowa składa się z pięciu rozdziałów. Rozdziały od drugiego do czwartego są rozdziałami głównymi. W drugim rozdziale zostały omówione alternatywne programy przeznaczone do symulacji układów cyfrowych. W trzecim rozdziale opisana jest instrukcja obsługi programu Digital Works. Pokazano tam jego funkcje i możliwości. Czwarty rozdział przedstawia analizę symulacyjną konwertera kodu BCD na 7 – segmentowy, rejestru przesuwnego dwukierunkowego, licznika prostego i rewersyjnego oraz multipleksera - generatora. Ostatnia część pracy to podsumowanie wyników symulacji i wyciągnięcie wniosków.

2. PRZEGLĄD DARMOWYCH PROGRAMÓW DO SYMULACJI UKŁADÓW CYFROWYCH

2.1. CircuitMod

CircuitMod jest programem napisanym w języku Java, udostępnionym na licencji GNU GPL (*General Public License*) więc jest wolnym i otwartym oprogramowaniem. Ostatnia wersja programu o oznaczeniu v2.7 została wydana w 2007 roku. Pomimo faktu, iż program nie jest już rozwijany, nadal można go bez problemu uruchomić na systemie operacyjnym Windows 10, Linux oraz Mac OS X. Program pozwala na symulację układów cyfrowych, a także obwodów prądu zmiennego i stałego. Znajdują się w nim zarówno podstawowe komponenty elektroniczne takie jak rezystory, kondensatory, cewki, diody, tranzystory, jak również elementy bardziej zaawansowane np.: wzmacniacze operacyjne, memrystory, diody tunelowe, linie transmisyjne, timery, przerzutniki Schmitta itp. Tworzenie układu elektronicznego polega na wybieraniu komponentów z listy, nanoszeniu ich na obszar roboczy i odpowiednim łączeniu. Następnie dokonuje się modyfikacji poprzez odpowiednie skróty klawiaturowe. CircuitMod posiada około 280 predefiniowanych schematów w celu nauki podstaw elektroniki i elektrotechniki. Dużą zaletą tego programu jest symulacja w czasie rzeczywistym. Można zaobserwować konwencjonalny lub niekonwencjonalny ruch ładunków elektrycznych poprzez wirtualne przewody i komponenty. Prędkość ruchu tych ładunków jest proporcjonalna do natężenia prądu płynącego w danej gałęzi obwodu i ustawia się ją poprzez suwak umieszczony w prawej części okna. W podobny sposób dynamicznie modyfikuje się wartości potencjometrów i innych komponentów o zmiennych parametrach głównych, a także jest możliwość zmiany szybkości symulacji i czasu jej kroku. Przebiegi podstawowych wielkości elektrycznych takich jak napięcie, natężenie prądu czy moc można obserwować na wirtualnym oscyloskopie umieszczonym w dolnej części okna. Po najechaniu kursorem na dowolny punkt przebiegu wyświetla się wartość chwilowa danej wielkości elektrycznej. Wartości i parametry wyświetlane na wirtualnym oscyloskopie mogą być dowolnie zmieniane. Dodatkowo, oscyloskop może pracować w trybie XY [3].

2.2. LogicCircuit

LogicCircuit jest programem edukacyjnym służącym do symulacji wyłącznie układów cyfrowych, udostępnionym na licencji GNU GPL. Posiada intuicyjny interfejs

graficzny pozwalający m.in. na: tworzenie nieograniczonej hierarchii obwodów za pomocą wielobitowych magistral, podgląd sygnałów wejściowych i wyjściowych na wirtualnym oscyloskopie oraz używanie wyświetlaczy segmentowych i matryc LED. Niewątpliwą zaletą jest fakt, że program jest stale rozwijany. Aby budować układy cyfrowe należy „przeciągać" komponenty z lewej części okna programu na obszar roboczy i łączyć je ze sobą klikając w ich wyprowadzone węzły. Program pozwala na tworzenie własnych złożonych układów cyfrowych zamykając je w jednym bloku z wyprowadzeniami co znacznie poprawia estetykę i czytelność schematu. W programie znajdują się takie elementy jak: interaktywne przyciski, porty wejścia – wyjścia, sensory (elementy generujące określoną lub losową sekwencję bitów), zegar, brzęczyk, diody LED, matryce i wyświetlacze LED, wyświetlacze graficzne, wszystkie bramki logiczne z opcją zwiększania liczby wejść, pamięć ROM i RAM. Podgląd wyników symulacji umożliwia próbnik stanów logicznych i wspomniany wcześniej oscyloskop. Imponującą funkcją jest niewątpliwie generator tablic prawdy. Jest to bardzo przydatne narzędzie do nauki podstaw elektroniki pomagające zrozumieć m.in. prawa De Morgana. Tworzenie tablicy prawdy jest możliwe poprzez skonstruowanie układu logicznego z bramek logicznych, a także za pomocą odpowiednich wyrażeń w pasku filtrowania wyrażeń boolowskich. Warto wspomnieć, że LogicCircuit posiada możliwość pisania skryptów w konsoli IronPython co znacznie poszerza możliwości tego programu [6].

2.3. Logisim

Logisim to program symulacyjny oparty na licencji GNU GPL, przeznaczony do projektowania i symulowania cyfrowych układów logicznych. Jego obsługa jest łatwa, ponieważ posiada prosty i intuicyjny interfejs graficzny. Dzięki możliwości budowania rozbudowanych obwodów z mniejszych podukładów może być używany do projektowania i symulacji mało skomplikowanych procesorów w celach edukacyjnych. Logisim jest używany przez studentów uczelni wyższych na całym świecie poczynając od budowy prostych układów logicznych, a kończąc na wielotygodniowych kursach z zakresu architektury komputerowej. Program przestał być rozwijany na czas nieokreślony od 11 października 2014 r., aczkolwiek działa na dowolnym systemie operacyjnym obsługującym Java 5. Podczas symulacji przewody oznaczone odpowiednimi kolorami w zależności od panującego na nich sygnału pomagają w odnajdywaniu błędów w układzie. Połączenia osadzają się automatycznie w pionie i w poziomie, dzięki temu łączenie komponentów jest bardzo szybkie i wygodne. Gotowe schematy układów można

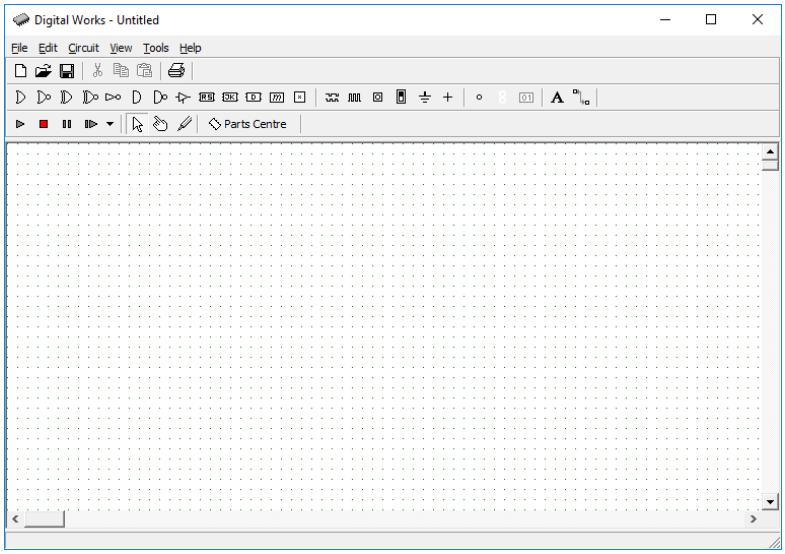
wyeksportować do pliku *gif*, wydrukować do pliku *pdf* lub na drukarce. W programie istnieją takie komponenty elektroniczne jak: moduły wejść i wyjść, bramki logiczne z możliwością ustawienia czasu propagacji sygnału, multipleksery, układy arytmetyczne, przerzutniki i pamięć RAM. Logisim posiada moduł analizy kombinacyjnej, który umożliwia konwersję między obwodami, tabelami prawdy i wyrażeniami logicznymi [2].

2.4. Atanua

Atanua to interaktywny program oparty na licencji GNU GPL, zaprojektowany w taki sposób, aby pomóc w nauce podstawowej logiki boolowskiej i elektroniki. Wykorzystuje przyspieszenie sprzętowe OpenGL. Według programisty, niestandardowy interfejs użytkownika ma na celu jak najszybszą naukę programu, pozwalając uczniom skoncentrować się na nauce przedmiotu zamiast spędzać czas na uczeniu się jego obsługi. Program zawiera trzy podstawowe sekcje: lista komponentów, obszar roboczy i przyciski z funkcjami typu zapisywanie i wczytywanie. Tworzenie obwodu polega na wybieraniu komponentów z listy i „przeciąganiu" ich na obszar roboczy. Następnie łączy się poszczególne elementy w funkcjonalną całość. Program zapewnia współpracę z takimi systemami operacyjnymi jak Windows, Linux i Mac OS X. Symulacja odbywa się w czasie rzeczywistym, a częstotliwość taktowania zegara może wynosić do 1000 Hz. W programie do wyboru są przerzutniki, multipleksery i najczęściej używane bramki logiczne zarówno o symbolice amerykańskiej jak i fińskiej. Dużą zaletą programu jest posiadanie rzeczywistych modeli układów cyfrowych z rodziny 74xx i możliwość symulacji mikrokontrolera 8051. Wejścia układów cyfrowych mogą być obsługiwane z przycisków klawiatury dzięki ich odpowiednikom w programie (zarówno litery jak i cyfry). Wyniki symulacji można obserwować na diodach i macierzach LED, wyświetlaczach 7 i 16 – segmentowych, a także na próbniku sygnałów logicznych. Tak jak w większości programów tego typu, istnieje możliwość tworzenia podukładów i łączenia ich z innymi tworząc zaawansowane obwody cyfrowe. Natychmiastowe sprzężenie zwrotne daje informacje o tym, w których miejscach obwodu jest stan wysoki lub niski. Ciekawostką jest fakt, że istnieje narzędzie przeznaczone tylko dla nauczycieli, które według autora programu ma za zadanie uniemożliwiać oszukiwanie przy odrabianiu prac domowych na tym programie przez uczniów [5].

3. PROGRAM DIGITAL WORKS

Digital Works (Rysunek 3.1) jest programem symulacyjnym opartym na licencji GNU GPL. Można go pobrać ze strony *www.mecanique.co.uk*. Jego autorem jest David John Barker, który jest także założycielem Mecanique – sklepu internetowego oferującego moduły i części elektroniczne. Digital Works posiada przejrzysty interfejs graficzny umożliwiający łatwe konstruowanie elektronicznych układów cyfrowych i ich analizę. Układy te mogą być budowane z podstawowych bramek logicznych (AND, OR, NAND, NOR, XOR, XNOR, NOT) i przerzutników (D, RS i JK). Dostępne są bramki trójstanowe, pamięci RAM i ROM umożliwiające tworzenie cyfrowych systemów magistralowych. W programie jest zaimplementowana funkcja wykrywająca hazard logiczny, a także nieprawidłowości wynikające z nieodpowiedniego połączenia bramek trójstanowych. Sygnały wejściowe mogą być wprowadzane poprzez przełączniki, zegary, generatory sekwencji bitowych i interaktywne przyciski. Istnieje możliwość podglądu przebiegów sygnałów w narzędziu *Logic History* (oscyloskop). W zakładce *Part Centre* znajdują się biblioteki rzeczywistych urządzeń i komponentów elektronicznych produkowanych przez firmy Motorola i Philips takich jak liczniki, rejestry przesuwne, rejestry pamiętające, kontrolery itp. Jedną z lepszych funkcji Digital Works jest możliwość tworzenia makr i umieszczenia ich w centrum części. Ta opcja pozwala na utworzenie nowego elementu logicznego złożonego z mniejszych podukładów i połączenie go z innymi elementami. Do utworzonego komponentu cyfrowego można dodać dokumentację w celu szybkiego podglądu nazw wyprowadzeń i ich funkcji [1].



Rysunek 3.1 Okno główne programu Digital Works

3.1. Podstawowe narzędzia

Pasek narzędzi programu Digital Works posiada zestaw ikon, które mogą być użyte w celu edytowania obszaru roboczego, zapisywania i otwierania plików oraz drukowania schematów [1].

New Circuit (Rysunek 3.2) – czyści obszar roboczy, umożliwiając stworzenie nowego obwodu logicznego. Jeżeli bieżący obwód uległ zmianie od czasu ostatniego zapisu, wyświetli się okno dialogowe z zapytaniem o nadpisanie tego pliku. Należy wybrać *Yes*, aby zapisać lub *No*, aby wyczyścić obszar roboczy bez zapisywania lub *Cancel*, by powrócić do bieżącego projektu [1].

Rysunek 3.2 Ikona *New Circuit* [1]

Open Circuit (Rysunek 3.3) – umożliwia otwarcie uprzednio zapisanego obwodu [1].

Rysunek 3.3 Ikona *Open Circuit* [1]

Save Circuit (Rysunek 3.4) – zapisuje bieżący obwód. Jeżeli układ nie został uprzednio zapisany, pojawi się okno dialogowe *Save As*. W przeciwnym przypadku projekt zostanie nadpisany [1].

Rysunek 3.4 Ikona *Save Circuit* [1]

Cut (Rysunek 3.5) – umożliwia usunięcie elementów znajdujących się w zaznaczonym obszarze. Należy zaznaczyć obszar, z którego mają być usunięte elementy i kliknąć *Cut*. Kliknięcie prawym przyciskiem myszy na wybranym obszarze lub osobnym komponencie umieszczonym w obszarze roboczym także wyświetla tę opcję [1].

Rysunek 3.5 Ikona *Cut* [1]

Copy (Rysunek 3.6) – umożliwia skopiowanie elementów znajdujących się w zaznaczonym obszarze. W tym celu należy zaznaczyć obszar, z którego mają być skopiowane elementy i kliknąć *Copy*, a następnie *Paste*, by umieścić zawartość ze schowka na obszarze roboczym [1].

Rysunek 3.6 Ikona *Copy* [1]

Paste (Rysunek 3.7) – umieszcza uprzednio skopiowaną zawartość na obszarze roboczym [1].

Rysunek 3.7 Ikona *Paste* [1]

Print Circuit (Rysunek 3.8) – wysyła obraz z obszaru roboczego do drukarki domyślnej. Automatycznie ustawiana jest orientacja pozioma papieru, którą można zmienić w opcjach wydruku [1].

Rysunek 3.8 Ikona *Pirnt Circuit* [1]

Object Selector (Rysunek 3.9) – zaznacza obiekt po kliknięciu na nim. Następnie może być przemieszczony poprzez przeciągnięcie go w inne miejsce obszaru roboczego. Niektóre elementy mogą być obrócone (np. komentarze, bramki logiczne) w momencie gdy pojawia się nad nimi okrąg z krzyżykiem w środku. Kursor przybierze kształt dłoni po umieszczeniu go nad tym okręgiem. Naciśnięcie lewego przycisku myszy i ruszenie kursorem obróci obiekt o pewien kąt. Istnieje możliwość zmiany ustawień obiektu poprzez kliknięcie na nim prawym przyciskiem myszy. Aby zaznaczyć więcej obiektów należy trzymać lewy przycisk myszy na obszarze roboczym i przeciągać kursor nad obszarem, który chcemy zaznaczyć. Pojawi się kropkowane obramowanie, które wyznacza granice zaznaczonego obszaru. Aby przenieść wszystkie elementy znajdujące się w tym obszarze, wystarczy kliknąć i przytrzymać jeden z nich i umieścić wszystkie w wybranym miejscu. W celu umieszczenia na obszarze roboczym wielu takich samych elementów, należy nacisnąć lewy *Control* na klawiaturze podczas wybierania elementu. Gdy obiekt jest zaznaczony, naciśnięcie klawisza *F1* na klawiaturze wyświetli tekst pomocy dotyczący tego obiektu. Usuwanie zaznaczonego obiektu odbywa się przez naciśnięcie klawisza *Delete* na klawiaturze, lecz trzeba pamiętać o tym, że nie można przywrócić tego elementu natychmiast kombinacją *Ctrl + Z* [1].

Rysunek 3.9 Ikona *Object Selector* [1]

Object Interaction Selector (Rysunek 3.10) – narzędzie do obsługi interaktywnych elementów takich jak przełączniki lub przyciski. Interaktywność nie działa gdy symulacja jest wyłączona. Aby zmienić stan interaktywnego elementu, należy kliknąć na nim lewym przyciskiem myszy podczas trwania symulacji [1].

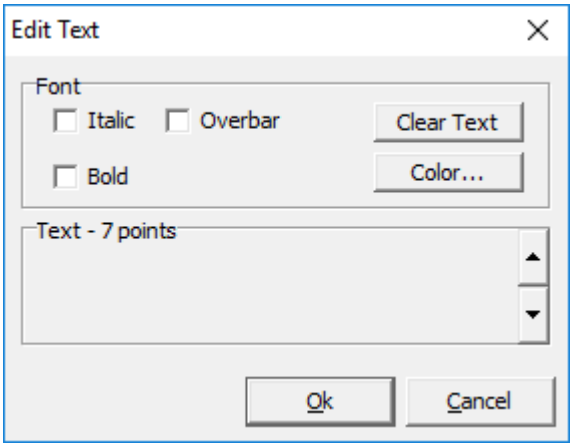
Rysunek 3.10 Ikona *Object Interaction Selector* [1]

Logic Probe (Rysunek 3.11) – umożliwia sprawdzenie stanów logicznych na wejściach lub wyjściach układów, a także na przewodach podczas symulacji. Próbnik stanów logicznych umieszcza się nad obiektem, a następnie w celu podglądu wartości sygnału klika się na nim lewym przyciskiem myszy [1].

Rysunek 3.11 Ikona *Logic Probe* [1]

Annotation (Rysunek 3.12) – narzędzie do umieszczania tekstu w obszarze roboczym. Aby dodać tekst, należy kliknąć w ikonę a następnie w miejsce gdzie ma być umieszczony. Automatycznie pojawi się okno dialogowe (Rysunek 3.13) z opcjami edycji. Można w nim ustawić czcionkę, zaznaczyć ustawienie znaku negacji (*Overbar*) i zmienić kolor [1].

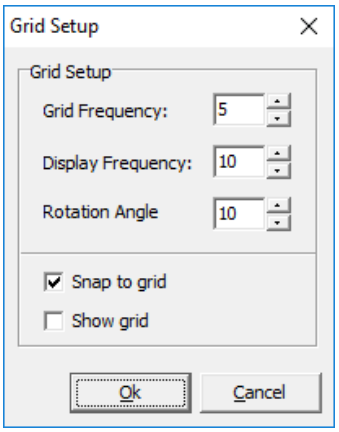
Rysunek 3.12 Ikona *Annotation* [1]



Rysunek 3.13 Okno dialogowe narzędzia Annotation

Grid Setup (Rysunek 3.14) – okno dialogowe pozwalające na konfigurację siatki obszaru roboczego. Wywołuje się go z menu głównego, klikając w zakładkę *View*, a następnie *Grid Setup*. Opcja *Grid Frequency* definiuje minimalną liczbę pikseli, z jaką będą się przesuwać elementy programu. Gdy ta liczba wynosi np. 5, oznacza to, że poruszany obiekt będzie przesuwał się co 5 pikseli lecz musi być zaznaczona opcja *Snap to grid*. W przeciwnym razie obiekt będzie przesuwany niezależnie od siatki. Opcja *Display frequency* definiuje sposób w jaki jest wyświetlana siatka. Przykładowo: jeżeli *Grid frequency* ma ustawioną wartość 5, a *Display frequency* 20, to obiekt przesuwa się co 5 pikseli, ale kropki siatki są rozmieszczone co 20 pikseli. Opcja *Rotation angle*

ustawia minimalny kąt obrotu elementu. Aby przesuwać obiekt co 1 piksel, należy używać strzałek na klawiaturze [1].



Rysunek 3.14 Okno dialogowe *Grid Setup*

3.2. Dodawanie obiektów i ich łączenie

Obiekty cyfrowe mogą być umieszczone w obszarze roboczym, a następnie połączone ze sobą poprzez narzędzie *Wiring Tool* (Rysunek 3.15) [1].

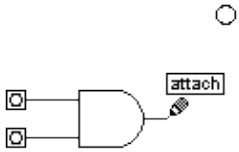
Rysunek 3.15 Ikona *Wiring Tool* [1]

3.2.1. Dodawanie obiektów

Istnieją dwa sposoby na umieszczenie w obszarze roboczym Digital Works komponentów wejściowych, wyjściowych i logicznych. Obiekty można wybierać z głównego paska narzędziowego lub z centrum części (*Parts Centre*). Aby dodać obiekt z głównego paska narzędziowego, należy kliknąć na pożądanej części, a następnie przenieść kursor na obszar roboczy i nacisnąć lewy przycisk myszy. W celu dodania elementu z centrum części, trzeba przeciągnąć go na obszar roboczy, a następnie zwolnić przycisk myszy [1].

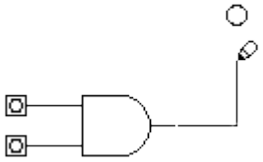
3.2.2. Prowadzenie przewodów

Po pierwsze, na obszar roboczy należy nanieść wszystkie obiekty, które mają być ze sobą połączone. Następnie, umieścić kursor nad punktem, od którego będzie rysowany przewód. Kursor zmieni swój kształt jeśli jest możliwe połączenie w tym miejscu (Rysunek 3.16). Kliknięcie lewym przyciskiem myszy rozpocznie procedurę łączenia [1].



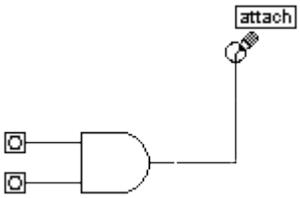
Rysunek 3.16 [1]

Nie trzeba prowadzić przewodu bezpośrednio od jednego punktu do kolejnego. Można kliknąć myszą w dowolne miejsce obszaru roboczego, aby powstała ścieżka połączeniowa, tak jak jest to pokazane na rysunku (Rysunek 3.17) [1].



Rysunek 3.17 [1]

Ostatecznie, umieszcza się kursor nad punktem zakończenia połączenia. Kursor ponownie zmieni swój kształt jeśli jest możliwość połączenia się z punktem (Rysunek 3.18). Kliknąć lewym przyciskiem myszy, aby zakończyć [1].

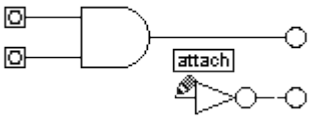


Rysunek 3.18 [1]

Należy pamiętać o tym, że proces łączenia dwóch elementów musi być zakończony przed przystąpieniem do innej czynności. Jeśli wybierze się inne narzędzie z menu podczas procesu łączenia, to zostanie on przerwany. Można anulować proces łączenia naciskając klawisz *Esc* lub *Delete* [1].

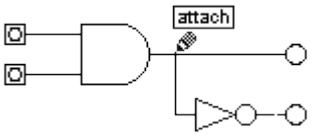
3.2.3. Łączenie obiektów

Przykładowo, do obwodu ma zostać dodany inwerter, po czym jego wejście ma być podłączone do wyjścia bramki AND. Należy odpowiednio ustawić elementy i poprowadzić przewód (Rysunek 3.19) [1].



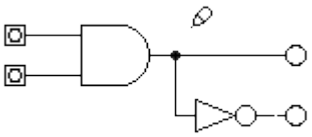
Rysunek 3.19 [1]

Następnie, przenieść kursor nad przewód, który jest podłączony do wyjścia bramki AND (Rysunek 3.20) [1].



Rysunek 3.20 [1]

Nacisnąć lewy przycisk myszy, aby zakończyć. Połączenie przewodu zostanie zaznaczone czarną kropką (Rysunek 3.21) [1].



Rysunek 3.21 [1]

Węzeł i ścieżka połączeniowa mogą być przemieszczane w inne miejsca, a także usuwane klawiszem *Delete*. Należy pamiętać o tym, że nie wolno łączyć ze sobą wyjść elementów elektronicznych, ponieważ grozi to zwarciem i ich uszkodzeniem, natomiast w programie pojawi się komunikat błędu. Wyjątkiem są bramki trójstanowe, których działanie zostanie opisane w dalszej części [1].

3.3. Przeglądanie przykładowych układów

Przed pierwszym użyciem programu Digital Works warto otworzyć jeden z przykładowych układów dostarczonych razem z programem w celu zapoznania się z ogólną zasadą jego działania. Przykładowe pliki są umieszczone w folderze *Sample Circuits*, który znajduje się w głównym katalogu programu. Na przykład: jeśli zainstalowano program w folderze o ścieżce: ‘*c:\Program Files\Digital Works’* to przykładowe pliki znajdują się w folderze o ścieżce: ‘*c:\Program Files\Digital Works\Sample Circuits’*. Należy wybrać *Open* z głównego paska narzędziowego, zmienić

folder na *Sample Circuits* i otworzyć plik. Następnie kliknąć *Run* w celu jego uruchomienia [1].

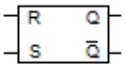
3.4. Przerzutniki

W programie znajdują się trzy podstawowe przerzutniki: RS (Rysunek 3.22), JK (Rysunek 3.23) i D (Rysunek 3.24). Znając podstawowe zasady projektowania układów sekwencyjnych można zbudować z nich inne – bardziej rozbudowane przerzutniki reagujące na odpowiedni sygnał zegarowy.

3.4.1. Przerzutnik RS

Tabela 3.1 Działanie przerzutnika RS w Digital Works[1]

R	S	Wyjścia
0	0	Bez zmian na wyjściu
0	1	Ustawianie $Q = 1, \bar{Q} = 0$
1	0	Resetowanie $Q = 0, \bar{Q} = 1$
1	1	Stan zabroniony. Symulacja przerwana.

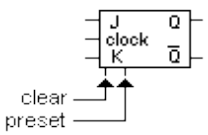


Rysunek 3.22 Przerzutnik RS w Digital Works [1]

3.4.2. Przerzutnik JK

Tabela 3.2 Działanie przerzutnika JK w Digital Works [1]

J	K	CLK	Wyjścia
0	0	1→0	Bez zmian na wyjściu
0	1	1→0	Resetowanie $Q = 0, \bar{Q} = 1$
1	0	1→0	Ustawianie $Q = 1, \bar{Q} = 0$
1	1	1→0	Przełączanie $Q = \bar{Q}$



Rysunek 3.23 Przerzutnik JK w Digital Works [1]

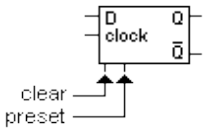
Stan wysoki na wejściu *Clear* bezwzględnie ustawia wyjście przerzutnika na stan niski, a stan wysoki na wejściu *Preset* bezwzględnie ustawia wyjście przerzutnika na stan

wysoki. Wejście *Preset* ma priorytet względem wejścia *Clear*. Taka sama zasada występuje w przerzutniku D [1].

3.4.3. Przerzutnik D

Tabela 3.3 Działanie przerzutnika D w Digital Works [1]

D	CLK	Wyjścia
0	0	Bez zmian na wyjściu
0	0→1	Resetowanie $Q = 0, \bar{Q} = 1$
1	0	Bez zmian na wyjściu
1	0→1	Ustawianie $Q = 1, \bar{Q} = 0$



Rysunek 3.24 Przerzutnik D w Digital Works [1]

3.5. Bramki logiczne

W Digital Works występują wszystkie podstawowe bramki logiczne oraz bramki trójstanowe. Każda podstawowa bramka logiczna może mieć maksymalnie cztery wejścia.

3.5.1. Bramka OR

Tabela 3.4 Działanie bramki OR w Digital Works [1]

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Rysunek 3.25 Symbol bramki OR w Digital Works [1]

3.5.2. Bramka NOR

Tabela 3.5 Działanie bramki NOR w Digital Works [1]

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Rysunek 3.26 Symbol bramki NOR w Digital Works [1]

3.5.3. Bramka XOR

Tabela 3.6 Działanie bramki NOR w Digital Works [1]

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Rysunek 3.27 Symbol bramki XOR w Digital Works [1]

3.5.4. Bramka XNOR

Tabela 3.7 Działanie bramki XNOR w Digital Works [1]

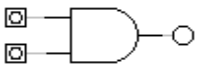
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Rysunek 3.28 Symbol bramki XNOR w Digital Works [1]

3.5.5. Bramka AND

Tabela 3.8 Działanie bramki AND w Digital Works [1]

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

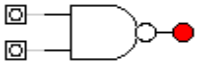


Rysunek 3.29 Symbol bramki AND w Digital Works [1]

3.5.6. Bramka NAND

Tabela 3.9 Działanie bramki NAND w Digital Works [1]

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

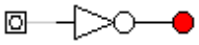


Rysunek 3.30 Symbol bramki NAND w Digital Works [1]

3.5.7. Bramka NOT

Tabela 3.10 Działanie bramki NOT w Digital Works [1]

A	Y
0	1
1	0



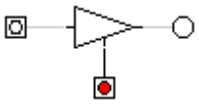
Rysunek 3.31 Symbol bramki NOT w Digital Works [1]

3.5.8. Bramka trójstanowa

Bramka trójstanowa (Rysunek 3.32) posiada specjalne wejście zwane *Enable*. Gdy na tym wejściu występuje stan wysoki to bramka powtarza na wyjściu stan wejścia. Natomiast gdy *Enable* przyjmuje stan niski, wyjście bramki trójstanowej jest fizycznie odłączone od wewnętrznego obwodu (wyjście pływające) [1].

Tabela 3.11 Działanie bramki trójstanowej w Digital Works [1]

A	Enable	Y
0	0	?
0	1	0
1	0	?
1	1	1



Rysunek 3.32 Symbol bramki trójstanowej w Digital Works [1]

3.6. Pamięć

Digital Works posiada moduł pamięci RAM i ROM. Liczba unikalnych lokalizacji w pamięci urządzenia może być określona poprzez ustawienie liczby linii adresowych (*adress lines*). Przykładowo, 4 linie adresowe zapewnią 2^4 unikalnych lokalizacji w pamięci, 8 linii adresowych zapewni 2^8 unikalnych lokalizacji w pamięci itd. Każda lokalizacja może przechowywać określoną liczbę bitów. Przykładowo, gdy moduł pamięci ma 4 linie danych to każda unikalna lokalizacja w pamięci może przechować 4 bity. Czyli do przechowania 1 bajta jest potrzebnych 8 linii danych. Podsumowując, całkowita pojemność pamięci w bitach jest wyrażona równaniem (3.1) [1]:

$$X = 2^n \cdot d \tag{3.1}$$

Gdzie:

X – pojemność całkowita pamięci [b]

n – liczba linii adresowych

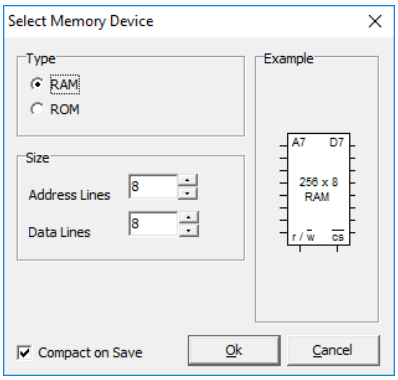
d – liczba linii danych

3.6.1. Dodawanie modułu pamięci

W celu dodania modułu pamięci RAM lub ROM w programie Digital Works należy kliknąć ikonę pamięci (Rysunek 3.33) na głównym pasku narzędziowym, a następnie kliknąć na obszarze roboczym [1].

Rysunek 3.33 Ikona *Memory* [1]

W kolejnym etapie wyświetli się okno dialogowe bloku pamięci (Rysunek 3.34) [1].

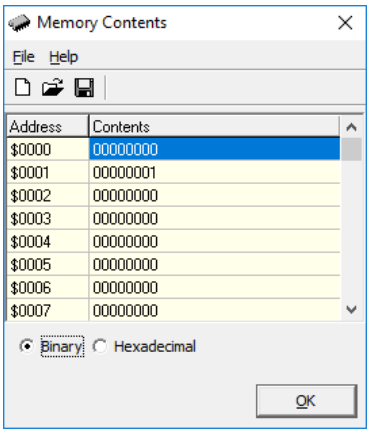


Rysunek 3.34 Okno dialogowe *Select Memory Device*

W powyższym oknie dialogowym można wybrać typ pamięci, liczbę linii adresowych i linii danych. Dla pamięci ROM istnieje tylko wejście $\overline{cs}$, które włącza lub wyłącza moduł. Jeżeli $\overline{cs}$ jest w stanie niskim, to moduł pamięci będzie włączony. Jeśli na tym wejściu będzie stan wysoki, to moduł pamięci będzie odłączony od magistrali danych. Dla pamięci RAM istnieje jeszcze dodatkowe wejście $r/\overline{w}$. Jeżeli na tym wejściu będzie stan wysoki, to pamięć będzie w trybie odczytywania danych. Jeśli będzie stan niski, to pamięć będzie w trybie zapisywania danych [1].

3.6.2. Edycja zawartości pamięci

Aby zmienić zawartość pamięci RAM lub ROM należy kliknąć prawym przyciskiem myszy na blok pamięci, a następnie wybrać *Edit Memory Contents* (Rysunek 3.35). Ta czynność jest możliwa nawet podczas trwania symulacji w celu podglądu aktualnych danych w pamięci. Zawartość pamięci może być wyświetlana w systemie binarnym lub szesnastkowym. Aby zmienić dane w adresie pamięci, należy wybrać odpowiedni wiersz w kolumnie *Contents* i kliknąć na nim lewym przyciskiem myszy lub wcisnąć klawisz *Enter*. Następnie usunąć bieżącą zawartość klawiszem *Backspace* i wpisać nowe dane odpowiednio w kodzie binarnym lub szesnastkowym [1].



Rysunek 3.35 Okno *Memory Contents*

Istnieje możliwość szybkiego wyzerowania zawartości adresów pamięci klikając na ikonę *Clear* (Rysunek 3.36) [1].

Rysunek 3.36 Ikona *Clear* [1]

Program umożliwia zapisanie konfiguracji pamięci do pliku z rozszerzeniem *.map (Rysunek 3.37). Można otworzyć taki plik klikając ikonę *Open* (Rysunek 3.38) [1].

Rysunek 3.37 Ikona *Save Memory Map* [1]

Rysunek 3.38 Ikona *Open Memory Map* [1]

3.7. Elementy wejściowe

W programie Digital Works sygnały wejściowe mogą być wprowadzane poprzez zegar, generator sekwencji bitowych, przełączniki, interaktywne przyciski i sygnały stałe. Każdy element wejściowy może mieć swoją etykietę w programie. Należy pamiętać o tym, że do interaktywnego przełączenia wszelkich przycisków służy narzędzie *Object Interaction Selector* [1].

3.7.1. Zegar

Zegar jest głównym komponentem taktującym w programie Digital Works. Generuje sygnał prostokątny o współczynniku wypełnienia równym 50% w zadanej częstotliwości. Wskaźnikiem jego działania jest dioda LED, która miga z częstotliwością taktowania. Przybiera kolor zielony gdy zegar jest w stanie wysokim, a kolor biały gdy jest w stanie niskim. Aby umieścić zegar na obszarze roboczym, należy kliknąć jego ikonę (Rysunek 3.39), a następnie kliknąć w dowolne miejsce obszaru roboczego. Częstotliwość taktowania zegara może być zmieniona w zakładce *Circuit,* wybierając opcję *Clock Speed* [1].

Rysunek 3.39 Ikona *Clock* [1]

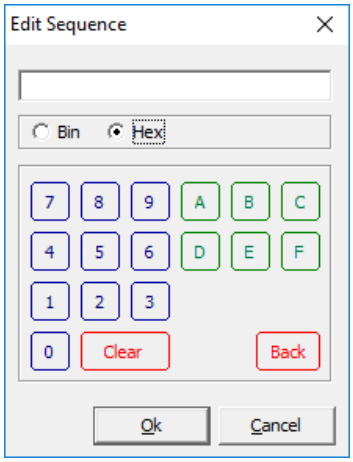
3.7.2. Generator sekwencji bitowych

Generator sekwencji bitowych jest używany w celu wygenerowania sygnału o określonej sekwencji bitowej. Wskaźnikiem jego pracy jest dioda LED, która miga w kolorze czerwonym jeśli bit ma wartość 1, a w kolorze białym jeśli bit ma wartość 0. Aby

umieścić ten obiekt na obszarze roboczym, należy kliknąć jego ikonę (Rysunek 3.40), a następnie kliknąć w dowolne miejsce obszaru roboczego [1].

Rysunek 3.40 Ikona *Sequence Generator* [1]

Klikając prawym przyciskiem myszy i wybierając *Edit Sequence* wyświetla się okno ustawień generatora (Rysunek 3.41) [1].



Rysunek 3.41 Okno *Edit Sequence*

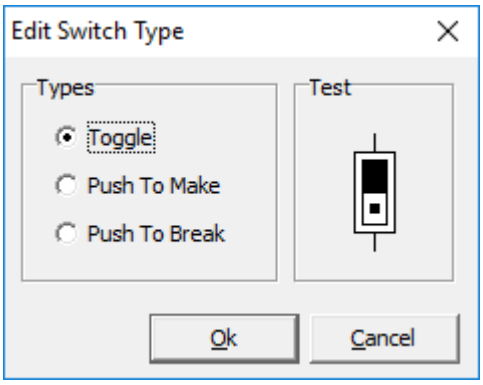
W edytorze można wpisać maksymalnie 64 – bitową sekwencję w postaci binarnej lub szesnastkowej. Wpisana sekwencja będzie powtarzała się cyklicznie [1].

3.7.3. Przełącznik

Aby umieścić przełącznik na obszarze roboczym, należy kliknąć jego ikonę (Rysunek 3.42), a następnie kliknąć w dowolne miejsce obszaru roboczego [1].

Rysunek 3.42 Ikona *Switch* [1]

Klikając prawym przyciskiem myszy na bloku przełącznika, ukaże się okno wyboru jego typu (Rysunek 3.43) [1].



Rysunek 3.43 Okno *Edit Switch Type*

Opcja *Toggle* zmienia przycisk na bistabilny. Każdorazowe kliknięcie zmienia jego stan na przeciwny. *Push To Make* zmienia przycisk na chwilowy, normalnie otwarty tzn. przycisk będzie w stanie aktywnym do momentu zwolnienia przycisku myszy. *Push To Break* zmienia przycisk na chwilowy, normalnie zamknięty tzn. przycisk będzie w stanie nieaktywnym do momentu zwolnienia przycisku myszy [1].

3.7.4. Przycisk interaktywny

Przycisk interaktywny służy do generowania sygnału wejściowego o wartości 0 lub 1. Po każdym kliknięciu zmienia się jego stan na przeciwny lecz może pracować także w trybie pulsującym zaznaczając w jego ustawieniach opcję *Pulse Input*. Gdy ma kolor czerwony, oznacza to że generuje stan wysoki, gdy ma kolor biały generuje stan niski. Aby umieścić przycisk interaktywny na obszarze roboczym, należy kliknąć w jego ikonę (Rysunek 3.44), a następnie w dowolne miejsce obszaru roboczego [1].

Rysunek 3.44 Ikona *Interactive Input* [1]

3.7.5. Sygnały stałe

Sygnałami stałymi w programie Digital Works są zasilanie (Rysunek 3.45) i masa (Rysunek 3.46). Aby umieścić zasilanie lub masę na obszarze roboczym, należy kliknąć w odpowiednią ikonę, a następnie w dowolne miejsce obszaru roboczego [1].

Rysunek 3.45 Ikona *Logic High* [1]

Rysunek 3.46 Ikona *Logic Low* [1]

3.8. Elementy wyjściowe

W programie Digital Works sygnały wyjściowe można obserwować na diodach LED, wyświetlaczu 7 – segmentowym, numerycznym wskaźniku stanu wyjść i oscyloskopie o nazwie *Logic History*. Każdy element wyjściowy może mieć swoją etykietę w programie [1].

3.8.1. Dioda LED

Dioda LED umożliwia podgląd stanu logicznego na wyjściu elementu w czasie rzeczywistym. Może być podłączona tylko do wyjścia komponentu. Aby dodać diodę LED do obszaru roboczego, należy kliknąć w jej ikonę (Rysunek 3.47), a następnie w

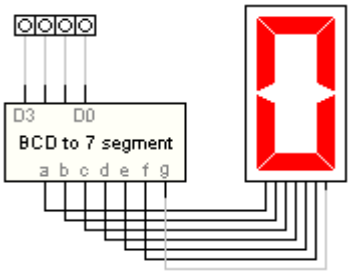
dowolne miejsce obszaru roboczego. W celu zmiany koloru diody LED, należy kliknąć na niej prawym przyciskiem myszy i wybrać polecenie *Change Color* [1].

Rysunek 3.47 Ikona *Light Emitting Diode* [1]

W trakcie trwania stanu wysokiego dioda LED świeci wybranym kolorem, a w trakcie trwania stanu niskiego świeci na biało. Jeżeli dioda jest podłączona do wyjścia bramki trójstanowej to stan *floating* będzie oznaczony innym kolorem [1].

3.8.2. Wyświetlacz 7 – segmentowy

Wyświetlacz 7 – segmentowy służy do monitorowania wyjść układu w postaci dziesiętnej. Zazwyczaj jest używany z konwerterem BCD (*Binary Coded Decimal*) na kod 7 – segmentowy (Rysunek 3.48) [1].



Rysunek 3.48 Wyświetlacz 7 – segmentowy z dedykowanym konwerterem kodu [1]

Aby umieścić wyświetlacz na obszarze roboczym, należy kliknąć w jego ikonę (Rysunek 3.49), a następnie w dowolne miejsce obszaru roboczego [1].

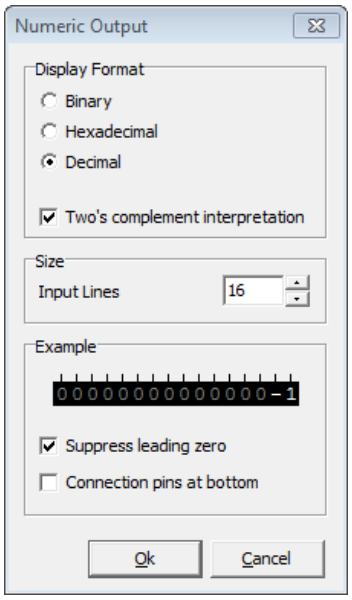
Rysunek 3.49 Ikona *Seven Segment LED* [1]

Po kliknięciu prawym przyciskiem myszy na wyświetlaczu, pojawi się okienko umożliwiające zmianę koloru wyświetlania cyfr, a także zmianę rodzaju wyświetlacza - tzn. ze wspólną anodą czy katodą [1].

3.8.3. Numeryczny wskaźnik stanu wyjść

Numeryczny wskaźnik stanu wyjść służy do pokazywania informacji wyjściowych urządzenia w postaci binarnej, szesnastkowej i dziesiętnej. Aby umieścić go na obszarze roboczym, należy kliknąć w jego ikonę (Rysunek 3.50), a następnie w dowolne miejsce obszaru roboczego. Po kliknięciu pojawi się okno dialogowe z ustawieniami (Rysunek 3.51) [1].

Rysunek 3.50 Ikona *Numeric Output Device* [1]



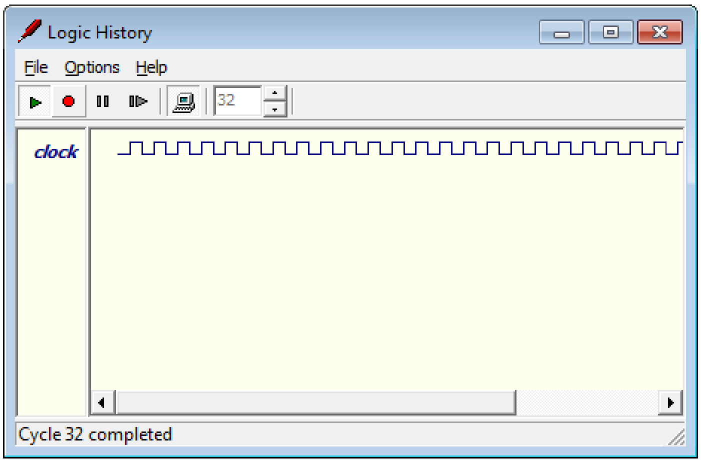
Rysunek 3.51 Okno dialogowe *Numeric Output*

W tym oknie istnieje możliwość ustawienia formatu pokazywanych danych, liczby linii wejściowych (maksymalnie 16), opcji wyświetlania zer w przypadku braku danych, a także umiejscowienia wyprowadzeń (góra lub dół). Jeśli wybrana jest opcja wyświetlania danych w formacie dziesiętnym, to możliwe jest zdecydowanie czy urządzenie ma interpretować wartość binarną na liniach wejściowych jako uzupełnienie do dwóch (wyświetlanie liczb ujemnych). Maksymalna wielkość wyświetlanej liczby jest definiowana przez liczbę linii wejściowych. Przykładowo dla liczb dodatnich, jeżeli jest 16 linii wejściowych, to największa wyświetlana liczba będzie wynosić: $2^{16} - 1$. W przypadku wyświetlania liczb ujemnych maksymalny zakres będzie się mieścił w przedziale od -2^{n-1} do $2^{n-1} - 1$, gdzie n to liczba linii wejściowych [1].

3.8.4. Oscyloskop

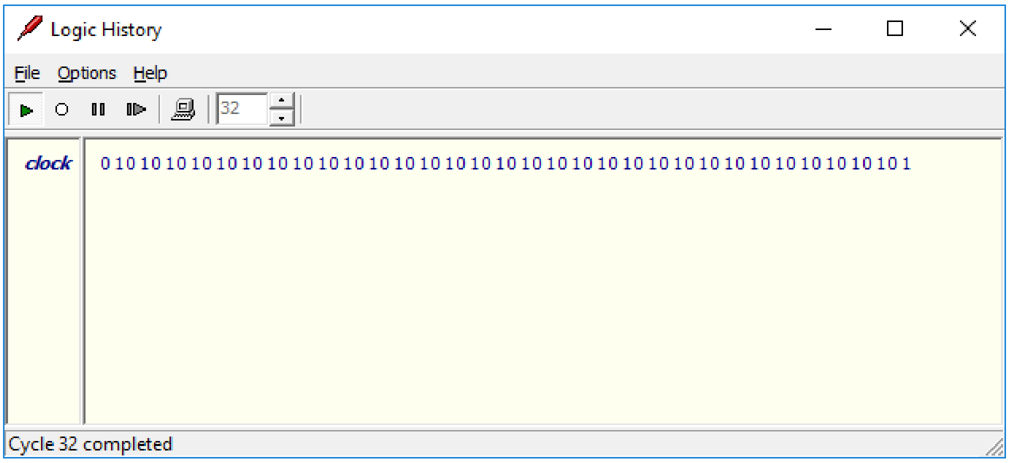
Oscyloskop (*Logic History)* (Rysunek 3.52) umożliwia podgląd przebiegów zdefiniowanych sygnałów wejściowych i wyjściowych. Jest to szczególnie przydatne narzędzie w celu zrozumienia zasady działania układów cyfrowych sekwencyjnych. Aby dodać sygnał do podglądu na oscyloskopie, należy kliknąć prawym przyciskiem myszy na obiekcie generującym lub wskazującym sygnał (*Sequence Generator*, *Clock, Interactive Input*, *LED*) i zaznaczyć opcję *Add to Logic History* [1]. Kolejność wyświetlanych sygnałów (patrząc od góry) jest następująca: zegar, sygnały wejściowe, sygnały wyjściowe. Uporządkowanie poszczególnych sygnałów wejściowych i wyjściowych na oscylogramie zależy od kolejności tworzenia obiektów generujących lub wskazujących sygnał, tzn. sygnał pierwszego obiektu generującego (np. włącznika

podłączonego do wejścia układu scalonego) będzie umieszczony najwyżej w stosunku do sygnałów później utworzonych obiektów generujących. Analogicznie jest z sygnałami wyjściowymi.



Rysunek 3.52 Okno *Logic History*

Na pasku narzędziowym znajdują się takie opcje jak: start (*run*), stop, pauza (*pause*) i krok (*step*). Przebiegi mogą być wyświetlane w postaci zer i jedynek (Rysunek 3.53) oraz istnieje możliwość ustawienia liczby cykli zegara. Przebieg może być generowany co połowę cyklu zegarowego lub pełen cykl zegarowy [1].

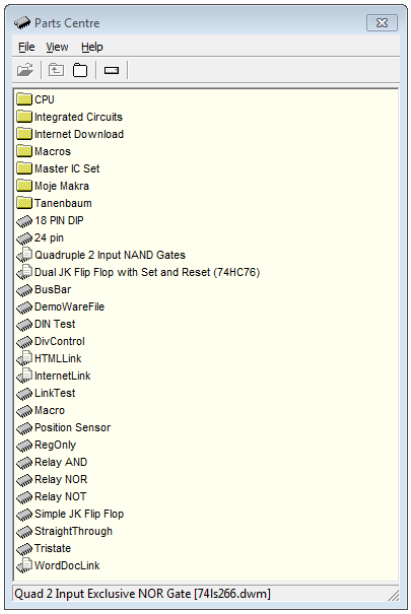


Rysunek 3.53 Przebieg sygnału w postaci cyfrowej

3.9. Centrum części

W centrum części (*Part* Centre) (Rysunek 3.54) znajdują się wszystkie złożone elementy i urządzenia dostępne w programie, które zostały zbudowane z podstawowych komponentów. Narzędzie jest umiejscowione na głównym pasku narzędziowym. Do wyboru są także komponenty firmy Philips i Motorola posiadające dokumentację techniczną (*datasheet*). Jeżeli wybrany komponent z centrum części posiada dokumentację, wyświetli się komunikat o jej dostępności tuż po umieszczeniu

komponentu na obszarze roboczym. Aby szybko otworzyć dokumentację urządzenia, należy go zaznaczyć i wcisnąć klawisz *F1*. Z poziomu okna *Part Centre* można edytować makra [1].



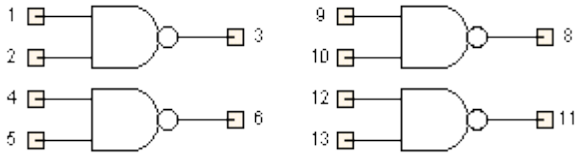
Rysunek 3.54 *Part Centre*

3.10. Tworzenie własnych układów

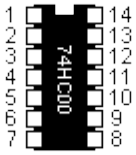
W celu zwiększenia możliwości programu Digital Works zostało stworzone narzędzie umożliwiające tworzenie własnych układów, które mogą być zapisane na dysku w postaci makr. Dzięki temu narzędziu zyskuje się czytelność schematu, ponieważ nie trzeba łączyć za każdym razem wielu elementów tworzących rozbudowane urządzenie lecz raz zapisane na dysku wywołuje się ponownie [1].

3.10.1. Tworzenie makra

Obwód zaprojektowany w programie Digital Works może wiązać się z dwoma stanami wizualnymi. Są one nazywane widokami implementacji (Rysunek 3.55) i interfejsu (Rysunek 3.56). Widok implementacji ukazuje schemat, w którym definiowana jest funkcjonalność obwodu. Na przykład, łącząc cztery przerzutniki, można utworzyć czterobitowy licznik. Wejścia i wyjścia widoku implementacji można następnie powiązać z wejściami i wyjściami widoku interfejsu. Widok interfejsu można uznać za obiekt lub "czarną skrzynkę" ukrywającą implementację obwodów. Widok interfejsu projektuje się w edytorze szablonu (*template editor*). Relację między tymi dwoma widokami pokazano poniżej [1].

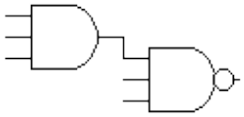


Rysunek 3.55 Widok implementacji [1]



Rysunek 3.56 Widok interfejsu [1]

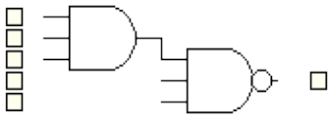
Widok implementacji jest tworzony przy użyciu takich samych technik co przy konstruowaniu zwykłego obwodu na obszarze roboczym. Tworzenie makra zostanie pokazane na przykładzie 5 – wejściowej bramki NAND. Początkowa implementacja tego układu jest pokazana poniżej (Rysunek 3.57) [1].



Rysunek 3.57 Implementacja bramki NAND [1]

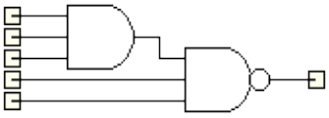
Należy umieścić odpowiednią liczbę makro tagów (Rysunek 3.58), które będą reprezentowały wejścia i wyjścia obwodu. W tym przykładzie ich liczba wynosi 6 (Rysunek 3.59) [1].

Rysunek 3.58 Makro Tag (*Tag Device*) [1]



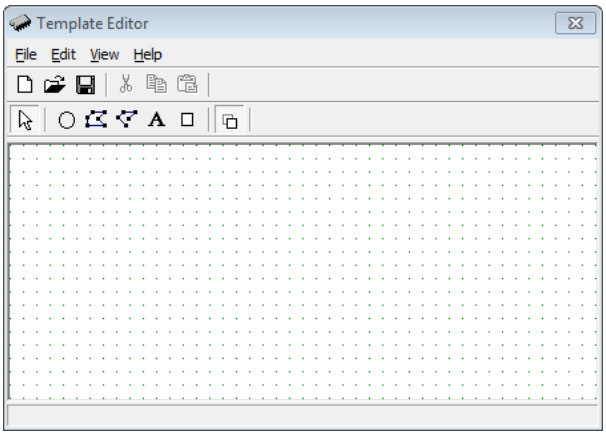
Rysunek 3.59 Rozmieszczenie makro tagów [1]

Należy połączyć wyprowadzenia bramek z tagami (Rysunek 3.60) [1].



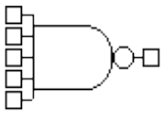
Rysunek 3.60 Połączenie wyprowadzeń z tagami [1]

Następnie, trzeba otworzyć edytor szablonu (Rysunek 3.61) w celu zaprojektowania wyglądu interfejsu używając dostępnych narzędzi lub wybrać gotowy szablon z folderu *Templates* z rozszerzeniem *dwt* [1].



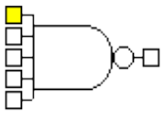
Rysunek 3.61 Okno *Template Editor*

Kolejnym etapem jest dodanie pinów w edytorze szablonów (Rysunek 3.62). Piny te będą łącznikami z tagami makra [1].



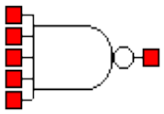
Rysunek 3.62 Widok interfejsu w edytorze szablonów [1]

Aby połączyć tag makra z pinem w edytorze szablonów należy kliknąć na nim prawym przyciskiem myszy, następnie wejść w edytor szablonów, kliknąć prawym przyciskiem myszy na pin i wybrać opcję *Associate with Tag*. Pin zmieni kolor na żółty (Rysunek 3.63), a tag makra otrzyma numer [1].



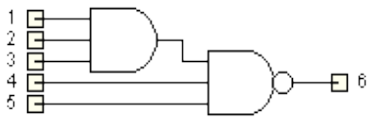
Rysunek 3.63 [1]

Należy powtórzyć kroki dla wszystkich pinów, aż wszystkie zostaną połączone. Finalny wygląd interfejsu wygląda następująco (Rysunek 3.64) [1]:



Rysunek 3.64 Finalny wygląd interfejsu 5 - wejściowej bramki NAND [1]

Natomiast finalny wygląd implementacji przykładowo (Rysunek 3.65) [1]:



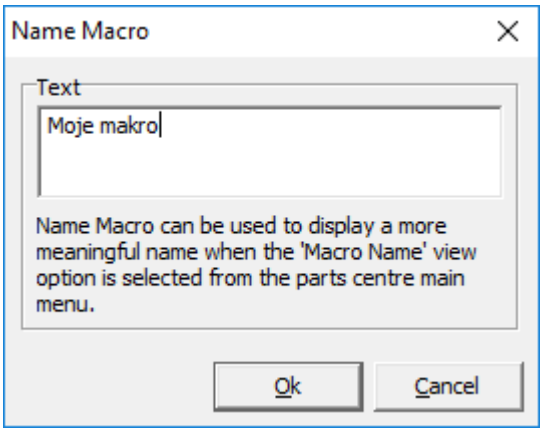
Rysunek 3.65 Finalny wygląd implementacji 5 – wejściowej bramki NAND [1]

Istnieje możliwość ukrycia pinów w edytorze szablonów poprzez kliknięcie w ikonę *Toggle Display Pins* (Rysunek 3.66), w takim przypadku na obszarze roboczym będą widoczne tylko wyprowadzenia komponentu [1].

Rysunek 3.66 Ikona *Toggle Display Pins* [1]

3.10.2. Nadanie nazwy makra

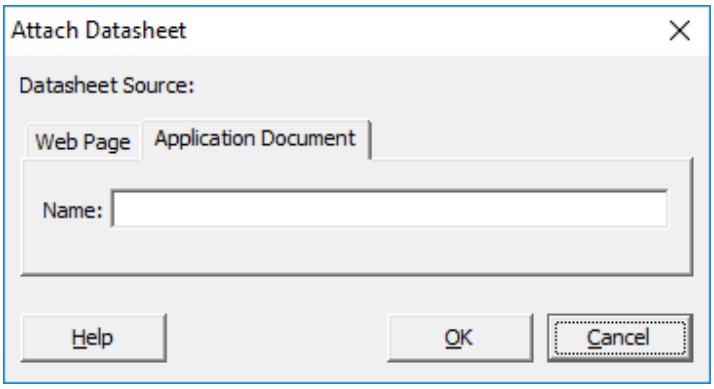
Digital Works może wyświetlać nazwę makra na dwa sposoby. Pierwszy z nich to wyświetlanie rzeczywistej nazwy pliku makra, natomiast drugim jest wyświetlanie nazwy stworzonej przez twórcę. Aby nazwać makro, należy wybrać zakładkę *Tools*, potem *Options*, następnie *Name Macro* (Rysunek 3.67) z menu głównego i wpisać nazwę. Można również nazwać makro w szybszy sposób, klikając prawym przyciskiem myszy obiekt adnotacji i wybierając opcję *Name Macro* [1].



Rysunek 3.67 Okno *Name Macro*

3.10.3. Dołączanie dokumentacji

Aby dodać dokumentację do makra, należy wybrać *Tools* z paska narzędzi głównych, następnie *Options* i *Attach Datasheet* (Rysunek 3.68). Wyświetli się okno dialogowe, w którym można podać nazwę pliku dokumentacji lub adres strony internetowej, na której jest umieszczona dokumentacja. Pliki dokumentacji należy zapisywać w popularnych formatach typu *pdf* lub *docx* w katalogu *Datasheets* programu Digital Works [1].



Rysunek 3.68 Okno *Attach Datasheet*

3.10.4. Dodawanie makra do centrum części

Aby dodać makro do centrum części, należy zapisać układ w folderze *Parts Centre*, który znajduje się w katalogu głównym programu Digital Works. Istnieje możliwość stworzenia własnego podfolderu dla lepszej organizacji plików [1].

3.11. Narzędzia symulacji

Run (Rysunek 3.69) – uruchamia symulację. Podczas symulacji możliwe jest używanie elementów interaktywnych poprzez narzędzia *Object Selector* i *Object Interaction Tool* [1].

Rysunek 3.69 Ikona *Run* [1]

Stop (Rysunek 3.70) – zatrzymuje symulację [1].

Rysunek 3.70 Ikona *Stop* [1]

Pause (Rysunek 3.71) – zawiesza tymczasowo symulację. Ponowne kliknięcie wznawia symulację bez resetowania [1].

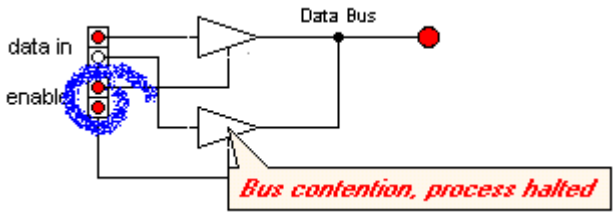
Rysunek 3.71 Ikona *Pause* [1]

Step (Rysunek 3.72) – umożliwia odtwarzanie symulacji krok po kroku tzn. po połowie lub pełnym cyklu zegarowym [1].

Rysunek 3.72 Ikona *Step* [1]

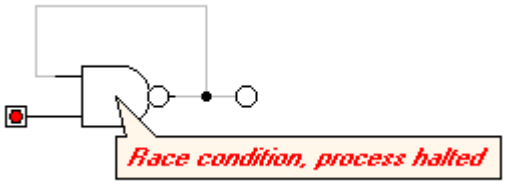
3.12. Hazard logiczny

Bus Contention – „spór magistralowy” (Rysunek 3.73) występuje, gdy dwie lub więcej bramek trójstanowych próbuje jednocześnie uzyskać dostęp do tej samej części magistrali [1].



Rysunek 3.73 Przykład sporu magistralowego [1]

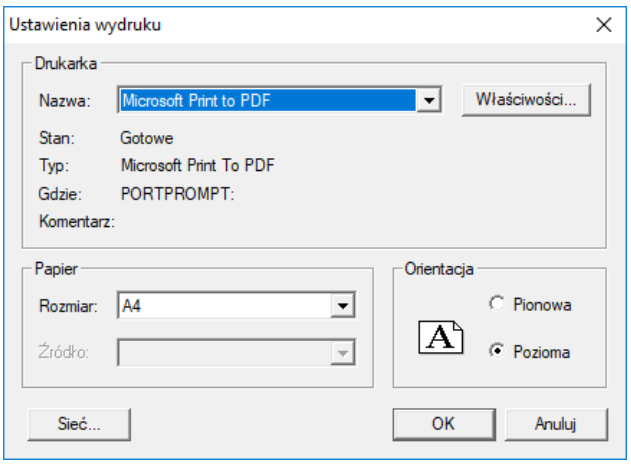
Race Condition – „wyścig logiczny” (Rysunek 3.74) występuje gdy, sygnał wyjściowy zależy tylko i wyłącznie od czasu propagacji bramek logicznych i nie jest stabilny [1].



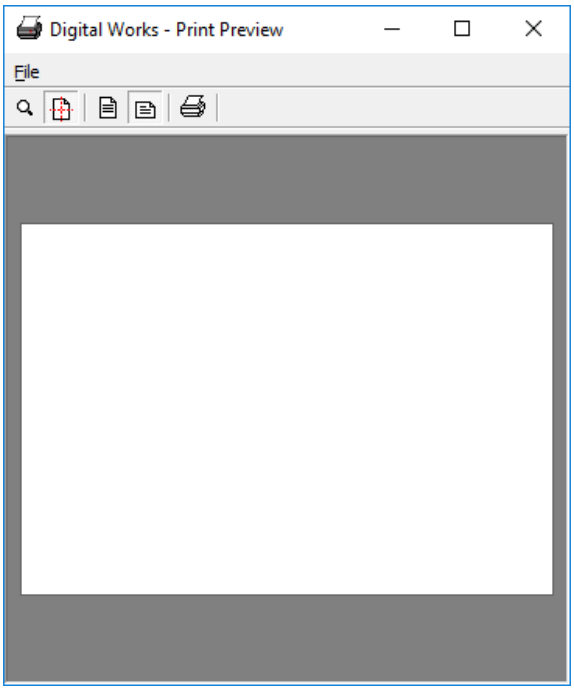
Rysunek 3.74 Przykład wyścigu logicznego [1]

3.13. Eksport schematu

Schemat układu można wydrukować lub wyeksportować do pliku wybierając w opcji *Print Setup* drukarkę *pdf* (Rysunek 3.75). Przed drukowaniem istnieje możliwość podglądu wydruku i ustawienia schematu względem papieru (Rysunek 3.76) [1].



Rysunek 3.75 Ustawienia wydruku

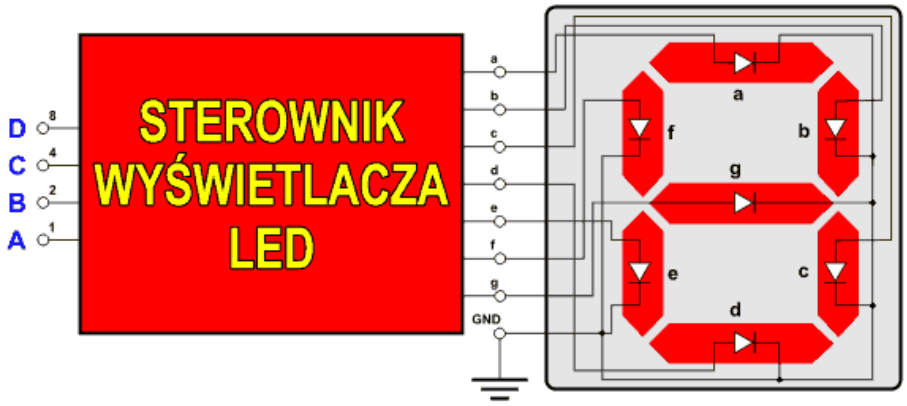


Rysunek 3.76 Podgląd wydruku (*Print Preview*)

4. MODELE WYBRANYCH UKŁADÓW I OCENA ICH DZIAŁANIA

4.1. Konwerter kodu BCD na kod 7 – segmentowy

Przedstawiany konwerter kodu BCD na kod 7 – segmentowy jest typowym układem kombinacyjnym, który zamienia kod binarny na kod stosowany w wyświetlaczach 7 – segmentowych LED. Dla wyświetlaczy 7 – segmentowych o wspólnej anodzie sygnałem aktywnym jest stan niski, natomiast dla wyświetlaczy 7 – segmentowych o wspólnej katodzie sygnałem aktywnym jest stan wysoki. Niniejszy układ został zaprojektowany dla drugiego typu wyświetlacza. Podłączenie wyświetlacza 7 – segmentowego do konwertera kodu BCD na 7 – segmentowy zostało pokazane na Rysunek 4.1. Sygnały wyjściowe układu w funkcji sygnałów wejściowych zostały pokazane w Tabela 4.1 [7].



Rysunek 4.1 Podłączenie wyświetlacza do układu [7]

Początkowym etapem projektowania układu jest określenie aktywnych segmentów odpowiadających cyfrze zapisanej w systemie binarnym (Tabela 4.1). Następnie tworzy się tablicę Karnaugha (Tabela 4.2 – 4.5) i minimalizuje funkcje logiczne dla każdego segmentu wyświetlacza (Tabela 4.6). Otrzymane wyrażenie logiczne opisujące dany segment to kanoniczna postać sumy, przekształcona w taki sposób, aby można ją było zrealizować wyłącznie na bramkach NAND. Przykładowa procedura minimalizacji funkcji logicznej została pokazana dla segmentu „a", ponieważ pozostałe wyrażenia określa się w analogiczny sposób. Składniki kanonicznej postaci sumy zostały oznaczone nazwą segmentu z indeksem dolnym (np. a_1) [7].

Tabela 4.1 Działanie układu w zależności od sygnałów wejściowych [7]

Wejście				Wyjście							Cyfra
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	
0	0	0	1	0	1	1	0	0	0	0	
0	0	1	0	1	1	0	1	1	0	1	
0	0	1	1	1	1	1	1	0	0	1	
0	1	0	0	0	1	1	0	0	1	1	
0	1	0	1	1	0	1	1	0	1	1	
0	1	1	0	1	0	1	1	1	1	1	
0	1	1	1	1	1	1	0	0	1	0	
1	0	0	0	1	1	1	1	1	1	1	
1	0	0	1	1	1	1	1	0	1	1	

Tabela 4.2 Tablica Karnaugha dla składnika a_1

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Tabela 4.3 Tablica Karnaugha dla składnika a_2

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Tabela 4.4 Tablica Karnaugha dla składnika a_3

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Tabela 4.5 Tablica Karnaugha dla składnika a_4

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Tabela 4.6 Minimalizacja składników segmentu „a”

a_1			
D	C	B	A
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	0	0
1	0	0	1
1	0	1	1
1	0	1	0

a_2			
D	C	B	A
0	0	1	1
0	0	1	0
0	1	1	1
0	1	1	0
1	1	1	1
1	1	1	0
1	0	1	1
1	0	1	0

a_3			
D	C	B	A
0	1	0	1
0	1	1	1
1	1	0	1
1	1	1	1

a_4			
D	C	B	A
0	0	0	0
0	0	1	0
1	0	0	0
1	0	1	0

Kanoniczna postać sumy dla segmentu „a” wygląda w następujący w sposób:

$$a = a_1 + a_2 + a_3 + a_4 = D + B + AC + \bar{A}\bar{C}. \tag{4.1}$$

Przekształcenie na wyrażenie złożone wyłącznie z funkcji NAND wykonuje się za pomocą II prawa De Morgana:

$$a = \overline{\overline{D + B + AC + \bar{A}\bar{C}}} = \overline{\overline{AC} \cdot \overline{\bar{A}\bar{C}} \cdot \bar{B} \cdot \bar{D}}. \tag{4.2}$$

Po zminimalizowaniu pozostałych funkcji logicznych stosując tylko bramki NAND otrzymuje się następujące wyrażenia logiczne dla poszczególnych segmentów [7]:

$$b = \overline{\overline{A\bar{B}} \cdot \overline{\bar{A}\bar{B}} \cdot C} \tag{4.3}$$

$$c = \overline{\bar{A} \cdot B \cdot \bar{C}} \tag{4.4}$$

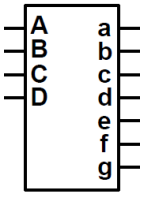
$$d = \overline{\overline{A\bar{B}} \cdot \overline{A\bar{B}C} \cdot \overline{\overline{A\bar{B}} \cdot \bar{C} \cdot \bar{D}}} \tag{4.5}$$

$$e = \overline{\overline{\bar{A} \cdot \bar{B}C}} \tag{4.6}$$

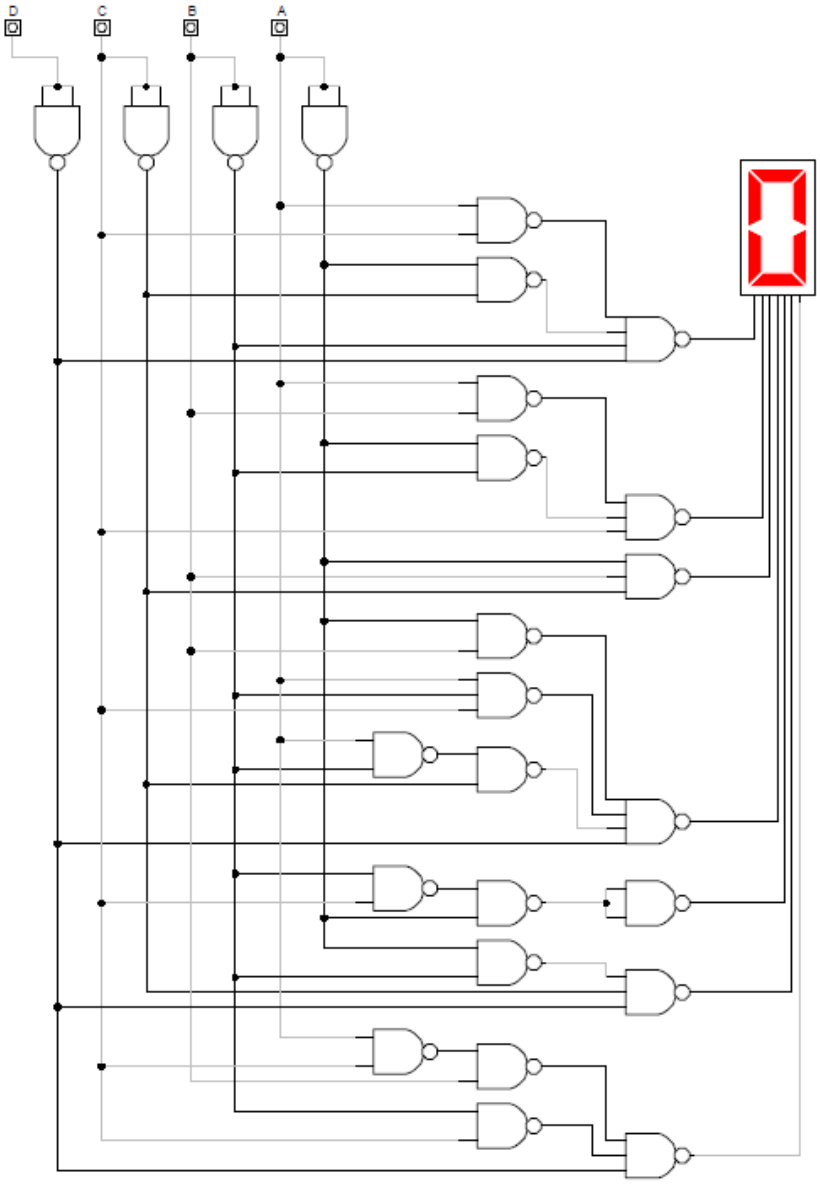
$$f = \overline{\overline{\bar{A}\bar{B}} \cdot \bar{C} \cdot \bar{D}} \tag{4.7}$$

$$g = \overline{\overline{\overline{AC}B} \cdot \overline{\bar{B}C} \cdot \bar{D}}. \tag{4.8}$$

Symbol konwertera kodu BCD na kod 7 - segmentowy został pokazany poniżej (Rysunek 4.2), natomiast jego sieć logiczna została pokazana na Rysunek 4.3.

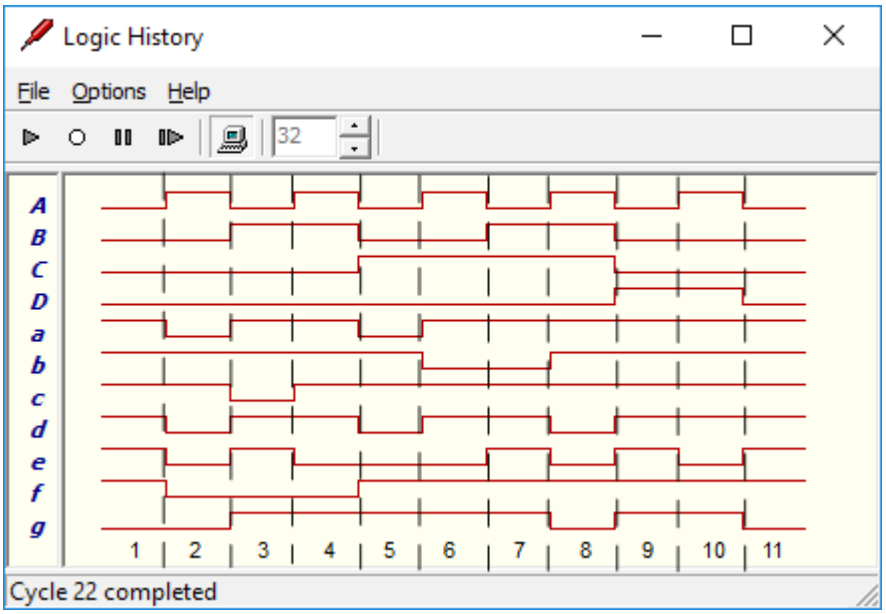


Rysunek 4.2 Symbol konwertera kodu BCD na kod 7 - segmentowy w programie Digital Works



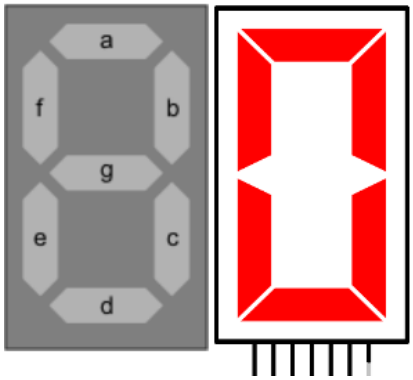
Rysunek 4.3 Sieć logiczna konwertera kodu BCD na 7 – segmentowy w programie Digital Works

Działanie układu w programie Digital Works można zaprezentować za pomocą narzędzia *Logic History* (Rysunek 4.4). Przebieg został podzielony na 11 sekcji dla ułatwienia opisu sygnałów. Dużymi literami oznaczono wejścia układu, które reprezentują cyfrę w postaci binarnej, a małymi literami oznaczono wyjścia układu – segmenty wyświetlacza.



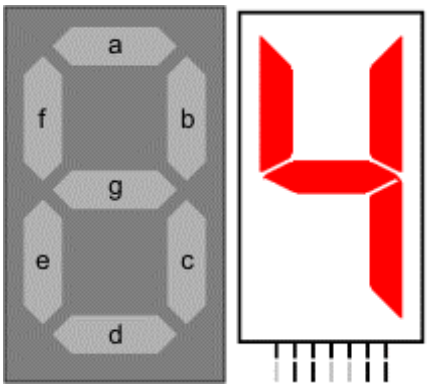
Rysunek 4.4 Przykładowy przebieg sygnałów konwertera kodu BCD na kod 7 – segmentowy

W sekcji pierwszej wszystkie wejścia układu są w stanie niskim więc w systemie binarnym mają postać 0000 co daje oczywiście wartość 0 w systemie dziesiętnym. Na wyświetlaczu pojawia się cyfra 0, ponieważ na wyjściu układu tylko segment „g" jest nieaktywny (Rysunek 4.5).



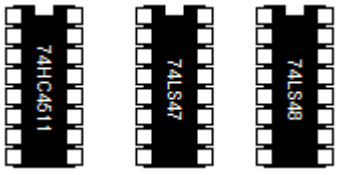
Rysunek 4.5

Pozostałe sekcje można opisać w analogiczny sposób. Przykładowo, w sekcji piątej na wejściu układu występuje cyfra w postaci binarnej 0100, która w systemie dziesiętnym jest równa 4. Na wyjściu układu występuje sygnał w postaci binarnej 1100110 więc aktywne są następujące segmenty: b, c, f, g (Rysunek 4.6).



Rysunek 4.6

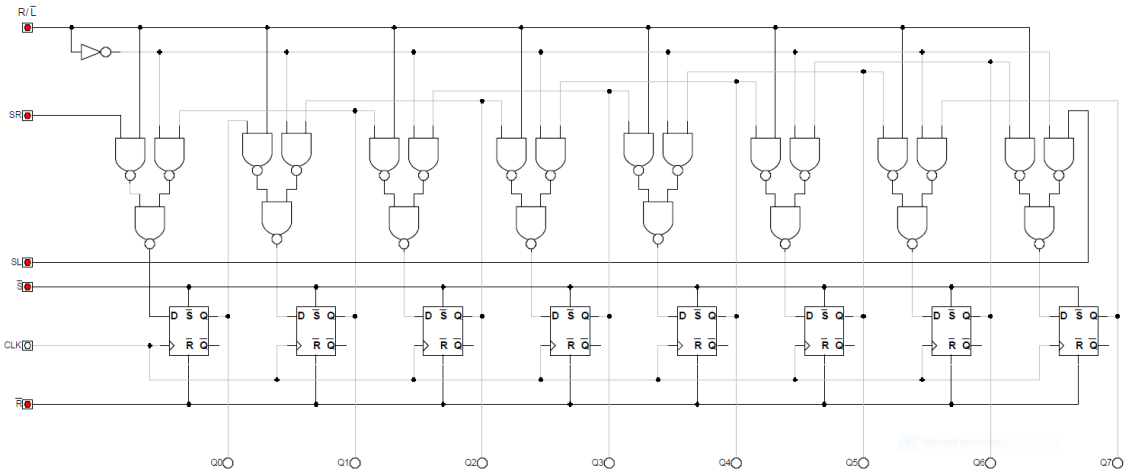
W programie Digital Works znajdują się gotowe sterowniki wyświetlaczy 7 – segmentowych w postaci układów scalonych, które dodatkowo posiadają dokumentację techniczną w załączniku *pdf* (Rysunek 4.7). Jednym z nich jest układ 74HC4511 firmy Philips przeznaczony do wyświetlaczy o wspólnej katodzie, posiadający dodatkowo wejścia $\overline{LE}$, $\overline{BI}$ i $\overline{LT}$. Gdy na wejściu $\overline{LE}$ (*latch enable*) występuje stan niski, stan wyjść układu jest natychmiast określany poprzez dane wejściowe (A, B, C, D). Natomiast gdy wejście $\overline{LE}$ jest w stanie wysokim, układ zapamiętuje w zatrzaskach dane z wejść, a stan na wyjściach jest stabilny – nie reaguje na zmiany danych wejściowych. Gdy wejście $\overline{LT}$ (*lamp test*) jest w stanie niskim, wszystkie wyjścia układu przechodzą w stan wysoki niezależnie od danych wejściowych (testowanie). Gdy wejście $\overline{LT}$ jest w stanie wysokim, a wejście $\overline{BI}$ (*blanking input*) w stanie niskim, wszystkie wyjścia przechodzą w stan niski (wygaszanie). Wejścia $\overline{BI}$ i $\overline{LT}$ nie wpływają na zatrzaski w układzie. Układ 74LS47 firmy Motorola jest przeznaczony do wyświetlaczy o wspólnej anodzie oraz posiada wejścia $\overline{LT}$, $\overline{BI}/\overline{RBO}$ i $\overline{RBI}$. Bliźniaczym do niego układem jest 74LS48, który jest przeznaczony do wyświetlaczy o wspólnej katodzie. Wejścia $\overline{BI}/\overline{RBO}$ i $\overline{RBI}$ służą kolejno do sterowania intensywnością świecenia (niedostępne w programie Digital Works) lub wygaszania zera oraz wygaszania początkowych zer w układach sterujących większą liczbą wyświetlaczy.



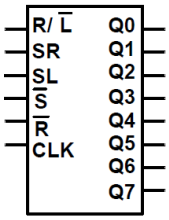
Rysunek 4.7 Sterowniki wyświetlaczy 7- segmentowych w programie Digital Works

4.2. Rejestr przesuwający dwukierunkowy

Opisywany rejestr przesuwający dwukierunkowy (Rysunek 4.8 oraz Rysunek 4.9) jest układem sekwencyjnym zbudowanym z ośmiu przerzutników typu D oraz bramek NAND. Jego działanie polega na przesuwaniu bitów podanych na wejście danych w takt narastających zboczy sygnału zegarowego, a także na zmianie kierunku ich przesuwania. Rejestr posiada dodatkowe wejścia asynchroniczne ustawiające i resetujące. Rejestry przesuwające dwukierunkowe są stosowane m.in. w układach sterowania do generowania określonej sekwencji stanów [8].



Rysunek 4.8 Sieć logiczna rejestru przesuwającego dwukierunkowego w programie Digital Works [7]



Rysunek 4.9 Symbol rejestru przesuwającego dwukierunkowego w programie Digital Works

Projektowanie rejestru przesuwającego tylko w jedną stronę za pomocą przerzutników typu D polega na połączeniu wejść D przerzutników z wyjściem Q poprzednich przerzutników. Wówczas wejście D pierwszego przerzutnika jest wejściem danych szeregowych. W celu dodania możliwości przesuwania w dwie strony, należy dołączyć odpowiedni układ z bramek NAND, który steruje wejściami przerzutników. Aby uzyskać przesuw w odwrotnym kierunku, do wejść D należy podłączyć wyjścia Q następnych, a nie poprzednich przerzutników poprzez układ bramek NAND [8]. Wejście $R/\overline{L}$ steruje kierunkiem przesuwu bitów. Stan wysoki na tym wejściu powoduje przesuwanie bitów w prawą stronę, a stan niski przesuwanie w lewą stronę. Wejścia SR i SL to odpowiednio szeregowe wejścia danych dla przesuwu w prawo i w lewo.

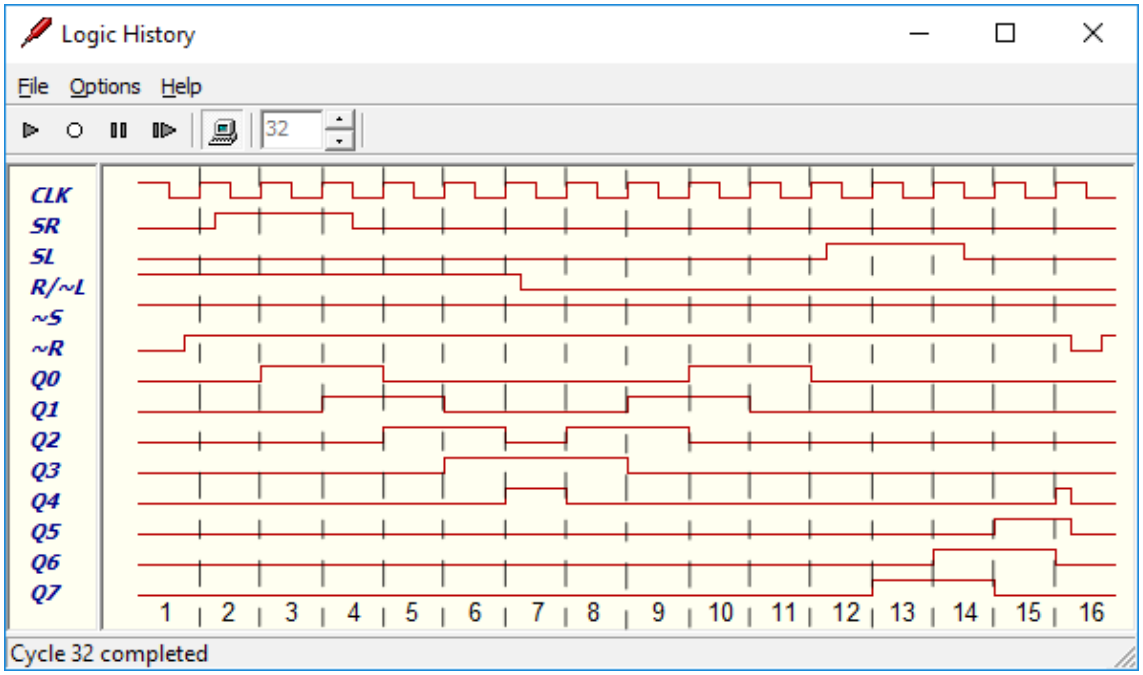
Wcześniej wspomniane wejścia asynchroniczne reagujące na poziom niski to ustawianie $\bar{S}$ oraz resetowanie $\bar{R}$. Wejście CLK to sygnał zegarowy. Działanie powyższego układu może być opisane za pomocą przebiegu (Rysunek 4.10), tabeli (Tabela 4.7) lub równań (4.9) przy czym zapis „(+)” oznacza tutaj stan wyjścia po przełączeniu narastającym zboczem zegarowym, a zapis „(-)” oznacza stan wyjścia przed przełączeniem narastającym zboczem zegarowym:

$$\begin{cases} Q0_{(+)} = SR \cdot R/\bar{L} + \overline{R/\bar{L}} \cdot Q1_{(-)} \\ Qi_{(+)} = Q(i-1)_{(-)} \cdot R/\bar{L} + \overline{R/\bar{L}} \cdot Q(i+1)_{(-)} \\ Q(n-1)_{(+)} = Q(n-2)_{(-)} \cdot R/\bar{L} + \overline{R/\bar{L}} \cdot SL \end{cases} \quad \begin{cases} n \in \mathbb{N}_+ \backslash \{0,1\} \\ i \in \mathbb{N}_+ \backslash \{0\} \\ i < n-1 \end{cases} \tag{4.9}$$

Gdzie:

n – liczba przerzutników w układzie

i – indeks przerzutnika



Rysunek 4.10 Przykładowy przebieg sygnałów rejestru przesuwającego dwukierunkowego

Tabela 4.7 Działanie 8 – bitowego rejestru przesuwającego dwukierunkowego

Sekcja	SR	SL	$R/\bar{L}$	$\bar{S}$	$\bar{R}$	$Q_7\ Q_6\ Q_5\ Q_4\ Q_3\ Q_2\ Q_1\ Q_0$
1	0	0	1	1	0→1	00000000
2	0→1	0	1	1	1	00000000
3	1	0	1	1	1	10000000
4	1→0	0	1	1	1	11000000
5	0	0	1	1	1	01100000
6	0	0	1	1	1	00110000
7	0	0	1→0	1	1	00011000
8	0	0	0	1	1	00110000
9	0	0	0	1	1	01100000
10	0	0	0	1	1	11000000
11	0	0	0	1	1	10000000
12	0	0→1	0	1	1	00000000
13	0	1	0	1	1	00000001
14	0	1→0	0	1	1	00000011
15	0	0	0	1	1	00000110
16	0	0	0	1	1→0→1	00001100→00000000

W sekcji pierwszej w stanie wysokim są tylko wejścia $R/\bar{L}$ (przesuw w prawo) i $\bar{S}$, natomiast wejście asynchroniczne $\bar{R}$ początkowo jest w stanie niskim w celu wyzerowania układu. Jeszcze przed drugim zboczem narastającym na wejściu $\bar{R}$ pojawia się stan wysoki więc minął stan początkowy. W sekcji drugiej pojawił się stan wysoki na wejściu SR więc przy następnym narastającym zboczu zegarowym na wyjściu $Q0$ pojawi się pierwszy bit. W sekcji trzeciej na wyjściu pojawia się bit więc wyjścia mają postać 10000000. W sekcji czwartej wejścia SR i $R/\bar{L}$ są jeszcze w stanie wysokim przy narastającym zboczu zegarowym więc następuje generowanie i przesuwanie starszego bitu w prawo. Wyjścia mają postać 11000000. W sekcji piątej na wejściu SR występuje stan niski więc układ nie generuje bitu, natomiast na wejściu $R/\bar{L}$ jest cały czas stan wysoki więc układ przesuwa bity o jedną pozycję w prawo. Wyjścia mają postać 01100000. W sekcji szóstej sytuacja jest taka sama jak w sekcji piątej więc układ przesuwa bity w prawo. Wyjścia mają postać 00110000. W sekcji siódmej wejście $R/\bar{L}$ jest jeszcze w stanie wysokim przy narastającym zboczu sygnału zegarowego więc następuje przesunięcie bitu w prawo. Wyjścia mają postać 00011000. W sekcji ósmej wejście $R/\bar{L}$ jest w stanie niskim więc bity zaczynają być przesuwane w lewo bez ich generowania, ponieważ wejście SL nadal jest w stanie niskim. Wyjścia mają postać 00110000. W sekcji dziewiątej występuje ta sama sytuacja co w sekcji ósmej więc

następuje przesunięcie bitów w lewo. Wyjścia mają postać 01100000. W sekcji dziesiątej występuje ta sama sytuacja co w sekcji dziewiątej więc następuje przesunięcie bitów w lewo. Wyjścia mają postać 11000000. W sekcji jedenastej występuje ta sama sytuacja co w sekcji dziesiątej więc następuje przesunięcie bitów w lewo. Wyjścia mają postać 10000000. W sekcji dwunastej wejście SL przechodzi w stan wysoki, a wszystkie wyjścia są wyzerowane. W sekcji trzynastej przy narastającym zboczu sygnału zegarowego wejście SL jest w stanie wysokim więc następuje generowanie bitu z prawej strony. Wyjścia mają postać 00000001. W sekcji czternastej wejście SL jest aktywne przy narastającym zboczu sygnału zegarowego więc następuje generowanie jeszcze jednego bitu z prawej strony i przesuw w lewo. Wyjścia mają postać 00000011. W sekcji piętnastej wejście SL jest w stanie niskim więc występuje tylko przesuw bitów w lewo. Wyjścia mają postać 00000110. W sekcji szesnastej występuje taka sama sytuacja co w sekcji czternastej lecz tylko na chwilę, ponieważ na wejściu asynchronicznym $\bar{R}$ pojawia się stan niski co oznacza reset układu. Wyjścia mają kolejno postać 00001100→00000000.

W programie Digital Works znajduje się gotowy uniwersalny 4 – bitowy rejestr przesuwający dwukierunkowy w postaci układu scalonego o nazwie 74HC194 (Rysunek 4.11). Dokumentacja nie została do niego dołączona, ale można ją znaleźć w Internecie. Układ posiada dwa wejścia $S0$ i $S1$ ustalające tryb pracy:

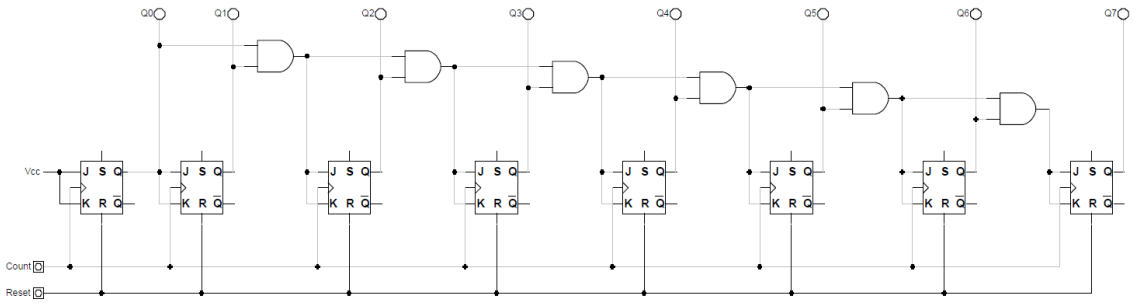
- $S0:S1$ = 0:0 Blokada zegara – nic nie jest wykonywane.
- $S0:S1$ = 1:0 Przesuw w prawo: w kierunku od QA do QD.
- $S0:S1$ = 0:1 Przesuw w lewo: w kierunku od QD do QA.
- $S0:S1$ = 1:1 Wpis równoległy.

Pozostałe wejścia to: wejście zegarowe reagujące na zbocze narastające, asynchroniczne wejście zerowania $\overline{CLR}$, wejście danych szeregowych przy przesuwie w prawo SR, wejście danych szeregowych przy przesuwie w lewo SL, cztery wejścia danych równoległych oraz cztery wyjścia [7].

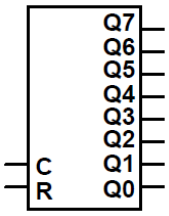
Rysunek 4.11 Rejestr uniwersalny w programie Digital Works

4.3. Licznik prosty

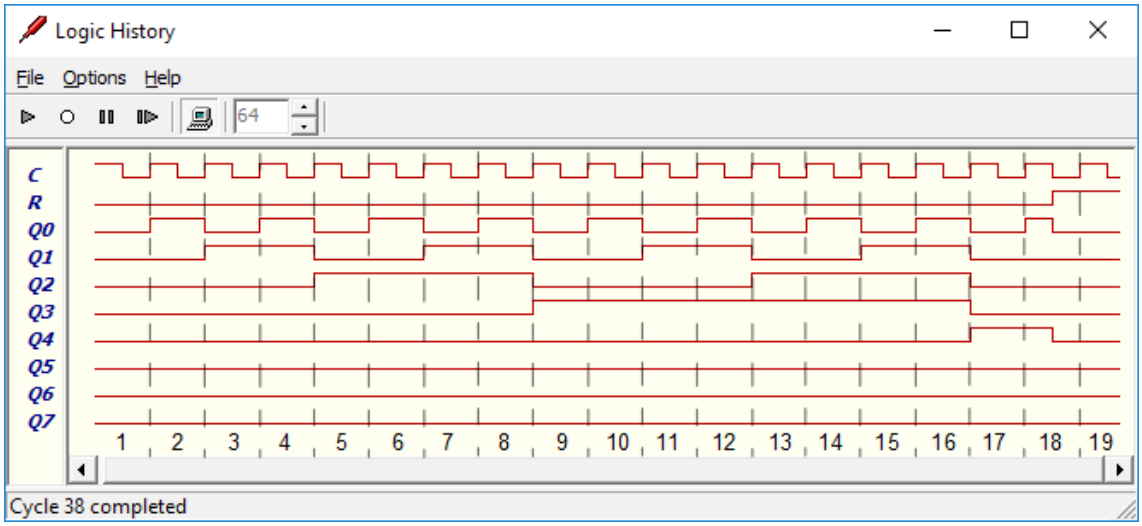
Prezentowany licznik prosty (Rysunek 4.12 oraz Rysunek 4.13) jest 8 – bitowym synchronicznym układem sekwencyjnym zliczającym dodatnie impulsy sygnału zegarowego. Został zbudowany z ośmiu przerzutników JK i sześciu bramek AND, które sterują wejściami J i K przerzutników. Licznik posiada asynchroniczne wejście resetujące *R* reagujące na stan wysoki. Działanie tego układu najlepiej zaprezentować na przebiegu (Rysunek 4.14) i tabeli (Tabela 4.8).



Rysunek 4.12 Schemat 8 – bitowego licznika synchronicznego w programie Digital Works [8]



Rysunek 4.13 Symbol licznika 8 – bitowego synchronicznego w programie Digital Works



Rysunek 4.14 Przebieg sygnałów 8 – bitowego licznika synchronicznego

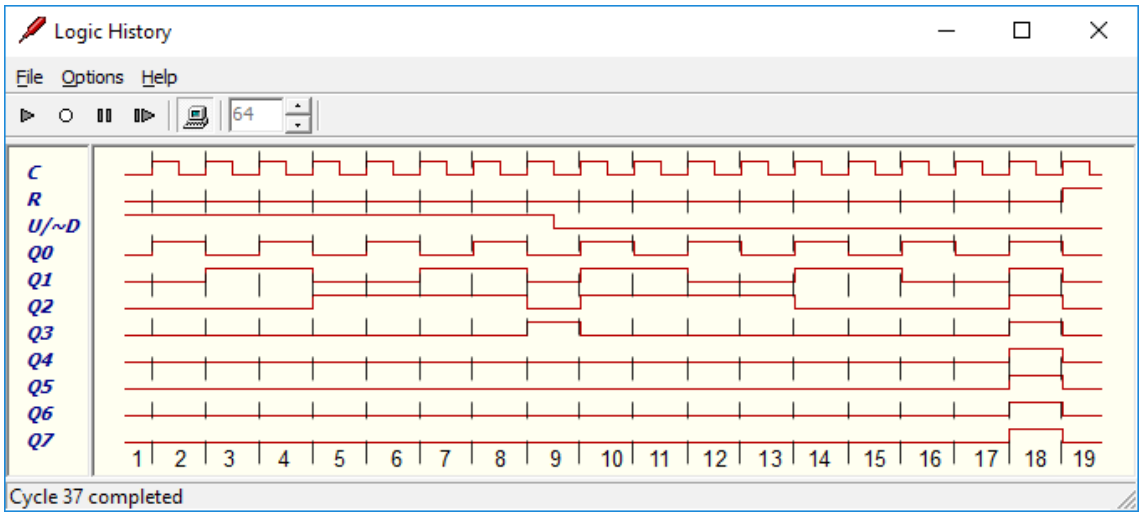
Tabela 4.8 Wartości binarne i dziesiętne 8 – bitowego licznika synchronicznego

Sekcja	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	Dziesiętnie
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	1	0	2
4	0	0	0	0	0	0	1	1	3
5	0	0	0	0	0	1	0	0	4
6	0	0	0	0	0	1	0	1	5
7	0	0	0	0	0	1	1	0	6
8	0	0	0	0	0	1	1	1	7
9	0	0	0	0	1	0	0	0	8
10	0	0	0	0	1	0	0	1	9
11	0	0	0	0	1	0	1	0	10
12	0	0	0	0	1	0	1	1	11
13	0	0	0	0	1	1	0	0	12
14	0	0	0	0	1	1	0	1	13
15	0	0	0	0	1	1	1	0	14
16	0	0	0	0	1	1	1	1	15
17	0	0	0	1	0	0	0	0	16
18	0	0	0	1	0	0	0	1	17
19	0	0	0	0	0	0	0	0	0

Jak widać na powyższej tabeli, układ prawidłowo zlicza dodatnie impulsy zegarowe. W sekcji osiemnastej licznik został zresetowany poprzez podanie na wejście R stanu wysokiego.

4.4. Licznik rewersyjny

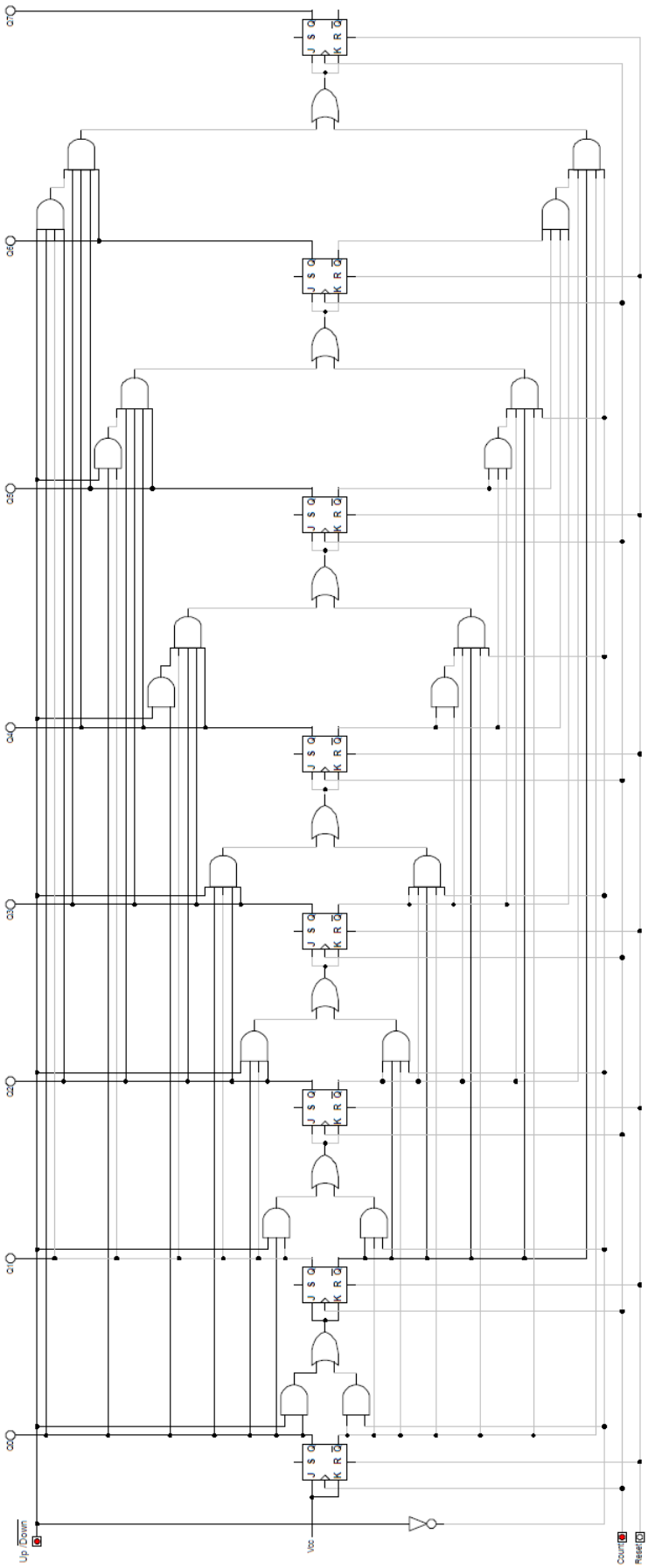
Prezentowany licznik rewersyjny synchroniczny 8 – bitowy (Rysunek 4.16 oraz Rysunek 4.17) jest układem sekwencyjnym, który oprócz zliczania impulsów w przód, potrafi zliczać także w tył poprzez odpowiedni sygnał na wejściu $U/\overline{D}$ ($Up/\overline{Down}$). Został zbudowany z ośmiu przerzutników JK oraz bramek AND, OR i jednej pomocniczej NOT. Stan wysoki na wejściu $U/\overline{D}$ powoduje zliczanie dodatnich impulsów zegarowych w przód, natomiast stan niski na tym wejściu powoduje zliczanie tych impulsów w tył (odejmowanie). Działanie tego układu najlepiej zaprezentować na przebiegu (Rysunek 4.15) i tabeli (Tabela 4.9).



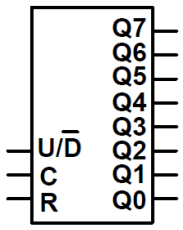
Rysunek 4.15 Przebieg sygnałów 8 – bitowego licznika synchronicznego rewersyjnego

Tabela 4.9 Wartości binarne i dziesiętne 8 – bitowego licznika synchronicznego rewersyjnego

Sekcja	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	$U/\overline{D}$	Dziesiętnie
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	1	1
3	0	0	0	0	0	0	1	0	1	2
4	0	0	0	0	0	0	1	1	1	3
5	0	0	0	0	0	1	0	0	1	4
6	0	0	0	0	0	1	0	1	1	5
7	0	0	0	0	0	1	1	0	1	6
8	0	0	0	0	0	1	1	1	1	7
9	0	0	0	0	1	0	0	0	1→0	8
10	0	0	0	0	0	1	1	1	0	7
11	0	0	0	0	0	1	1	0	0	6
12	0	0	0	0	0	1	0	1	0	5
13	0	0	0	0	0	1	0	0	0	4
14	0	0	0	0	0	0	1	1	0	3
15	0	0	0	0	0	0	1	0	0	2
16	0	0	0	0	0	0	0	1	0	1
17	0	0	0	0	0	0	0	0	0	0
18	1	1	1	1	1	1	1	1	0	255



Rysunek 4.16 Schemat 8 – bitowego synchronicznego licznika rewersyjnego [8]

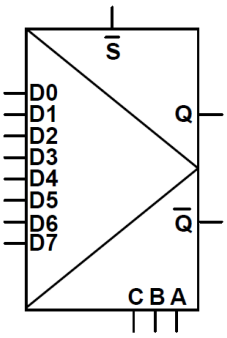


Rysunek 4.17 Symbol 8 – bitowego licznika rewersyjnego w programie Digital Works

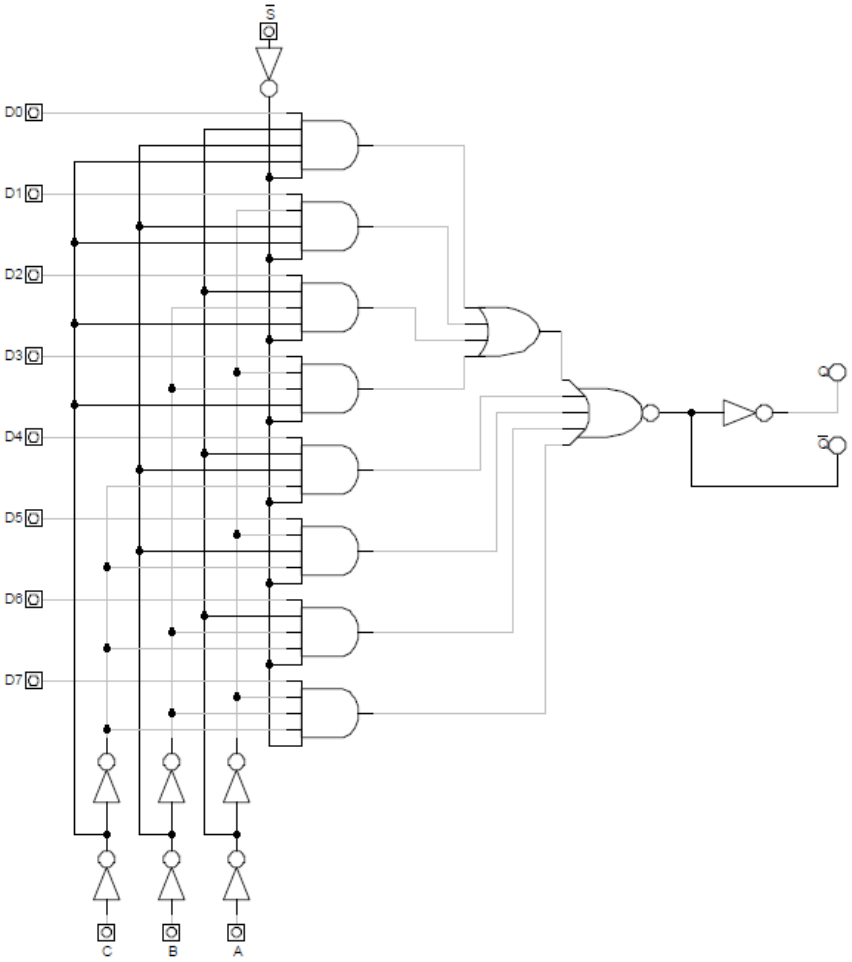
Układ działa zgodnie z oczekiwaniami. Prawidłowo zlicza impulsy sygnału zegarowego zarówno w przód jak i w tył. Gdy wszystkie wyjścia są w stanie niskim oraz wejście $U/\bar{D}$ jest również w stanie niskim, przy dodatnim zboczu narastającym sygnału zegarowego na wyjściu pojawi się stan 11111111, który w systemie dziesiętnym daje liczbę 255, ponieważ pojemność tego licznika jest równa $2^8 - 1$.

4.5. Multiplekser – generator

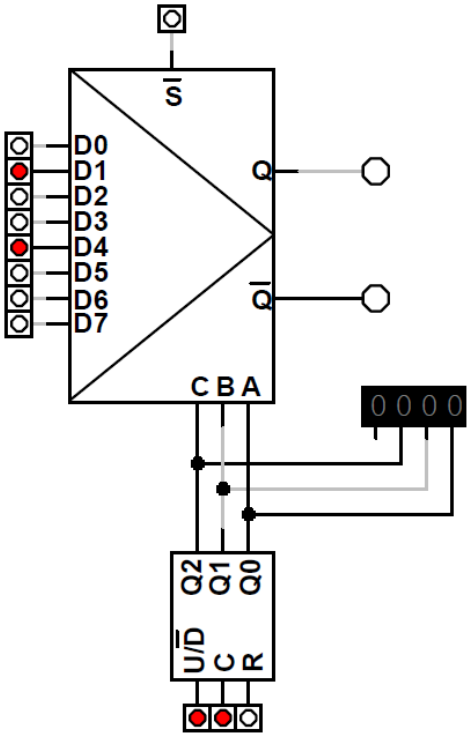
Układ składa się z 8 - bitowego multipleksera i 3 – bitowego licznika rewersyjnego, którego wyjścia zostały połączone z wejściami adresowymi multipleksera. Licznik rewersyjny został przedstawiony w podrozdziale 4.4. Multiplekser jest układem kombinacyjnym, który umożliwia przeniesienie sygnału z jednego wejścia danych na wyjście, poprzez podanie adresu tego wejścia danych na wejścia adresowe (C, B, A). Nazywany także selektorem danych lub wielopozycyjnym cyfrowym przełącznikiem. Multiplekser posiada wejście strobujące (blokujące) $\bar{S}$, które w przypadku stanu wysokiego uniemożliwia działanie układu i wyjście komplementarne $\bar{Q}$. Multiplekser można zaprojektować korzystając z tablic Karnaugha lecz taki sposób projektowania ma sens w przypadku multipleksera 4 – bitowego, ponieważ istnieje tylko 2^4 kombinacji sygnałów wejściowych i tablica Karnaugha nie byłaby trudna do rozwiązania. Natomiast w przypadku 8 – bitowego multipleksera liczba kombinacji wzrasta do 2^8 przy trzech wejściach adresowych co znacznie komplikuje projektowanie. Symbol 8 – bitowego multipleksera został przedstawiony na Rysunek 4.18, a jego sieć wewnętrzna na Rysunek 4.19. Przedstawiany multiplekser został zbudowany z ośmiu bramek NOT, ośmiu bramek AND, jednej bramki OR oraz NOR, które mogłyby być zastąpione jedną ośmiowejściową bramką NOR. Układ multipleksera – generatora został pokazany na rysunkuRysunek 4.20.



Rysunek 4.18 Symbol 8 – bitowego multipleksera w programie Digital Works

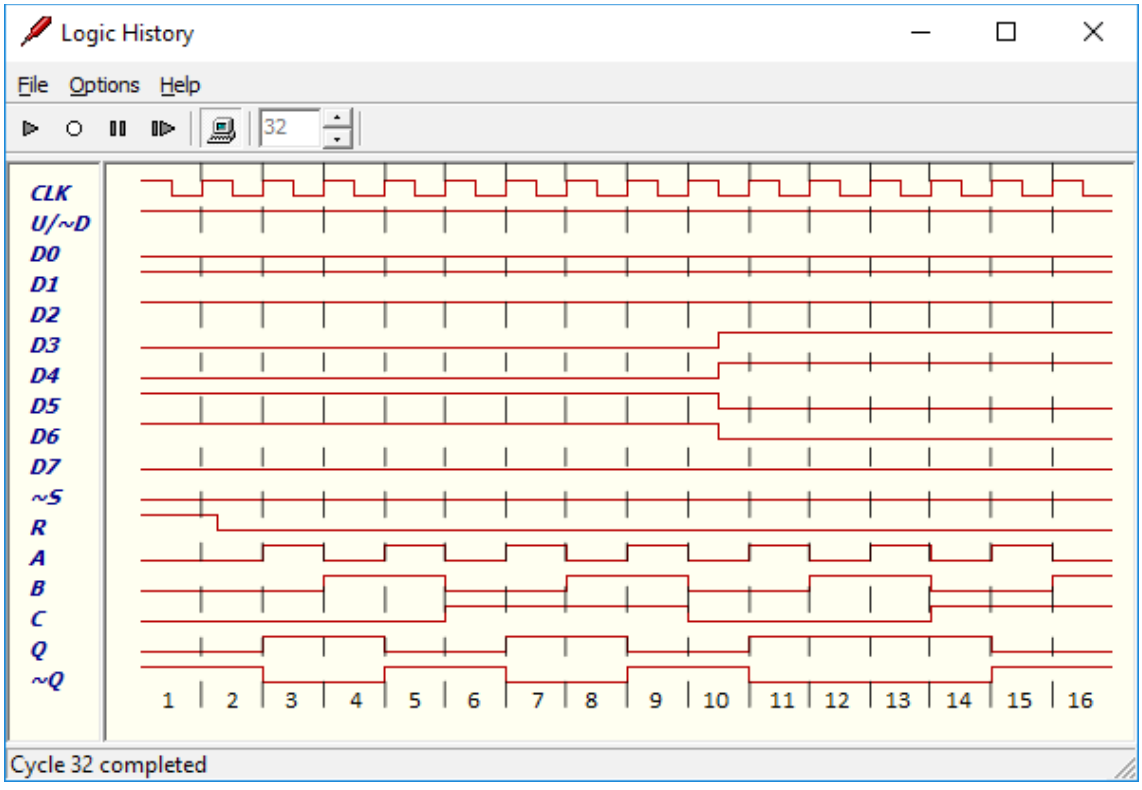


Rysunek 4.19 Sieć logiczna 8 – bitowego multipleksera [4]



Rysunek 4.20 8 – bitowy multiplekser - generator

Działanie multipleksera – generatora zostało pokazane na poniższym przebiegu (Rysunek 4.21) i tabeli 4.10.



Rysunek 4.21 Przykładowy przebieg 8 – bitowego multipleksera – generatora

Tabela 4.10 Działanie 8 – bitowego multipleksera - generatora

Sekcja	$U/\overline{D}$	$\bar{S}$	R	$C\ B\ A$	$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	Q
1	1	0	1	000	01100110	0
2	1	0	1→0	000	01100110	0
3	1	0	0	001	01100110	1
4	1	0	0	010	01100110	1
5	1	0	0	011	01100110	0
6	1	0	0	100	01100110	0
7	1	0	0	101	01100110	1
8	1	0	0	110	01100110	1
9	1	0	0	111	01100110	0
10	1	0	0	000	01100110→00011110	0
11	1	0	0	001	01111000	1
12	1	0	0	010	01111000	1
13	1	0	0	011	01111000	1
14	1	0	0	100	01111000	1
15	1	0	0	101	01111000	0
16	1	0	0	110	01111000	0

Jak wspomniano wcześniej, do wejść adresowych multipleksera dołączono 3 – bitowy licznik rewersyjny, ponieważ jego pojemność wynosi 7, co odpowiada liczbie wejść multipleksera (licząc od 0). Sygnał zegarowy został dołączony do wejścia C licznika rewersyjnego, który synchronizuje cały układ. Adresy wejść multipleksera są wybierane sekwencyjnie od 0 do 7 w postaci kodu BCD lub od 7 do 0 w przypadku stanu niskiego na wejściu $U/\overline{D}$. Na wejścia multipleksera należy podać określone stany, aby uzyskać żądany przebieg na wyjściu. W powyższym przykładzie wejście $U/\overline{D}$ cały czas jest w stanie wysokim więc licznik zlicza sygnały zegarowe w przód. W sekcji pierwszej na wejściach danych multipleksera od D0 do D7 występuje stan 01100110, który zmieni się dopiero w sekcji dziesiątej. Na wejściu adresowym jest stan 000 co oczywiście odpowiada adresowi D0. Wyjście multipleksera Q jest w stanie niskim, ponieważ na wejściu D0 panuje taki stan. W sekcji drugiej wejście adresowe ma nadal postać 000 więc stan na wyjściu multipleksera nie zmienia się. W sekcji trzeciej wejście adresowe ma postać 001 co odpowiada wejściu D1 multipleksera, które jest w stanie wysokim więc stan na wyjściu zmienia się na wysoki. W sekcji czwartej wejście adresowe ma postać 010 co odpowiada wejściu D2, które jest w stanie wysokim więc stan na wyjściu nie zmienia się. W sekcji piątej wejście adresowe ma postać 011 co odpowiada wejściu D3, które jest w stanie niskim więc wyjście multipleksera przechodzi w stan niski. W sekcji

szóstej wejście adresowe ma postać 100 co odpowiada wejściu D4, które jest w stanie niskim więc stan na wyjściu multipleksera nie zmienia się. W sekcji siódmej wejście adresowe ma postać 101 co odpowiada wejściu D5, które jest w stanie wysokim więc stan na wyjściu zmienia się na wysoki. W sekcji ósmej wejście adresowe ma postać 110 co odpowiada wejściu D6, które jest w stanie wysokim więc stan na wyjściu nie zmienia się. W sekcji dziewiątej wejście adresowe multipleksera ma postać 111 co odpowiada wejściu D7, które jest w stanie niskim więc stan na wyjściu zmienia się na niski. W sekcji dziesiątej wejście multipleksera jest w stanie 00011110 i układ generuje taki przebieg na wyjściu w dotychczas opisany sposób.

W programie Digital Works znajdują się rzeczywiste liczniki (74LS90, 74HC161, 74HC163, 74HC390, 74HC4040, 74HC4024, 74HC4017) i multiplekser (74HC157) w postaci układów scalonych, do których została dodana dokumentacja techniczna.

5. ZAKOŃCZENIE

W niniejszej pracy inżynierskiej pokazano najlepsze (zdaniem autora) darmowe programy przeznaczone do symulacji elektronicznych układów cyfrowych: CircuitMod, LogicCircuit, Logisim, Atanua i Digital Works, który wybrano jako program główny do symulacji układów cyfrowych przedstawionych w tej pracy. Jak wspomniano we wstępie, nie istnieją darmowe programy do symulacji wszystkich układów, a w szczególności takich zawierających procesory lub mikrokontrolery, ponieważ wymaga to użycia zaimplementowanego symulatora programowania. Wybierając darmowe programy należy liczyć się z ograniczonymi możliwościami, jednakże całkowite przestudiowanie ich instrukcji obsługi pozwoli na wykorzystanie maksymalnych możliwości, które powinny w zupełności wystarczyć uczniom techników, studentom uczelni wyższych oraz początkującym elektronikom. Należy wspomnieć o darmowych programach wspomagających projektowanie układów elektronicznych (np. EasyEDA), które ostatnimi czasy coraz częściej bazują na przeglądarce internetowej. Większość takich programów również posiada możliwość symulacji lecz jej procedura jest nieco bardziej skomplikowana i niewygodna, ponieważ najczęściej wykorzystuje się do tego skrypty, których składni trzeba się nauczyć – może się to okazać czasochłonne. Tego typu programy bazują głównie na rzeczywistych elementach elektronicznych oferowanych przez najpopularniejszych producentów.

Program Digital Works okazał się najbardziej optymalnym programem, który nie powinien sprawić większych problemów początkującym studentom lub elektronikom. Łatwa i intuicyjna obsługa, oscyloskop, wszystkie podstawowe elementy cyfrowe, możliwość projektowania własnych dowolnie rozbudowanych układów w obudowach typu DIL z możliwością umieszczenia dokumentacji – wszystkie te zalety w głównej mierze przyczyniły się do jego wyboru. Program posiada kilka wad. Brak możliwości przybliżania przeszkadza przy projektowaniu symboli własnych układów. Nie działa kombinacja *Ctrl+Z*, która umożliwia szybkie przywrócenie przypadkowo usuniętych elementów. Tworzenie układów o dużej liczbie połączeń może stać się bardzo pracochłonne, a ich schemat może okazać się nieczytelny. Najlepszym rozwiązaniem byłoby tutaj umożliwienie tworzenia połączeń za pomocą przypisanych nazw do wyprowadzeń lub za pomocą szyny magistralowej. Tego typu rozwiązanie można spotkać w programie Eagle lub omówionym Logisim. Brak symulacji programowania mikrokontrolerów ogranicza ten program do jeszcze szerszych zastosowań. Pomimo tych

wad z powodzeniem udało się przeprowadzić symulację i analizę przedstawionych układów cyfrowych. Ciekawym zadaniem w celu edukacyjnym dla początkującego elektronika lub studenta, który nie posiada jeszcze doświadczenia w projektowaniu rzeczywistych układów elektronicznych byłoby skonstruowanie przedstawionych układów z podstawowych elementów elektronicznych na płytce PCB i sprawdzenie ich działania.

6. BIBLIOGRAFIA

[1] Barker David John, Plik pomocy programu Digital Works, https://www.mecanique.co.uk/shop/index.php?route=product/category&path=89, 2014.

[2] Burch Carl, Plik pomocy programu Logisim, http://www.cburch.com/logisim/, 2011.

[3] Falstad Paul, CircuitMod, http://circuitmod.sourceforge.net/, 2007.

[4] Głocki Wojciech, Układy Cyfrowe, Wydawnictwa Szkolne i Pedagogiczne, Warszawa, 1998.

[5] Komppa Jari, Atanua, http://sol.gfxile.net/atanua/downloads.html, 2013.

[6] Ramalhete Bruno i in., Logic Circuit, http://www.logiccircuit.org/download.html, 2017.

[7] Wałaszek Jerzy, Bit w zastosowaniach, https://eduinf.waw.pl/inf/alg/ 002_struct /index.php, 2012.

[8] Wilkinson Barry, Układy Cyfrowe, Wydawnictwa Komunikacji i Łączności, Warszawa, 2003.