



Uniwersytet w Białymstoku

Instytut Informatyki

Aplikacja symulująca działanie
sumatora/subtraktora w oparciu o jego
cyfrowe układy logiczne

Artur Bucki
Numer albumu: 80212

Promotor:
DR INŻ. WIESŁAW PÓŁJANOWICZ

Białystok 2022r

Spis treści

1	Wstęp	3
2	Organizacja i architektura klasycznego komputera	5
2.1	Arytmetyka w systemach cyfrowych	5
2.1.1	Pozycyjne systemy liczbowe	8
2.2	Układy cyfrowe - bramki logiczne	9
2.3	Procesor	12
2.3.1	Jednostka arytmetyczno-logiczna (ALU)	13
2.3.2	Jednostka sterująca	14
2.3.3	Zespół rejestrów	15
2.4	Pamięć	16
2.5	Urządzenia wejścia/wyjścia	17
2.6	Magistrale systemowe	18
3	Działanie jednostki arytmetyczno-logicznej ALU	21
3.1	Układ sumatora/subtraktora	21
3.2	Działanie układu ALU Simulator - EE3221 Digital Systems II	23
4	Programy symulujące działanie układów cyfrowych w komputerze	26
4.1	Digital Works	26
4.2	Cedar logic Simulator	28
4.3	Win Logi Lab	31
4.4	Multimedia Logic	33
4.5	Logisim	36
5	Technologie informatyczne wykorzystywane przy budowie aplikacji symulujących działanie układów cyfrowych	39
5.1	C#	39
5.2	Adobe Photoshop	40
6	Projekt i realizacja aplikacji symulującej działanie sumatora/subtraktora (Add Sub)	42
6.1	Wymagania funkcjonalne i niefunkcjonalne aplikacji	43
6.2	Budowa modułowa aplikacji (Add Sub)	44
6.3	Testowanie aplikacji	47
7	Podręcznik użytkownika aplikacji	49
8	Podsumowanie	51

9 Bibliografia	52
10 Spis tablic/rysunków	54

1 Wstęp

Informatyka należy do dyscyplin naukowych, które charakteryzują się szerokim obszarem zainteresowań badawczych. W ramach informatyki występują zarówno aspekty związane z przetwarzaniem informacji, jak również tworzeniem programów i części komputerowych, opisem procesów algorytmicznych, czy rozwiązywaniem problemów obliczeniowych. Specyfikacja niniejszej pracy ukierunkowana jest na kwestie przetwarzania informacji oraz budowy i działania elektronicznej części komputera.

Techniczne dziedziny, takie jak informatyka i elektronika są najszybciej rozwijającymi się branżami w obecnych czasach. Przedsiębiorstwa chcące sprostać oczekiwaniom swoich klientów ciągle rozwijają gałąź związaną z wszelką elektroniką. Urządzenia elektroniczne poddawane są różnorodnym modyfikacjom i ulepszeniom, tak aby otrzymać sprzęt szybszy i wydajniejszy. Istnieją jednak schematy, które bardzo ciężko jest przeskoczyć i zaproponować lepsze rozwiązanie. Koncepcja architektury von Neumanna została opracowana w 1945 roku, a wykorzystywana jest do tej pory w dzisiejszych komputerach. Mikroprocesory, jak i pamięci, czy inne urządzenia wchodzące w skład komputera ciągle są rozwijane pod względem mocy, wielkości czy szybkości, lecz sam schemat działania i przeznaczenia jest niemal identyczny. Reprezentacja informacji w komputerze jest niezmienną od 1938 roku, w którym została stworzona pierwsza maszyna licząca używająca symboli 0 oraz 1. Sam system binarny był jednak używany już w 1679 roku. Taki ciąg zdarzeń pokazuje jak bardzo dobra koncepcja może utrzymywać się przez wiele dekad.[1]

Architektura von Neumanna pozwala rozbić procesor na: jednostkę arytmetyczno-logiczną (ALU), jednostkę sterującą oraz zespół rejestrów. W ALU jednym z ważniejszych układów cyfrowych jest właśnie sumator. Oczywiście układ ten znajduje się także w innych częściach procesora. Jest jednak niezastąpioną częścią, która realizuje wiele ważnych funkcji. [1]

Celem pracy licencjackiej jest stworzenie aplikacji symulującej działanie układu sumatora/subtraktora w oparciu o jego cyfrowe układy logiczne. Praca składa się z siedmiu rozdziałów.

Szczegółowa zawartość rozdziałów:

- W rozdziale II omówiono organizację i architekturę klasycznego komputera, sposób reprezentacji liczb w urządzeniach elektronicznych, bramki logiczne wykorzystywane w elektronice, funkcję procesora, budowę i za-

sadę działania. Podstawowe urządzenia wejścia-wyjścia oraz magistralę systemowe.

- W rozdziale III skupiono się głównie na zasadzie działania jednostki arytmetyczno-logicznej (ALU). Przedstawiono przykładowy układ sumatora/subtraktora oraz jego układ elektroniczny. Dodatkowo pokazano działanie gotowego symulatora tekstowego.
- W rozdziale IV pokazano różne programy pozwalające na tworzenie układów cyfrowych oraz ich symulację. Dzięki nim można tworzyć własne koncepcje oraz realizować określone projekty.
- W rozdziale V przedstawiono programy, które mogą być przydatne do zrealizowania własnego programu graficznego pozwalającego na symulację działania układów cyfrowych. Zostały zaprezentowane tam opcje, które zostały wykorzystane przy budowie symulacji sumatora/subtraktora.
- W rozdziale VI zrealizowano praktyczną część projektu, czyli realizację aplikacji symulującej działanie sumatora/subtraktora. Dokonano testów programu oraz przedstawiono wymagania funkcjonalne i nie-funkcjonalne, oraz podzielono ją na budowę modułową.
- W rozdziale VII pokazano sposób działania aplikacji i podręcznik jej użytkowania.

2 Organizacja i architektura klasycznego komputera

Architektura klasycznego komputera została opracowana w 1945 roku, przez matematyka John'a Von Neumanna oraz jego współpracowników - John'a W. Mauchly'ego i Johna Presper Eckerta. W pierwszej połowie XX wieku maszyny obliczeniowe były tworzone w konkretnym celu, wykonywały tylko i wyłącznie określone zadania, nie były modyfikowalne, jeśli chodzi o samo działanie danego programu. Klasyczny komputer został zaprojektowany tak, aby mógł wykonywać różne rozkazy, pozwalało to na większą swobodę oraz zaoszczędzenie czasu i energii. Maszyna licząca zaprojektowana przez Von Neumanna składa się z trzech głównych komponentów - procesora, pamięci oraz urządzeń wejścia-wyjścia. Takie zestawienie pozwala użytkownikowi na pełną modyfikację, kompilację oraz uruchamianie aplikacji. W pamięci komputera przechowywana jest informacja zapisana w postaci binarnej, która jest przekazywana łańcuchowo. Procesor dekoduje informacje i potokowo je przetwarza. Użytkownik, jako osoba decyzyjna, integruje się z maszyną za pomocą urządzeń wejścia-wyjścia, deklaruje w jaki sposób i kiedy program powinien zostać uruchomiony. Dlatego aby wszystko funkcjonowało poprawnie maszyna powinna posiadać skończony zestaw instrukcji, który jest ogólnie dostępny dla procesora wraz z danymi. [2]

2.1 Arytmetyka w systemach cyfrowych

Systemy cyfrowe to elektroniczne urządzenia, które bazują na określonej strukturze alfanumerycznej. Najczęściej spotykanym zapisem danych w elektronice jest naturalny kod binarny, w którym informacje reprezentują cyfry o wartości 1 lub 0. Przykładami takiego sprzętu mogą być telefony komórkowe lub radio. Liczby binarne zazwyczaj reprezentują określoną wartość np. 0 - fałsz, 1 - prawda lub 0 - nie, 1 - tak. Oczywiście wartość danego bitu zależy od miejsca, w którym się znajduje. Do reprezentowania większych ilości danych można przedstawiać informacje za pomocą listy bitów, czyli ciągu znaków złożonych z zer oraz jedynek.

Arytmetyka w słowniku języka polskiego oznacza dział matematyki, odnoszący się do podstawowych zasad działania na liczbach. W praktyce oznacza to, że zasady dodawania, odejmowania, dzielenia oraz mnożenia obowiązują także w systemach cyfrowych. Do reprezentacji liczb naturalnych najczęściej stosuje się kody - NKB, Gray'a, 1 z 10, BCD oraz Johnsona. Natomiast liczby całkowite używają trzech podstawowych kodów - znak-moduł, zapis uzupełnień do jedności (U1) oraz zapis uzupełnień do dwóch (U2).

W naturalnym kodzie binarnym działania arytmetyczne możemy porównać do tych które są nam znane z systemu dziesiętnego. Jeśli chcielibyśmy dodać liczbę A oraz liczbę B możemy użyć odpowiedniej tabliczki dodawania dla danego kodu. [4]

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

Rysunek 1: Schemat dodawania liczb binarnych

Liczba A: 101010

Liczba B: 011101

$$\begin{array}{r}
 1 1 \\
 1 0 1 0 1 0 \\
 + 1 1 1 0 1 \\
 \hline
 1 0 0 0 1 1 1
 \end{array}$$

Rysunek 2: Przykład dodawania liczb binarnych

Dwójkowy system liczbowy posiada dwa stany 0 lub 1, dlatego aby zapisać liczbę ujemną, wymagany jest pewien system arytmetyczny, który na to pozwoli. Aby rozwiązać ten problem, wprowadzono bit znaku, który reprezentuje wartość ujemną lub dodatnią. Najbardziej popularnym i najczęściej używanym kodem jest zapis uzupełnień do dwóch. Rozwiązuje on problem podwójnego zera, który powstaje w przypadku U1 oraz ZM. Dodatkowo przeniesienia z bitu znakowego są ignorowane, co eliminuje wadę dodatkowych obliczeń. Najstarszy bit określa znak liczby. Jeśli bit jest ustawiony na “1”, to liczba jest ujemna, w przeciwnym wypadku - dodatnia.

Tablica 1: Wartości dodatnie dla 4 bitów - U2

Wartość	Zapis U2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Tablica 2: Wartości ujemne dla 4 bitów - U2

Wartość	Zapis U2
-8	1000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001

Działania arytmetyczne w kodzie U2 działają na takiej samej zasadzie jak w naturalnym kodzie binarnym. Jedyną różnicą jest interpretacja danego wyniku. Należy także pamiętać o sytuacji, w której może dojść do przepełnienia, czyli liczbie, która nie mieści się w danym zakresie. Overflow powstaje, gdy przeniesienie, które wchodzi do bitu znaku ma przeciwną wartość niż przeniesienie, które wychodzi z bitu znaku. Przykładem takiej sytuacji może być dodawanie dwóch liczb gdy, liczba A = 7 oraz liczba B = 3 w 4-bitowym formacie.

Liczba A: 0111

Liczba B: 0011

$$\begin{array}{rcccc}
 & & 0 & 1 & 1 & 1 \\
 +7 & & & 0 & 1 & 1 & 1 \\
 +3 & + & & 0 & 0 & 1 & 1 \\
 \hline
 & & 1 & 0 & 1 & 0 &
 \end{array}
 \quad \text{Wynik} = -6$$

Rysunek 3: Dodawanie liczb U2 - overflow

2.1.1 Pozycyjne systemy liczbowe

W matematyce, aby zapisywać liczby, stosuje się określony system, który jest zbudowany ze skończonej ilości znaków. System pozycyjny charakteryzuje się tym, że jego liczby są zapisywane od lewej do prawej. Ponieważ mamy skończoną ilość znaków, to pojedyncze symbole, możemy zapisać tylko dla kilku początkowych liczb. Aby zapisać liczbę większą niż podstawa danego systemu, tworzymy ciąg znaków, który odpowiednio odczytujemy. Każda cyfra posiada określoną wartość zależną od miejsca, w którym się znajduje, oraz systemu, w którym jest zapisana. Podstawa systemu pozycyjnego określa, ile symboli znajduje się w danym zbiorze. Aby policzyć wartość danej liczby ustawiamy jej cyfry na określonych pozycjach, zaczynając od strony prawej oraz numerując je od zera. Każda pozycja posiada wartość, która jest nazywana wagą pozycji. Liczby podnosimy do potęgi zależnie od podstawy danego systemu oraz miejsca danej cyfry. Miejsce określa, ile razy waga danej pozycji uczestniczy w wartości liczby. Na koniec sumujemy iloczyn cyfr poprzez wagi ich pozycji. [7]

Tablica 3: Schemat - pozycyjne systemy liczbowe

Waga	p^n	p^{n-1}	p^{n-2}	p^2	p^1	p^0
Cyfra	C	C	C...	C	C	C
Pozycja	n	n - 1	n - 2	2	1	0

C - cyfra o podstawie p

p - podstawa systemu pozycyjnego

Najbardziej powszechnym systemem pozycyjnym jest system dziesiętny, którym posługujemy się na co dzień. Jak sama nazwa wskazuje posiada on

10 cyfr licząc od 0 do 9. Na jego przykładzie można wyprowadzić wzór, który sprawdzi się dla każdego innego pozycyjnego systemu liczbowego.

Przykład: 645_{10}

Wynik - $6 * 10^2 + 4 * 10^1 + 5 * 10^0 = 645$

Wzór - $C_{n-1} * p^{n-1} + C_{n-2} * p^{n-2} + \dots + C_2 * p^2 + C_1 * p^1 + C_0 * p^0$

Komputer jest maszyną binarną, dlatego nie jest możliwe, aby informacje w nim zostały zapisywane w postaci dziesiętnej. Aby zamienić język maszynowy na inny określony system liczbowy, wymagana jest konwersja. Każdą liczbę dziesiętną można wyrazić w innym zapisie, tak jak każdą liczbę w innym zapisie można wyrazić w postaci dziesiętnej. Pomijając system dwójkowy, bardzo popularny jest także system ósemkowy oraz szesnastkowy. Aby zamienić liczbę dziesiętną na binarną, wystarczy napisać wagi poszczególnych miejsc łańcucha, po czym uzupełnić odpowiednie miejsca jedynekami i dodać do siebie wagi, pod którymi znajduje się liczba 1. [6]

Przykład: $29_{10} \rightarrow 11101_2$

$$\begin{array}{cccccc} 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 32 & 16 & 8 & 4 & 2 & 1 \end{array} \begin{array}{l} \leftarrow \text{zapis dwójkowy} \\ \leftarrow \text{wagi} \end{array}$$

$$16 + 8 + 4 + 1 = 29$$

Oczywiście możemy także dokonać odwrotnej konwersji i zamienić system binarny na dziesiętny.

Przykład: $10100_2 \rightarrow 20_{10}$

$$\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 32 & 16 & 8 & 4 & 2 & 1 \end{array} \begin{array}{l} \leftarrow \text{zapis dwójkowy} \\ \leftarrow \text{wagi} \end{array}$$

$$16 + 4 = 20$$

2.2 Układy cyfrowe - bramki logiczne

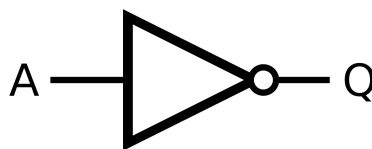
Bramki logiczne to fizyczne elementy występujące w procesorach, zbudowane są za pomocą tranzystorów. Można je sobie wyobrazić jako skrzynkę,

do której wchodzi linie wejściowe i z której wychodzi linia wyjściowa. Same linie to przewody elektryczne w których może płynąć prąd. Przewody mogą mieć dwa stany, wysoki oraz niski. [8]

- Stan wysoki, prąd płynie przez przewód, otrzymujemy na wejściu wartość binarną 1.
- Stan niski, prąd nie płynie przez przewód, otrzymujemy na wejściu wartość binarną 0.

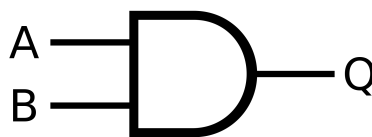
Każda bramka logiczna zamienia sygnały wejściowe na odpowiedni sygnał wyjściowy. Najprostsze bramki posiadają dwa wejścia, ale także istnieją modele z wieloma wejściami. Należy także zwrócić uwagę, że zawsze występuje tylko jedno wyjście, o wartości 1 lub 0.

Najczęściej wykorzystywane bramki logiczne:



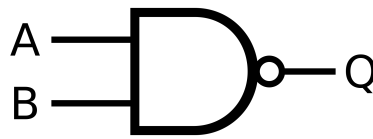
Rysunek 4: Bramka logiczna - NOT

Bramka logiczna NOT jest najprostszą pod względem działania, zamienia sygnał wejściowy na przeciwny, jeżeli na wejściu nadamy sygnał o wartości 1 to na wyjściu pojawi się 0. Natomiast jeżeli na wejściu pojawi się 0 to otrzymamy 1.



Rysunek 5: Bramka logiczna - AND

Bramka logiczna AND daje wartość 1 tylko i wyłącznie wtedy, gdy na wszystkich wejściach otrzymujemy stan wysoki. W przeciwnym wypadku otrzymujemy 0.



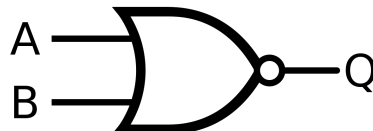
Rysunek 6: Bramka logiczna - NAND

Bramka logiczna NAND jest dokładnie odwrotnością bramki AND, tylko i wyłącznie, gdy na wejściu, na wszystkich stanach pojawia się wartość binarna 1, to otrzymamy na wyjściu 0. W przeciwnym wypadku zawsze otrzymamy wartość 0.



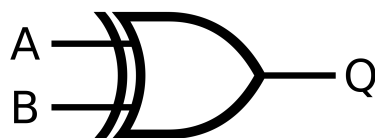
Rysunek 7: Bramka logiczna - OR

Bramka logiczna OR daje wartość 1 zawsze, gdy któreś z wejść będzie w stanie wysokim, w przeciwnym wypadku będzie wartość 0.



Rysunek 8: Bramka logiczna - NOR

Bramka logiczna NOR jest dokładnie odwrotnością bramki OR, tylko i wyłącznie, gdy na wejściu, na wszystkich stanach pojawi się wartość binarna 0, to otrzymamy na wyjściu 1. W przeciwnym wypadku zawsze będzie to wartość 0.



Rysunek 9: Bramka logiczna - XOR

Bramka logiczna XOR, jest wyjątkowa, zawsze na wejściu posiada tylko dwie zmienne. Gdy jedna z nich posiada wartość 1 to wyjście jest równe logicznej jedynce. W przeciwnym wypadku otrzymujemy wartość 0.

Tabela wartości bramek logicznych:

Tablica 4: Tabela prawdy - bramki logiczne

A	B	AND	NAND	OR	NOR	XOR
0	0	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0

2.3 Procesor

W dzisiejszym świecie procesor odgrywa naprawdę ważną rolę, jest to jednostka, która znajduje się niemal w każdym urządzeniu elektronicznym. Jest odpowiedzialny za przetwarzanie informacji oraz wykonywanie instrukcji. Można powiedzieć, że pełni rolę “mózgu” we wszelkiej elektronice. Procesor jest budowany przy pomocy mikroskopijnych tranzystorów, które sterują określonymi zadaniami, są to w istocie bramki logiczne, które włączają się lub wyłączają przekazując wartość binarną jako informacje. Podstawową czynnością procesora jest wykonywanie programów, czyli inaczej sekwencji zapisanych instrukcji. Aby wykonać określone zadanie, procesor na początku pobiera dane z pamięci RAM systemu, następnie dekoduje informacje oraz przetwarza je tak, aby określone części procesora wykonały swoją pracę. Na koniec wywołuje daną instrukcję, określona część procesora uaktywnia się, dając odpowiedni rezultat. [9]

Główne elementy procesora:

- Jednostka arytmetyczno-logiczna (ALU)
- Jednostka sterująca

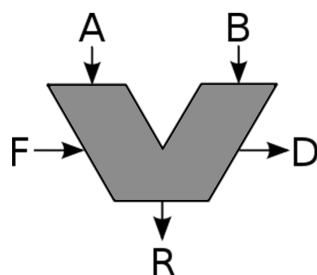
- Zespół rejestrów

Największymi producentami procesorów, które znajdują się w większości komputerów osobistych jest Intel oraz AMD. Warto nadmienić, że nie każdy procesor jest taki sam. Szybkość przetwarzania informacji w mikroprocesorach zależy od rdzeni, wątków oraz szybkości zegarów. Rdzeń w procesorze jest fizyczną częścią, która odpowiada za realizację operacji obliczeniowych, jeden rdzeń może pracować nad jednym zadaniem. Wątek ściśle zależy od ilości rdzeni, dzieli rdzeń na dwa wirtualne, dzięki czemu mogą one wykonywać dwie linie wykonawcze jednocześnie. W niektórych przypadkach może zwiększyć to wydajność określonej instrukcji. Wątki są mniej wydajne od rdzeni procesora, ponieważ korzystają z tych samych zasobów. Szybkość zegarów odgrywa także kluczową rolę, w dzisiejszych procesorach wyraża się ją w GHz. Określa ona, ile rozkazów może obsłużyć procesor w ciągu sekundy.

Pierwsze procesory posiadały tylko i wyłącznie jeden rdzeń. Aktualnie w większości, mikroprocesory są wielordzeniowe. Można powiedzieć, że dzięki temu na jednym sprzęcie posiadamy kilka oddzielnych procesorów. Współcześnie procesory w komputerach osobistych zazwyczaj posiadają od 2 do nawet 32 rdzeni.

2.3.1 Jednostka arytmetyczno-logiczna (ALU)

Jednostka arytmetyczno-logiczna (ALU), to układ cyfrowy służący do wykonywania operacji arytmetycznych oraz logicznych, pomiędzy dwoma liczbami. Jest podstawowym blokiem procesora i jest wykorzystywany praktycznie zawsze przez resztę elementów CPU. Jak sama nazwa wskazuje, jednostka składa się z dwóch części - arytmetycznej oraz logicznej. Część arytmetyczna zajmuje się takimi operacjami jak odejmowanie czy dodawanie. Zatem, część logiczna, pozwala na zastosowanie operacji takich jak Ex-Or pomiędzy dwoma liczbami. Blok ALU może także wykonywać operacje jednoargumentowe – negacje bitów, przesunięcia bitowe czy nawet inkrementacje. Najczęściej ALU posiada dwa wejścia oraz jedno wyjście wynikowe. Warto zaznaczyć, że niektóre operacje arytmetyczne takie jak np. dzielenie, jest droższe i trudniejsze w implementacji niż dodawanie. [13]



Rysunek 10: Typowy symbol - ALU

Źródło: https://pl.wikipedia.org/wiki/Jednostka_arytmetyczno-logiczna

Gdzie:

A i B - Operandy;

R - Wyjście;

F - Wejście z jednostki kontrolnej

D - Status wyjścia

Pierwotne procesory zawierały tylko i wyłącznie jedno ALU. Aktualnie architektura pozwala na implementację kilku bloków w procesorze. Dzięki temu jednostki mogą wykonywać różne instrukcje w jednym czasie. Często także są to dwa różne typy jednostek, niektóre z nich są precyzowane na określone zadanie np. mnożenie. Wiele typów innych urządzeń elektronicznych niż komputer wykorzystuje miniaturowe wersje ALU. Najbliższym przykładem może być zegarek elektroniczny, który inkrementuje sekundy.

2.3.2 Jednostka sterująca

Jednostka sterująca jest częścią procesora która kieruje instrukcjami wykonywanymi przez CPU urządzenia. Pobiera informacje z pamięci komputera oraz urządzeń wejścia/wyjścia, a następnie przekształca odpowiednio sygnały, tak, aby przekazać je dalej procesorowi, który następnie wykonuje dane zadanie. Blok jednostki sterującej jest zaprojektowany w taki sposób, że wykonuje rozkazy i “zarządza” różnymi elementami. Posiada różne funkcje, które są wymagane do prawidłowego funkcjonowania procesora. Sama sekcja sterującą także ściśle związana jest z ALU, ponieważ kontroluje jej działanie, oraz także innych jednostek wykonawczych. Warto nadmienić także, że obsługuje wiele zadań takich jak pobieranie, dekodowanie czy przechowywanie wyników. Służy do interpretacji rezultatów oraz steruje przepływami danych wewnątrz procesora. [12]

Można rozróżnić dwa typy budowy jednostki sterującej:

- Układowa jednostka sterująca
- Mikroprogramowalna jednostka sterująca

Układowa jednostka sterująca jest skonstruowana w taki sposób, że układ logiczny może generować sygnały sterujące bez zmiany struktury układu. Dzięki tej metodzie procesor otrzymuje dane które nie mogą być modyfikowane. Rekordy zostają przesłane do dekodera, który konwertuje informacje po czym powstają sygnały wyjściowe. Następnie przekazywane są do generatora macierzy, który tworzy sygnały sterujące do wykonania określonego programu.

Mikroprogramowalna jednostka sterująca współgra wraz z pamięcią urządzenia. Działa na zasadzie przechowywania sygnałów sterujących. Dekodowanie danej informacji odbywa się w trakcie wykonywania danego programu. Budowa tej jednostki, jak sama nazwa wskazuje, realizuje mikrooperacje, które wykonuje się w celu realizacji mikroinstrukcji.

2.3.3 Zespół rejestrów

Zespół rejestrów jest pamięcią, z której korzysta procesor. Przechowywane są w nim niewielkie dane, najczęściej o rozmiarze od 4 do 128 bitów. W hierarchii samych pamięci plasuje się na samym szczycie, ponieważ jest najszybszą możliwą drogą do komunikacji z mikroprocesorem. Może przechowywać informacje o instrukcjach, adresach pamięci, sekwencjach bitów lub pojedynczych znakach. Większość procesorów wykonuje operacje na tym typie danych przenosząc je bezpośrednio z pamięci urządzenia. Po zakończonej pracy zwraca wyniki do poprzedniego obszaru i opróżnia miejsce w rejestrze. Zespół rejestrów jest wymagany w procesorze, ponieważ dzięki niemu, wszystkie dane które tam trafiają mogą być sprawnie manipulowane.[21]

Dwa najważniejsze rodzaje rejestrów to:

- Rejestr danych
- Rejestr adresowy

Rejestry danych przechowują informacje o argumentach, wynikach obliczeń, itd. składowane są tam rekordy całkowitoliczbowe.

Rejestr adresowy przechowują informacje o adresach pamięci, dodatkowo służą do obliczania adresów następnych instrukcji, które są wykonywane se-

kwencyjnie.

Tablica 5: Najczęściej wykorzystywane rejestry

Źródło: <https://www.javatpoint.com/computer-registers>

Rejestr	Symbol	Ilość bitów	Funkcja
Rejestr danych	DR	16	Dane całkowitoliczbowe
Rejestr adresowy	AR	12	Adres pamięci
Rejestr wejściowy	INPR	8	Znak wejściowy
Rejestr wyjściowy	OUTR	8	Znak wyjściowy
Licznik programu	PC	12	Adres instrukcji
Rejestr tymczasowy	TR	16	Dane tymczasowe
Rejestr instrukcji	IR	16	Kod instrukcji
Akumulator	AC	16	Rejestr procesora

2.4 Pamięć

Pamięć to fizyczne urządzenie, które może przechowywać dane tymczasowe. Jest to narzędzie, które wykorzystuje układy scalone. Używa się jej w oprogramowaniach oraz systemach operacyjnych. Można ją podzielić na dwa typy: ulotną oraz stałą

- Pamięć ulotna przechowuje informacje, które po wyłączeniu jednostki tracą swoją zawartość. Przykładem takiej pamięci może być RAM w komputerze. Używa się ich, ponieważ są bardzo szybkie, w porównaniu do pamięci stałych.
- Pamięć stała z drugiej strony zapisuje swoje dane nawet po wyłączeniu zasilania. Często jest nazywana NVRAM, przykładem może być pamięć EPROM.

Zawartość pamięci może być różna - przechowywane są w niej informacje wejściowe, wyjściowe, wyniki czy instrukcje. Podstawową jednostką jest bit i to on reprezentuje określone dane. Pamięć tą można scharakteryzować na podstawie dwóch czynników - pojemności oraz prędkości. Pierwsze z nich określa, ile danych może przechowywać dana jednostka, drugie zaś, mówi o czasie dostępu pomiędzy żądaniem odczytu i zapisu. Często pamięć pierwotna jest mylona z pamięcią wtórną, ponieważ pamięć jako pojęcie jest tożsame. Pamięć wtórna, inaczej też nazywana jako masowa, odnosi się do dysków twardych np. HDD. Informacje w takich jednostkach są stałe. Różnica pomiędzy tymi pamięciami nie wynika tylko z tego jak są przechowywane

dane. Zazwyczaj komputery osobiste posiadają mniej pamięci RAM niż pamięci masowej. Wynika to z faktu, że programy tymczasowe, nie potrzebują aż tak dużej pojemności, aby je uruchomić, czy na nich pracować. Za to potrzebują szybkiego dostępu i dlatego RAM odgrywa bardzo ważną rolę. Oczywiście, aby wszystko funkcjonowało poprawnie w komputerze, wymagane są dwa typy pamięci. Działa to na takiej zasadzie, że pamięci komunikują się ze sobą. Informacje są pobierane z pamięci masowej i przekazywane do pamięci wtórnej. [23]

Pamięć ROM (read-only memory) czyli tylko do odczytu, można podzielić na: PROM, EPROM oraz EEPROM.

PROM (programmable read-only memory) jest układem pamięci programowalnej, może zostać zaprogramowana tylko raz. Przykładem tej pamięci jest BIOS, czyli program używany przez procesor do uruchamiania komputera oraz systemu komputerowego.

EEPROM (electrically erasable programmable read-only memory) - jest układem pamięci programowalnej, różni się tym od PROM, że jej zawartość może być kasowana i ponownie programowana. Aktualnie jest używana w komputerach osobistych, zastąpiła starsze wersje PROM oraz EPROM

Pamięć RAM (random-access memory) czyli pamięć o swobodnym dostępie można podzielić na: EDO RAM, SDRAM, DDR RAM.

SDRAM jest pamięcią, która ma możliwość synchronizowania się z zegarem systemowym, dzięki temu może pracować z większą szybkością.

Najczęściej używaną pamięcią w komputerach osobistych aktualnie jest DDR4. Jest rodzajem pamięci SDRAM. Została opracowana w 2014r. Prędkość pamięci DDR4 wynosi pomiędzy 800 a 1600MHz. Za to pojemność waha się od 4 do 128GB.

2.5 Urządzenia wejścia/wyjścia

Urządzenia wejścia-wyjścia pozwalają na komunikację użytkownika z komputerem lub innym urządzeniem elektronicznym. Dzięki nim możliwe jest wprowadzenie danych do komputera jak i ich wyprowadzanie. Urządzenia wejściowe przesyłają informacje, które następnie są przetwarzane w urzą-

dzeniu. Natomiast urządzenia wyjściowe, pozwalają na wyświetlanie czy odtwarzanie tego, co aktualnie wykonujemy. Większość urządzeń można podzielić na typowo wyjściowe lub wejściowe, ale oczywiście istnieją hybrydy, które wykonują obydwie czynności. Warto nadmienić także, że nie wszystkie urządzenia wejścia-wyjścia muszą być podłączone za pomocą kabli. Istnieją nowości, które pozwalają na komunikację poprzez fale radiowe lub podczerwień. Dodatkowo komputery posiadają takie urządzenia wewnątrz obudowy, przykładem może być płyta główna, która posiada czujniki temperatury czy zasilania.[24]

Urządzenia wejścia:

- Klawiatura, mysz komputerowa - przyjmują dane wejściowe od użytkownika, nie mogą przyjmować informacji ani ich odtwarzać.
- Mikrofon – odbiera fale dźwiękowe generowane przez źródło wejściowe a następnie wysyła je do komputera.
- Kamera internetowa – odbiera obraz a następnie wysyła je do komputera.

Urządzenia wyjścia:

- Monitor – odbiera dane z komputera, dzięki niemu mamy wgląd na to, co robimy, wyświetla informacje w postaci tekstów i obrazów.
- Głośniki - odbierają dane dźwiękowe z komputera, pozwalają na odtwarzanie słyszalnych dźwięków użytkownikowi.

Urządzenia wejścia-wyjścia:

- Pamięć USB – odbiera lub zapisuje dane z komputera, dodatkowo nośnik wysyła dane do komputera.
- Napęd CD – odbiera dane z komputera w celu skopiowania ich na dysk, dodatkowo wysyła dane do komputera.

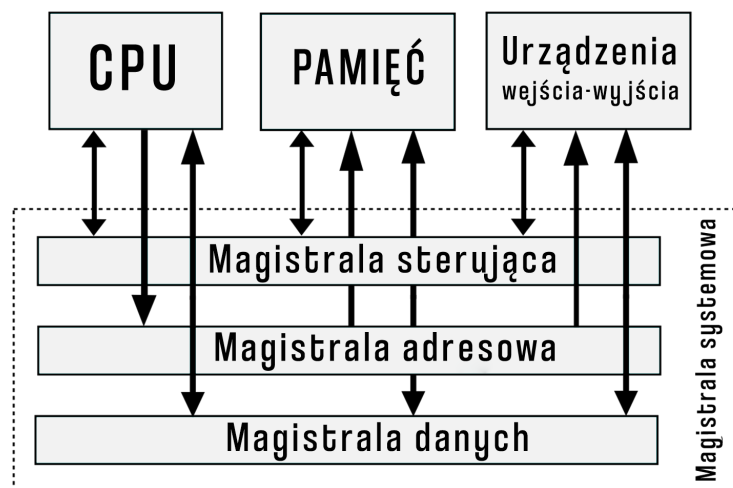
2.6 Magistrale systemowe

Magistrala systemowa jest ścieżką składającą się z złączy oraz kabli. Pozwala na przesyłanie danych pomiędzy różnymi elementami komputera. Łączy procesor z pamięcią RAM, dyskiem twardym, urządzeniami wejścia-wyjścia oraz innymi komponentami. [25]

Magistrala systemowa posiada trzy podstawowe rodzaje szyn:

- Sterująca - przenosi sygnały sterujące, służy do zarządzania, określa rodzaj operacji.

- Adresowa - określa miejsce, gdzie powinny być przenoszone dane aktualnej operacji.
- Danych – pozwala na przenoszenie danych rzeczywistych, między procesorem, pamięcią oraz urządzeniami wejścia-wyjścia



Rysunek 11: Magistrala systemowa

Źródło: https://pl.wikipedia.org/wiki/Szyna_danych.

Konstrukcja magistrali systemowej może być różna, zależy od potrzeb określonego procesora, systemu czy długości słowa danych. Przesył danych można wyrazić w bitach i jest zależny od ilości przewodów lub złącz znajdujących się w magistrali. Na przykład, jeśli mamy możliwość przesyłania jednocześnie 32 bitów, to nasza magistrala posiada również 32 przewody lub złącza. Rozmiar i konstrukcja danej magistrali, zależy od szybkości oraz ilości przesyłanych danych w tym samym czasie. Można je także podzielić na typ oraz sposób prowadzonej transmisji. [25]

Podział ze względu na typ prowadzonej transmisji:

- Równoległe - informacje są przenoszone równoległe przewodami lub ścieżkami. Przykładami takich magistral są złącza PCI, AGP czy FSB
- Szeregowe – informacje są przenoszone szeregowo kanałami. Przykładami takich magistral są złącza USB, RS-232 oraz PCI Express

Podział ze względu na sposób prowadzonej transmisji:

- Jednokierunkowe – dane przesyłane są tylko w jednym kierunku, transmisja odbywa się jednostronnie z nadajnika do odbiornika.
- Dwukierunkowe – dane przesyłane są w obu kierunkach, transmisja odbywa się w dwie strony z nadajnika do odbiornika lub z odbiornika do nadajnika. Możliwa jest także transmisja danych jednocześnie w dwie strony.

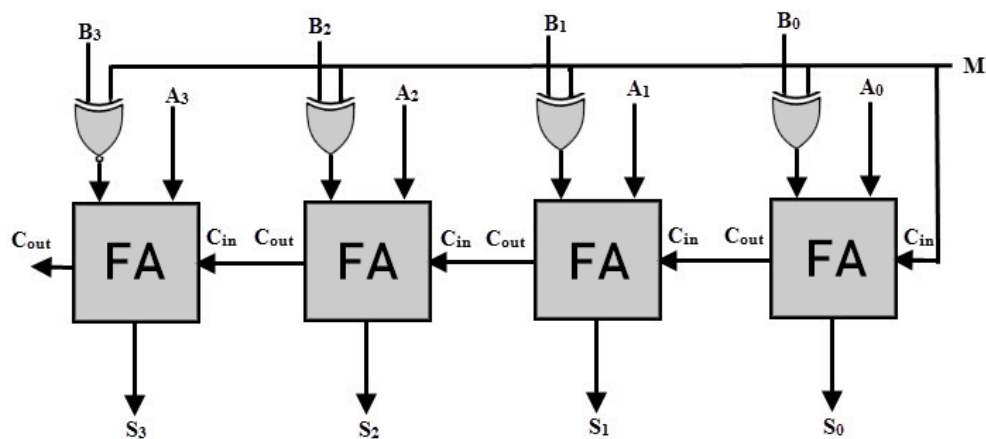
3 Działanie jednostki arytmetyczno-logicznej ALU

Działanie jednostki arytmetyczno-logicznej ALU można zaprezentować różnymi symulacjami czy aplikacjami. Najprostszym przykładem może być układ sumatora/subtraktora lub komparatora, który przedstawi nam działanie samej arytmetyki oraz logiki w jednostce. ALU działa w systemie zero-jedynkowym, dlatego każda z tych symulacji działa na podobnych schematach. Można powiedzieć, że są to systemy stworzone na bramkach logicznych. Oczywiście każdy z tych układów może być zrealizowany na różne sposoby. Istnieją symulacje graficzne oraz typowo tekstowe. Graficzne symulacje pozwalają na pokazanie, jak dokładnie zbudowana jest dana jednostka. Można wtedy zobaczyć na jakich bramkach oraz blokach cyfrowych jest realizowany dany układ. Symulacje tekstowe pozwalają na sprawdzenie wyniku określonego działania w błyskawiczny sposób. Są to po prostu skutki danej operacji, którą aktualnie wykonujemy, przykładem może być operacja dodawania.

3.1 Układ sumatora/subtraktora

Układ sumatora/subtraktora pozwala na dodawanie oraz odejmowanie liczb binarnych. Są to dwie podstawowe operacje, które muszą być wykonywane w każdym cyfrowym komputerze. Instrukcja, która jest wykonywana, związana jest z wartością binarną sygnału sterującego. Jest to linia, odpowiadająca za rodzaj operacji odejmowania lub dodawania. Dlatego aby ten sam układ mógł dodawać oraz odejmować liczby, należy także zastosować bramki Ex-Or do każdego pełnego sumatora. Układ może posiadać N wejść dla liczby A i B oraz N wynikowych wyjść. Aby dodatkowo można było wprowadzić na wejściu liczby ujemne, należy przyjąć odpowiedni system reprezentacji liczb. Najbardziej odpowiednim sposobem zapisu liczb całkowitych w tym przypadku jest kod uzupełnień do dwóch (U2). [26]

Przykład sumatora/subtraktora 4 bitowego:



Rysunek 12: Sumator/Subtraktor - 4 bit

Źródło: <https://www.geeksforgeeks.org/4-bit-binary-adder-subtractor/>

Gdzie:

A0, A1, A2, A3 - Liczba A;

B0, B1, B2, B3 - Liczba B;

M - Linia sterująca, określa operacje, dodawanie lub odejmowanie;

Cout - Bit przeniesienia;

Cin - Bit wejścia;

S0, S1, S2, S3 - Wyjście;

FA - Układ sumatora pełnego;

Pierwszy sumator pełny posiada wejście bezpośrednio powiązane z linią sterującą M, jest to wejście Cin. Najmniej znaczący bit liczby A – A0 jest bezpośrednio wprowadzany do FA. Trzecim wejściem jest najmniej znaczący bit liczby B – B0 który jest dodany z bramką logiczną Ex-Or. Sumator posiada także dwa wyjścia - bit przeniesienia (Cout) oraz bit wynikowy (S0). Gdy M=1, układ odejmuje liczby, a gdy M=0 układ dodaje liczby. Wartość liczby B jest powiązana z linią sterującą dzięki bramce logicznej Ex-Or. Jeśli przyjmujemy wartość 0 na linii sterującej, to każdy bit liczby B stanie niskim posiada na wejściu FA wartość 0. W przeciwnym wypadku, jeśli bit liczby B jest w stanie wysokim, na wejściu FA otrzymamy wartość 1. Przyjmując wartość 1 na linii sterującej odwraca się nam sytuacja, wszystkie wejścia liczby B stanie niskim, posiadają na wejściu FA wartość 1, dodatkowo bit

wejścia Cin pierwszego sumatora pełnego zostaje uaktywniony.

Tablica 6: Tablica prawdy - FA

Ai	Bi	Cin	Cout	Si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3.2 Działanie układu ALU Simulator - EE3221 Digital Systems II

Układ ALU Simulator – EE3221 Digital Systems II jest aplikacją pozwalającą na symulację działania jednostki arytmetyczno logicznej. Program został opracowany przez Prof. Richarda Tervo na wydziale Inżynierii Elektrycznej i Komputerowej w Kanadzie, Uniwersytetu Nowego Brunswiku. Aplikacja pozwala na wykonywanie operacji binarnych oraz logicznych na 16-bitowych wejściach. Wyjście ALU to także 16-bitowy wynik uzupełniony o różne bity (Z, C, N, V) w rejestrze flag. [27]

- Z – Sprawdza czy operacja daje wynik zerowy; 0 – niezerowy, 1 – zerowy.
- C – Sprawdza czy wynik przekroczył zakres 16 bitów, flaga przeniesienia; 0 – brak przekroczenia zakresu, 1 – przekroczenie zakresu
- N – Sprawdza znak wyniku; 0 – nieujemny, 1 – ujemny.
- V – Sprawdza czy powstaje przepełnienie; 0 – brak przepełnienia, 1 – przepełnienie

Użytkownik może wprowadzić dwa operandy RB oraz RA w postaci szesnastu bitów lub czterech cyfr heksadecymalnych. Następnie mamy do wyboru 5 opcji: ADD, SUB, AND, OR, XOR.

ADD – Operacja dodawania, sumator.
 SUB – Operacja odejmowania, subtraktor.
 AND – Operacja na bramce logicznej AND.
 OR – Operacja na bramce logicznej OR.
 XOR – Operacja na bramce logicznej XOR.

Przykład:

Wprowadzamy 16 bitów dla wartości RB oraz RA. Wybieramy opcję która nas interesuje, w tym przypadku jest to dodawanie.

1. Wprowadzamy 16 bitów dla wartości RB oraz RA. Wybieramy opcję która nas interesuje, w tym przypadku jest to dodawanie.

The screenshot shows the ALU Simulator interface. At the top, there are two input fields: 'RB' with the value '1001100110011001' and 'RA' with the value '0101101110011101'. Below these fields are five radio buttons for selecting an operation: 'ADD' (selected), 'SUB', 'AND', 'OR', and 'XOR'. At the bottom center is a 'Compute' button.

Rysunek 13: ALU Simulator - wprowadzanie wartości RB oraz RA

2. Otrzymujemy wynik w postaci 16 bitów oraz czterech cyfr decymalnych. Dodatkowo przy każdej flag pojawia się odpowiednia wartość binarna.

RB	9999	1001100110011001	
RA	5B9D	0101101110011101	FLAGS
	----	-----	V C N Z
RC	F536	1111010100110110	0 0 1 0

Rysunek 14: ALU Simulator - wynik operacji

3. Na koniec otrzymujemy także informacje co oznacza każda flaga.
 - (a) $Z = 0$, ponieważ wynik jest niezerowy;
 - (b) $N = 1$, ponieważ wynik jest ujemny;
 - (c) $C = 0$ (przeniesienie), ponieważ wynik dodawania nie przekroczył 16-bitów, 0 w tym przypadku oznacza, że operacja została zrealizowana w prawidłowy sposób.
 - (d) $V = 0$ (przepełnienie), ponieważ suma liczb zawiera się w przedziale, 0 oznacza, że nie występuje overflow.

Flags

The Zero flag (Z) is **0** because the result is non-zero

The Sign flag (N) is **1** because the result appears negative (the MSB is 1)

The Carry flag (C) is **0** (good) because the addition result did not exceed 16-bits.
This would be of interest if (RA,RB) represent unsigned integers (0..65535).

The Overflow flag (V) is **0** (good) because the sum of positive and negative numbers will always be in range.
This would be of interest if (RA,RB) represent signed integers (-32768..+32767).

Rysunek 15: ALU Simulator - flagi

4 Programy symulujące działanie układów cyfrowych w komputerze

Programy, symulujące działanie układów cyfrowych w komputerze, pomagają zrozumieć, jak działa dana konfiguracja elementów. Dzięki nim, możemy także projektować własne układy cyfrowe oraz symulować ich działanie. W internecie jest wiele dostępnych aplikacji, które służą właśnie do tych celów. Same programy można podzielić na dwa typy – graficzne oraz tekstowe. Programy graficzne zazwyczaj pozwalają na zbudowanie określonej siatki połączeń. Aplikacje tekstowe natomiast, pokazują wynik operacji, które otrzymalibyśmy budując odpowiedni układ.

4.1 Digital Works

Digital Works jest programem graficznym pozwalającym projektować układy cyfrowe i logiczne, z możliwością symulacji ich działania. Do tworzenia obwodów możemy wykorzystać bramki logiczne oraz proste przerzutniki. Dane wejściowe można uzyskać za pomocą generatora sekwencji, wejść interaktywnych, przełączników czy zegara. Dane wyjściowe mogą być wyświetlane za pomocą diod LED, diod 7-segmentowych czy urządzeń numerycznych. Program posiada także możliwość tworzenia makr, czyli bloków, w których znajduje się określony obwód logiczny. Można je dodawać do schematu jako integralną część bardziej złożonego projektu. [16]

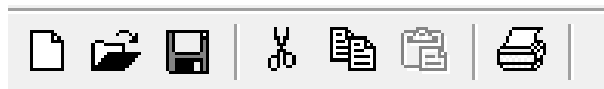
Program posiada 4 panele, w których możemy wybierać odpowiednie opcje.

1. Jest to typowy panel, w którym możemy wykonywać różne funkcje, od tworzenia nowego projektu do usuwania odpowiednich elementów. Możemy także rozpocząć w nim symulowanie programu lub ustawić odpowiednią szybkość zegara.



Rysunek 16: Digital Works - panel wyboru

2. Pozwala na szybki dostęp do najważniejszych funkcji takich jak: zapisywanie, kopiowanie, drukowanie.



Rysunek 17: Digital Works - panel szybkiego wyboru

3. Znajdują się w nim elementy układów elektronicznych. Dzięki nim możemy tworzyć projekt. Wybieramy je z panelu po czym umieszczamy w pustym polu programu.



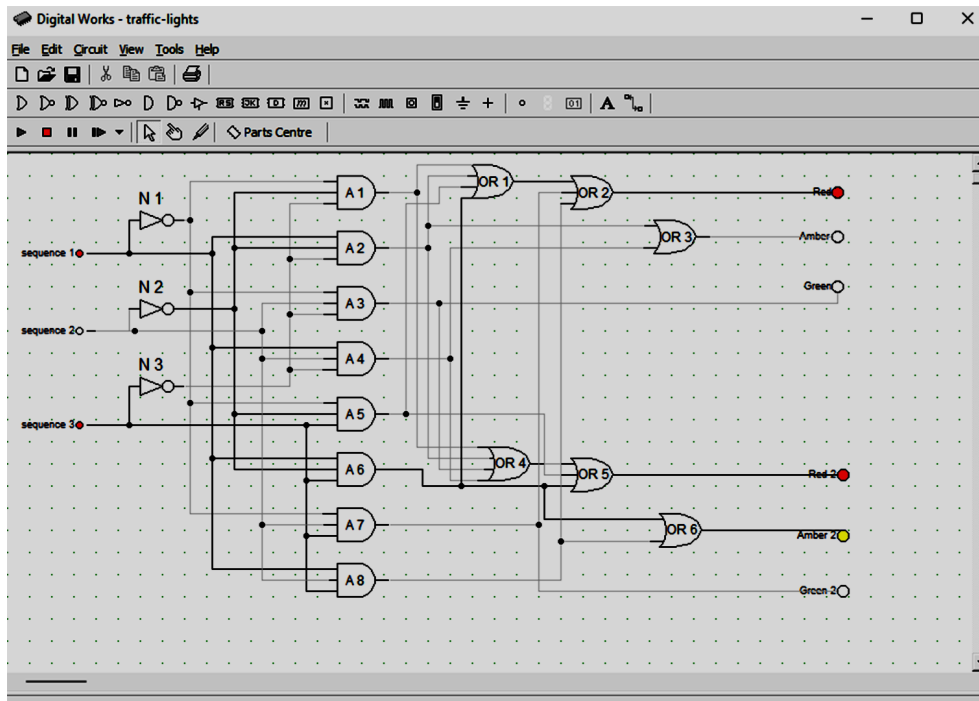
Rysunek 18: Digital Works - panel funkcyjny

4. Najważniejsze skróty, które mogą się przydać użytkownikowi, takie jak np. start, stop symulacji.



Rysunek 19: Digital Works - panel symulacji

Przykład projektu: sygnalizacja świetlna



Rysunek 20: Digital Works

4.2 Cedar logic Simulator

Cedar Logic simulator został zaprojektowany przez profesorów Uniwersytetu Cedarville. Jest aplikacją graficzną pozwalającą na projektowanie układów logicznych oraz cyfrowych. Schematy elektroniczne w programie możemy budować za pomocą bramek logicznych, rejestrów, emulatorów, itp. Sama aplikacja dodatkowo pozwala na testowanie zbudowanych układów poprzez symulację. [17]

Program posiada 4 panele w których możemy wybierać odpowiednie opcje.

1. Jest to typowy panel, w którym posiadamy wiele opcji. Dzięki niemu możemy otwierać nowe projekty lub zapisywać ich stan w odpowiednim miejscu na dysku. Pozwala także na drukowanie oraz kopiowanie schematu. Dodatkowo opcja "help" pozwala na zapoznanie się z komponentami programu.

File Edit View Help

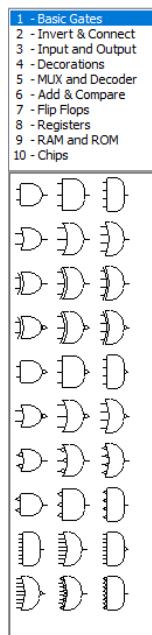
Rysunek 21: Cedar - panel wyboru

2. Jest panelem szybkiego wyboru, posiada opcje takie jak: zapisywanie projektu, cofanie do poprzedniego stanu, kopiowanie, start oraz stop symulacji czy przybliżanie panelu projektowego.



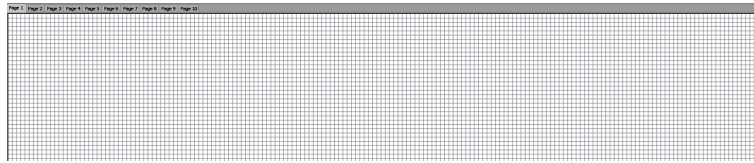
Rysunek 22: Cedar - panel szybkiego wyboru

3. Wybiermy w nim odpowiednie komponenty które chcemy dodać do naszego schematu.



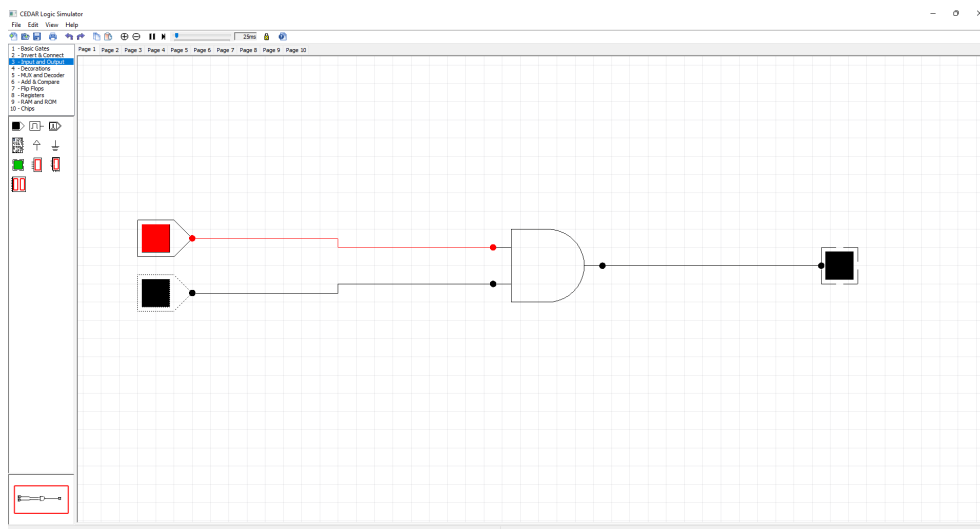
Rysunek 23: Cedar - panel wyboru komponentów

4. Tworzymy w nim układ elektroniczny przenosząc na pole odpowiednie komponenty. Panel posiada także możliwość przechodzenia na inną stronę, aby można było wykonywać kilka projektów równocześnie.



Rysunek 24: Cedar - panel do tworzenia układów elektronicznych

Przykład projektu: Bramka AND

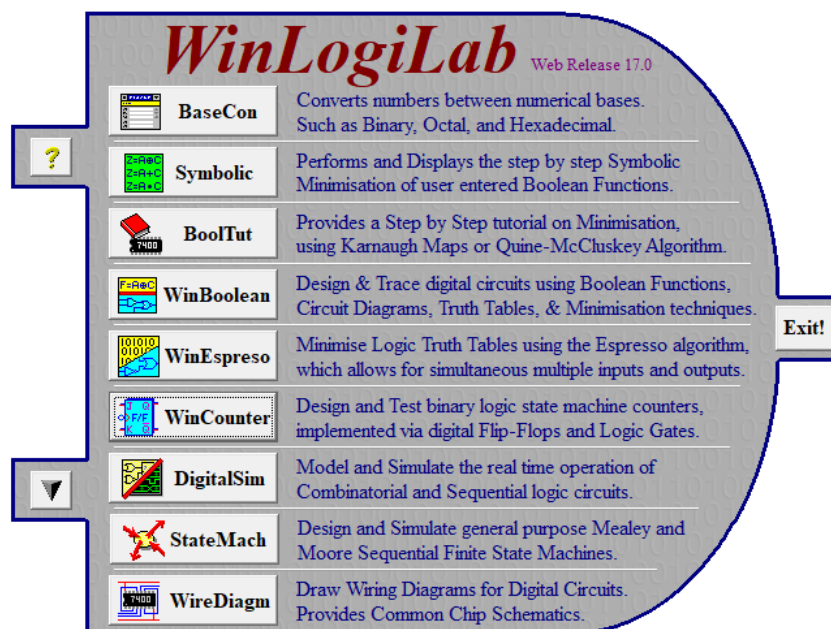


Rysunek 25: Cedar Logic simulator

4.3 Win Logi Lab

WinLogiLab jest interaktywnym pakietem dydaktycznym, przeznaczonym do projektowania kombinacyjnych oraz sekwencyjnych układów logicznych. Aplikacja została zaprojektowana przez Charles'a Hacker'a - nauczyciela akademickiego. WinLogiLab składa się z samouczków, symulacji oraz podstawowych narzędzi wspomagających projektowanie, jak i analizowanie układów elektornicznych. [18]

Program posiada 9 podprogramów, w których możemy wykonywać różne opcje.



Rysunek 26: WinLogiLab

1. BaseCon - konwertowanie liczb na różne systemy numeryczne.
2. Symbolic - symulacja krok po kroku minimalizacji funkcji boolowskich wprowadzonych przez użytkownika.
3. BoolTut - symulacja krok po kroku minimalizacji tablicy prawdy za pomocą mapy Karnaugh lub algorytmu Quine-McCluskey.

4. WinBoolean - projektowanie oraz symulacja stworzonych obwodów elektronicznych, tablicy prawdy oraz minimalizacji z wykorzystaniem funkcji boolowskich.
5. WinEspresso - minimalizowanie tablicy prawdy za pomocą algorytmu Espresso.
6. WinCounter - projektowanie i testowanie maszyn liczących, zaprojektowanych przy pomocy bramek logicznych oraz przerzutników.
7. DigitalSim - projektowanie układu elektronicznego z możliwością symulacji.
8. StateMach - projektowanie oraz symulacja automatów skończonych.
9. WireDiagm - rysowanie diagramów układów cyfrowych.

Przykład podprogramu BaseCon: konwersja liczby dziesiętnej na inne systemy liczbowe.

Base Conversion

File Edit Options Help

Convert : 10 From Base 10
Decimal

To :

Binary 1010 of Base 2

Octal 12 of Base 8

Decimal 10 of Base 10

Hexadecimal A of Base 16

Other 22 of Base 4

BINARY CODED DECIMAL

BCD 0001 0000

BCD Ex-3 (Excess-3) 0100 0011

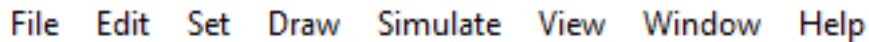
Rysunek 27: BaseCon

4.4 Multimedia Logic

MultimediaLogic jest aplikacją pozwalającą na rysowanie, modyfikowanie oraz symulację obwodów logicznych. Program został zaprojektowany przez Petera Robinson'a, Profesora Uniwersytetu w Cambridge na wydziale technologii komputerowych. [19]

Program posiada 4 panele w których możemy wybierać odpowiednie opcje.

1. Typowy panel, w którym posiadamy wiele opcji. Dzięki niemu możemy otwierać nowe projekty lub zapisywać ich stan w odpowiednim miejscu na dysku. Pozwala także na drukowanie schematu, rozpoczęcie symulacji czy wybranie odpowiedniego komponentu.



File Edit Set Draw Simulate View Window Help

Rysunek 28: MultimediaLogic - panel wyboru

2. Panel szybkiego wyboru, posiada opcje takie jak: zapisywanie projektu, tworzenie nowego projektu, start oraz stop symulacji czy przybliżanie panelu projektowego



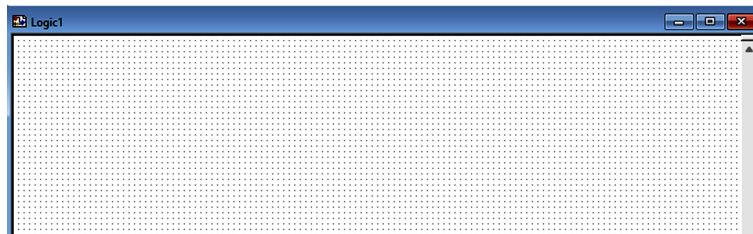
Rysunek 29: MultimediaLogic - panel szybkiego wyboru

3. Paleta różnych najpotrzebniejszych komponentów oraz funkcji.



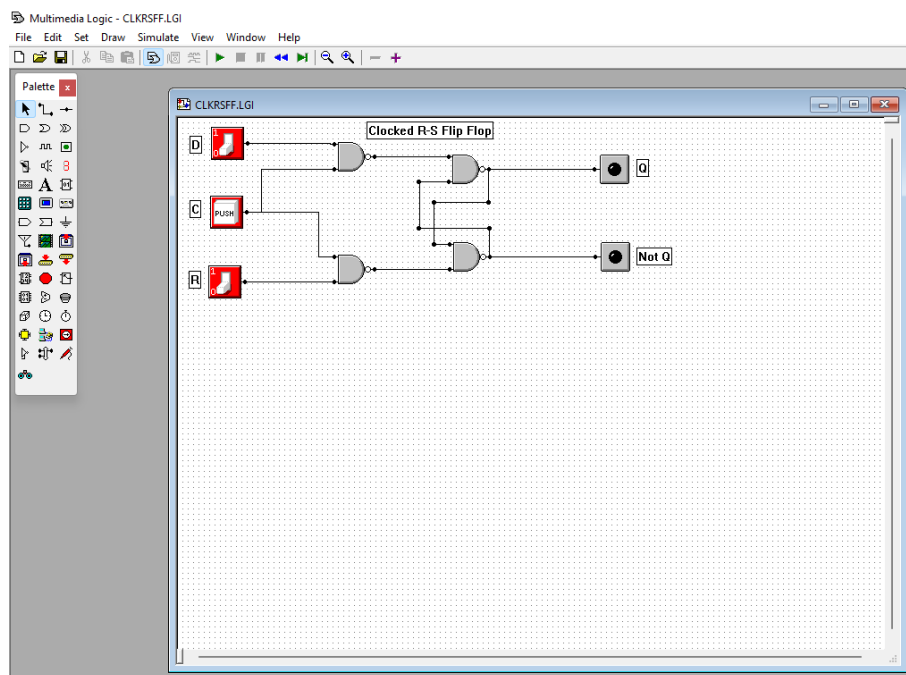
Rysunek 30: MultimediaLogic - paleta różnych funkcji

4. Panel projektowy, w którym tworzymy układy elektroniczne, przenosząc odpowiednie komponenty na wyznaczone do tego pole.



Rysunek 31: MultimediaLogic - panel do tworzenia układów elektronicznych

Przykład projektu: Przerzutnik RS



Rysunek 32: MultimediaLogic - Program

4.5 Logisim

Logisim jest programem pozwalającym na projektowanie oraz symulowanie cyfrowych obwodów logicznych. Aplikacja wykorzystywana jest głównie przez studentów w celach edukacyjnych. [20]

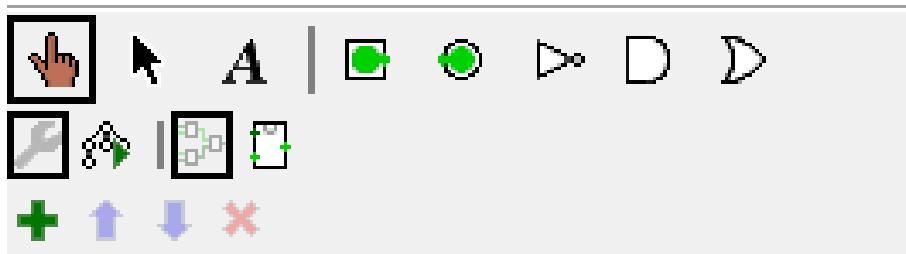
Program posiada 4 panele, w których możemy wybierać odpowiednie opcje.

1. Typowy panel, w którym posiadamy wiele opcji, dzięki niemu możemy otwierać nowe projekty lub zapisywać ich stan w odpowiednim miejscu na dysku. Pozwala także na drukowanie schematu, rozpoczęcie symulacji lub dodanie odpowiedniego komponentu na panel projektowy. Dodatkowo dzięki opcji "help" możemy otworzyć samouczek programu.

File Edit Project Simulate Window Help

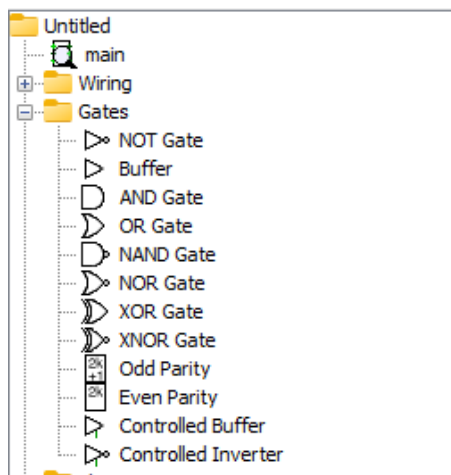
Rysunek 33: Logisim - panel wyboru

2. Panel szybkiego wyboru, posiada najpotrzebniejsze funkcje oraz komponenty potrzebne do budowy schematu elektronicznego.



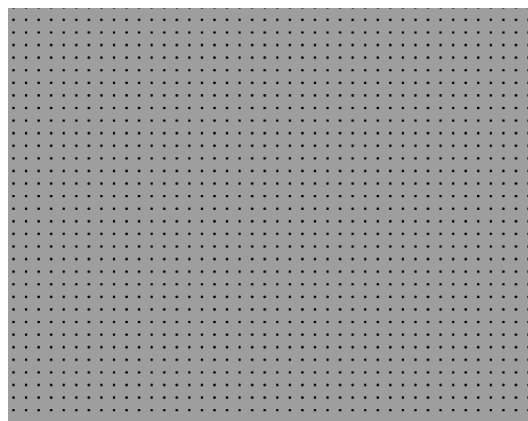
Rysunek 34: Logisim - panel szybkiego wyboru

3. Wybieramy w nim odpowiednie komponenty do naszego projektu, które chcemy przenieść na panel projektowy. Znajdują się w nim moduły takie jak: bramki logiczne, urządzenia wejścia-wyjścia czy proste układy kombinacyjne.



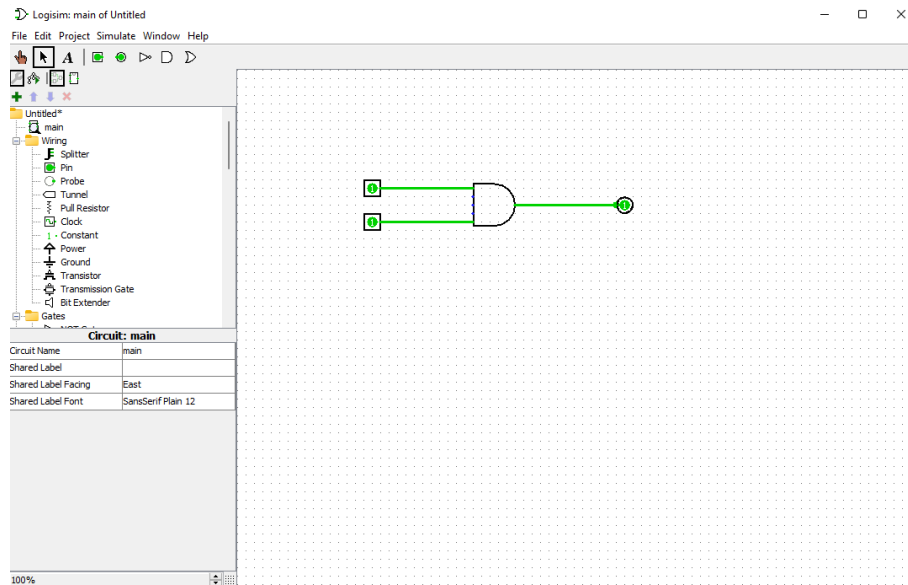
Rysunek 35: Logisim - paleta różnych funkcji

4. Panel projektowy, w którym tworzymy układy elektroniczne, przenosząc odpowiednie komponenty na wyznaczone do tego pole.



Rysunek 36: Logisim - panel do tworzenia układów elektronicznych

Przykład projektu: Bramka AND



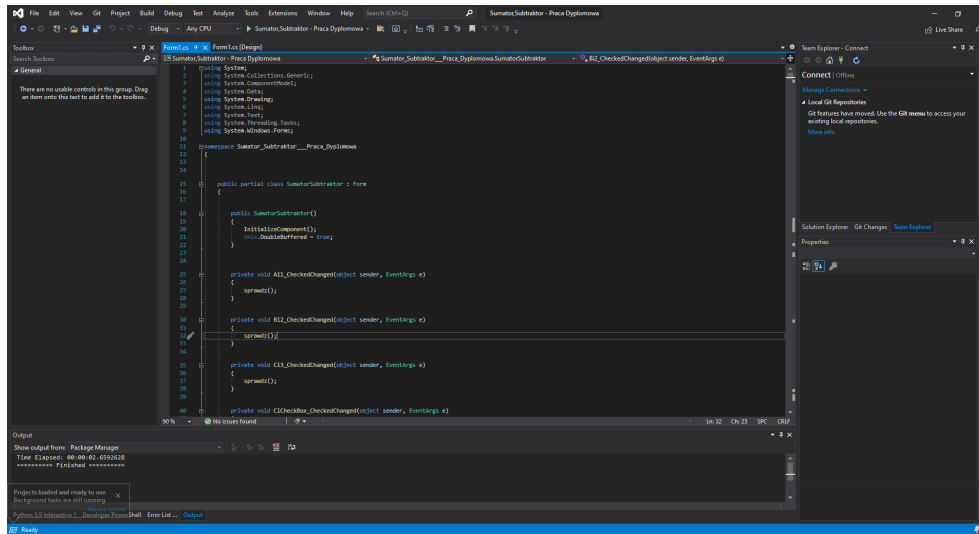
Rysunek 37: Logisim - Program

5 Technologie informatyczne wykorzystywane przy budowie aplikacji symulujących działanie układów cyfrowych

Technologie informatyczne wykorzystywane przy budowie aplikacji symulujących działanie układów cyfrowych mogą być różne. Zazwyczaj do stworzenia takich programów wykorzystuje się języki programowania oraz edytory graficzne. Symulacje można zobrazować za pomocą graficznej reprezentacji oraz tekstowej, dlatego niemal każdy język programowania będzie do tego odpowiedni. Najczęściej takie projekty są aplikacjami desktopowymi, czyli programami, które bezpośrednio instalujemy na urządzeniu. Oczywiście istnieją także aplikacje webowe, ale głównie dominują w tej dziedzinie języki wysokiego poziomu. Programy, które opisywałem w rozdziale czwartym, to głównie aplikacje oparte na języku C++ oraz C#. Ja osobiście swoją aplikację zbudowałem na bazie języka C# i edytora graficznego Adobe Photoshop.

5.1 C#

C# jest językiem programowania ogólnego przeznaczenia. Został opracowany pod szyldem firmy Microsoft przez Anders 'a Hejlsberg'a w 2000 roku. Język najczęściej używany jest w technologii Windows .NET, ale także świetnie sprawuje się na platformach open source. Zbudowany został na wzór języka C++ jako język obiektowy. C# może być wykorzystywany w różnych celach i zadaniach. Język pozwala na tworzenie aplikacji desktopowych, aplikacji webowych, stron internetowych czy nawet gier. Najbardziej odpowiednim zintegrowanym środowiskiem graficznym jest Visual Studio, który także został stworzony przez firmę Microsoft. [28]



Rysunek 38: Okno programu Visual Studio

5.2 Adobe Photoshop

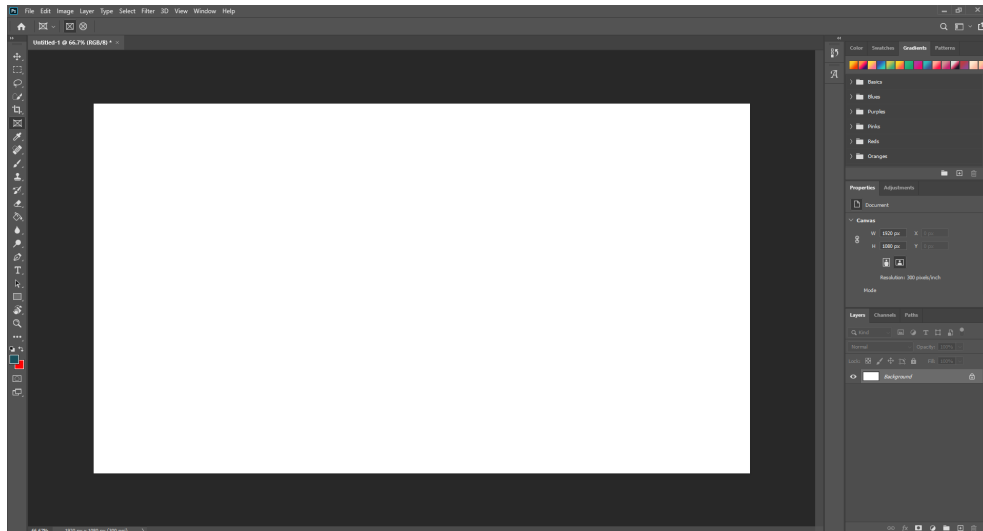
Adobe Photoshop jest programem graficznym służącym do edycji obrazów rastrowych. Został zaprojektowany przez Thomasa oraz Johna Knolla w 1988 roku. Aplikacja jest kompatybilna z dwoma najpopularniejszymi systemami: Windows oraz MacOS. Niestety firma Adobe nie udzieliła wsparcia systemowego dla oprogramowania Linux. [29]

Program posiada bardzo wiele narzędzi, które pozwalają wykonać niemal każdą przydatną operację do zrealizowania określonego celu w zakresie edycji obrazów. Photoshop wykorzystuje warstwowość, dzięki czemu jest możliwe proste manipulowanie. Warstwy mogą działać jako maski czy też filtry. Możliwe jest także stosowanie na nakładki efektów oraz kolorów np. CMYK, RGB.

Przykładowe narzędzia Photoshop:

1. Lasso tool - odręczne obrysowanie określonego obszaru.
2. Crop tool - przycina lub rozszerza krawędzie zdjęcia.
3. Brush tool - malowanie niestandardowych pociągnięć pędzlem.
4. Blur tool - bluruje wyznaczony obszar na zdjęciu.

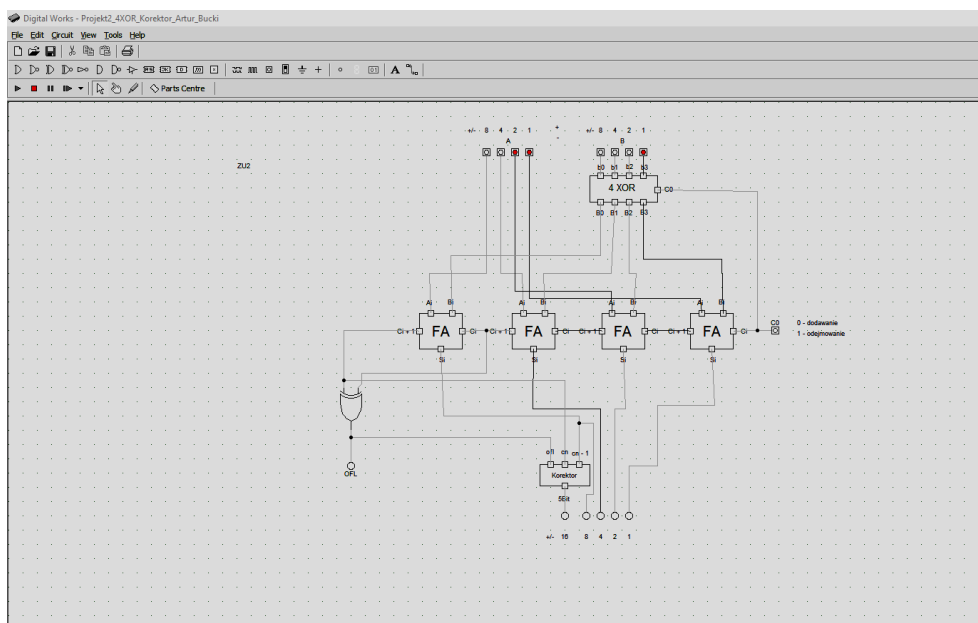
5. Rectangle tool - rysuje prostokąty.



Rysunek 39: Photoshop

6 Projekt i realizacja aplikacji symulującej działanie sumatora/subtraktora (Add Sub)

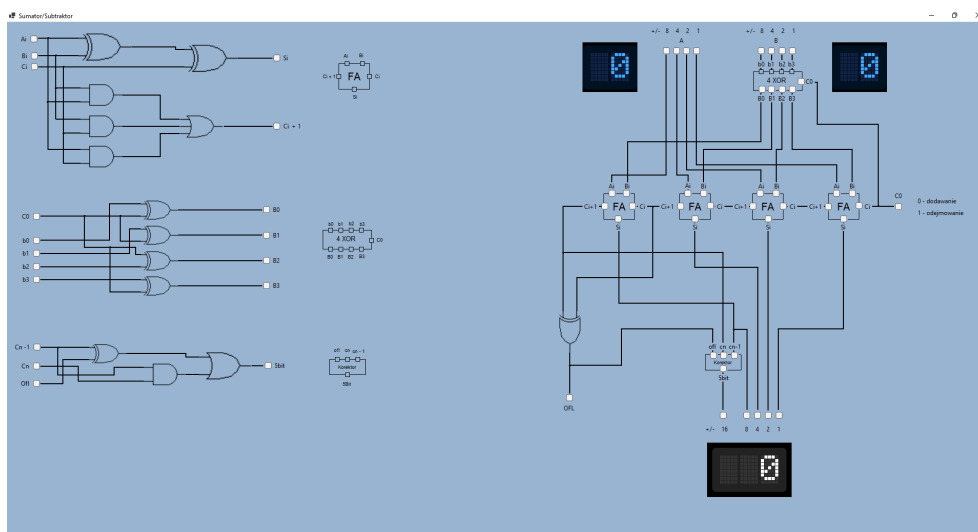
Aplikacja symulująca działanie sumatora/subtraktora została zrealizowana przy pomocy technologii C# [5.1] oraz edytora graficznego Adobe Photoshop [5.2]. Realizacja projektu została rozpoczęta od stworzenia schematu w programie Digital Works [4.1]. Na początku należało wybrać odpowiedni system reprezentacji liczb całkowitych, który pozwolił na zapis ujemnych liczb. Najbardziej odpowiednim do tego konceptu okazał się zapis uzupełnień do dwóch (U2). Projekt został zrealizowany na 4 bitowych wejściowych liczb A oraz B. Aby zrealizować sumator 4 bitowy należało połączyć odpowiedni sposób cztery sumatory pełne. Do tego zadania zostały wykorzystane makra, które pozwalają na eliminację powtarzających się układów. Dodatkowo, aby użytkownik mógł wykorzystać w pełni potencjał programu został dodany dodatkowy bit wyjściowy, który pokazuje poprawnie liczby, nawet w przypadku sytuacji, w której występuje overflow. Aby odwzorować jednak faktyczne działanie prawdziwego sumatora istnieje dioda LED informująca o przepełnieniu. Aby układ mógł także odejmować liczby, zostało stworzone makro z czterech bramek XOR, do zamiany liczby B na ujemną.



Rysunek 40: Aplikacja - Digital Works

Projekt stworzony w Digital Works został "przeniesiony" do języka pro-

gramowania C#. Logika jak i sama symulacja została odwzorowana za pomocą przycisków wyboru, grafik oraz pióra. Aby użytkownik mógł dokładniej zrozumieć sposób działania układu, z lewej strony widnieją schematy makr wykorzystanych w projekcie. Układy logiczne bloków także są interaktywne co pozwala na dogłębne przeanalizowanie określonej operacji. Po prawej stronie został zbudowany sumator/subtraktor z dodatkowymi wyświetlaczami pokazującymi wynik dziesiętny liczb A, B oraz wyniku.



Rysunek 41: Aplikacja sumatora/subtraktora w programie C#

6.1 Wymagania funkcjonalne i нефunkcjonalne aplikacji

Celem projektu jest stworzenie graficznej aplikacji pozwalającej na symulację działania sumatora i subtraktora w czasie rzeczywistym. Aby zrealizować poprawnie aplikację zostały opracowane założenia, które powinny zostać spełnione. Do wymagań funkcjonalnych należą podstawowe udogodnienia, które powinien oferować system. Wymagania нефunkcjonalne za to są ograniczeniami jakościowymi.

Tablica 7: Wymagania funkcjonalne aplikacji

Wymagania funkcjonalne
1. Panel pokazujący działanie makr wykorzystanych w sumatorze/subtraktorze: 4XOR, FA, Korektor.
2. Panel sumatora/subtraktora z wejściem N oraz wyjściem $N + 1$.
3. Zegary zmieniające liczbę binarną z zapisu U2 na system dziesiętny.

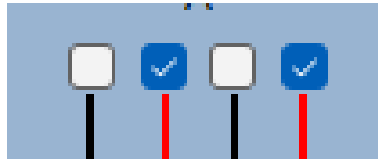
Tablica 8: Wymagania нефункционалне aplikacji

Wymagania нефункционалне
1. Aplikacja powinna działać w czasie rzeczywistym.
2. Przetwarzanie każdego rzędania powinno odbywać się natychmiastowo.
3. Aplikacja powinna umożliwiać pracę na dwóch panelach równolegle bez ich konfliktów: panel sumatora/subtraktora, panel makr.

6.2 Budowa modułowa aplikacji (Add Sub)

Aplikację można podzielić na dwie części. Gdy program zostanie uruchomiony po lewej stronie widnieją układy cyfrowe makr wykorzystanych w sumatorze/subtraktorze. Jest to odłam, który pozwala na przeanalizowanie co znajduje się w określonych blokach. Z prawej strony zbudowana została symulacja sumatora i subtraktora. Obie części jednak składają się z identycznych modułów. Do stworzenia takiej aplikacji wykorzystano funkcje oraz narzędzia z języka programowania C#. Przyrządy, które zostały wykorzystane to: CheckBox, PictureBox, TextBox oraz funkcja Pen

1. CheckBox – Realizuje logikę całej aplikacji. Pozwala użytkownikowi wybierać odpowiednie kombinacje odnośnie liczb wejściowych. Dzięki niemu możemy także zmienić sumator na subtraktor. Pokazuje także wynik wyjściowy oraz overflow.



Rysunek 42: CheckBox

Przykład użycia w kodzie programu:

```
if (Ci3.Checked == false && Bi2.Checked == false && Ai1.Checked == false) // 0
{
    Si1.Checked = false;
    Ci.Checked = false;
}
else if (Ci3.Checked == false && Bi2.Checked == false && Ai1.Checked == true) // 1
{
    Si1.Checked = true;
    Ci.Checked = false;
}
```

Rysunek 43: CheckBox - użycie w kodzie

Gdzie:

Ci3, Bi2, Ai1, Si1, Ci - odpowiednie bity sumatora pełnego;

Checked - metoda sprawdzająca czy CheckBox jest zaznaczony.

2. PictureBox - służy do wyświetlania grafik. W programie, dzięki tej opcji, zostały dodane bramki logiczne, makra oraz zegary zmieniające kod U2 na dziesiętny.



Rysunek 44: PictureBox

Przykład użycia w kodzie programu:

```

if (wynik == 0)
{
    pictureBox24.Image = Properties.Resources.END0;
}
if (wynik == 1)
{
    pictureBox24.Image = Properties.Resources.END1;
}
if (wynik == 2)
{
    pictureBox24.Image = Properties.Resources.END2;
}
if (wynik == 3)
{
    pictureBox24.Image = Properties.Resources.END3;
}

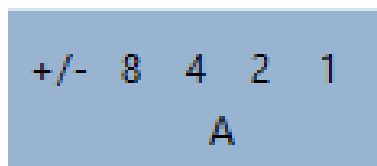
```

Rysunek 45: PictureBox - użycie w kodzie

Gdzie:

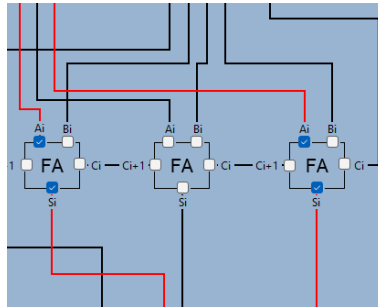
Wynik - określona wartość wyjścia zmieniona na system dziesiętny;
 pictureBox.Image - metoda odpowiadająca za podmienianie obrazka
 przy zmianach wartości w programie.

3. TextBox - Wyświetla proste teksty przy różnych elementach w programie.



Rysunek 46: TextBox

4. Pen – jest funkcją pozwalającą na rysowanie linii. W aplikacji służy jako reprezentacja przewodów łączących różne elementy.



Rysunek 47: Pen

Przykład użycia w kodzie programu:

```
Color ColorOn = Color.Red;
Color ColorOff = Color.Black;

Pen Line1 = new Pen(Color.Black);
if (Ai1.Checked == true)
{
    Pen(e, Line1, ColorOn, 60, 32, 150, 32);
}
else if (Ai1.Checked == false)
{
    Pen(e, Line1, ColorOff, 60, 32, 150, 32);
}
```

Rysunek 48: Pen - użycie w kodzie

Gdzie:

Color - funkcja zmieniająca kolor linii;

Pen - funkcja tworząca pióro z określonymi atrybutami;

6.3 Testowanie aplikacji

Aby w pełni przetestować wszystkie możliwe kombinacje w sumatorze/subtraktorze 4 bitowym należałoby wykonać 1024 operacje. Przy tak dużej ilości zestawień jest to niemal niemożliwe do zrealizowania bez pomyłki

użytkownika. Dlatego przy testowaniu aplikacji zostały wyszczególnione najbardziej wrażliwe fuzje.

Mając do wykorzystania 4 bitową liczbę **A** oraz **B** najbardziej podatnym na błędy będzie ich kombinacja odnośnie dodawania oraz odejmowania. Większość operacji, które są wykonywane często mają ten sam schemat działania, ale inne wielkości liczb. Dlatego aby móc sprawdzić różne rodzaje operacji arytmetycznych zostało wyróżnione 8 przypadków.

Tablica 9: Wartości oczekiwane

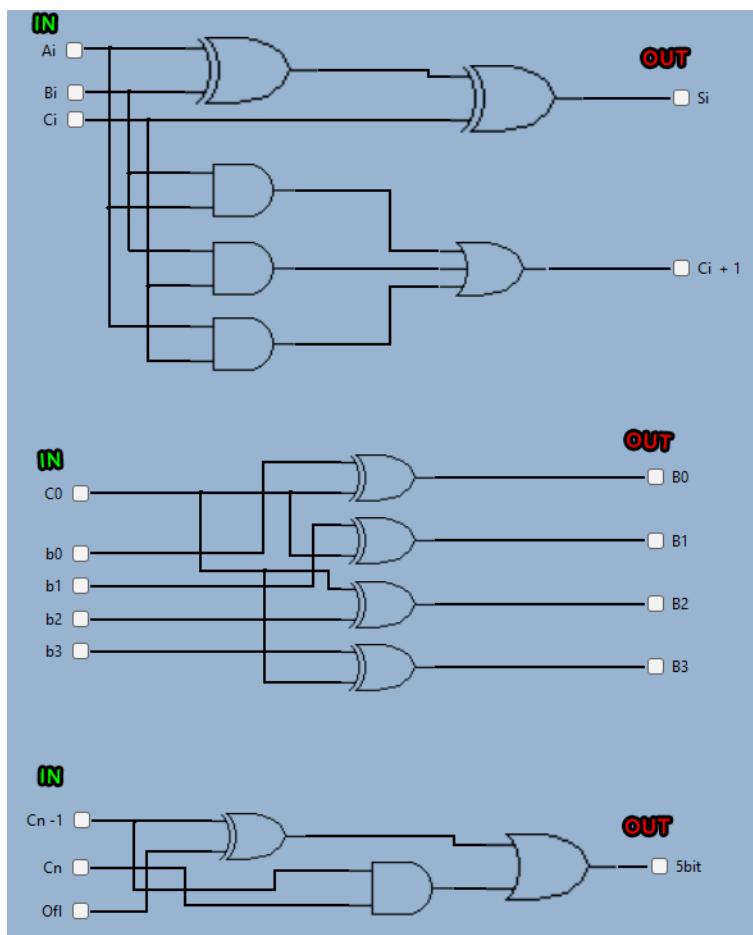
Operacja	Wartości liczb: A, B		
	A = 1; B = 3	A = -3; B = 7	A = 6; B = -8
A + B	4	4	-2
A + (-B)	-2	-10	14
-A + B	2	10	-14
-A + (-B)	-4	-4	2
A - B	-2	-10	14
A - (-B)	4	4	-2
-A - B	-4	-4	2
-A - (-B)	2	10	-14

Tablica 10: Wartości otrzymane w programie

Operacja	Wartości liczb: A, B		
	A = 1; B = 3	A = -3; B = 7	A = 6; B = -8
A + B	4	4	-2
A + (-B)	-2	-10	14
-A + B	2	10	-14
-A + (-B)	-4	-4	2
A - B	-2	-10	14
A - (-B)	4	4	-2
-A - B	-4	-4	2
-A - (-B)	2	10	-14

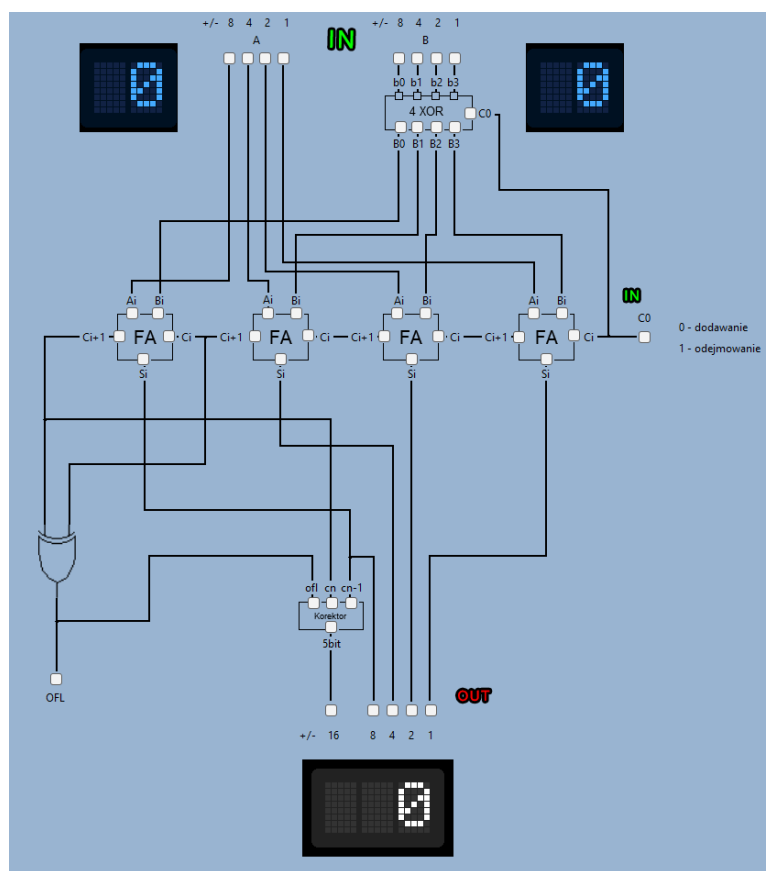
7 Podręcznik użytkownika aplikacji

Użytkownik w programie jest osobą decyzyjną. Po lewej stronie widnieją układy elektroniczne makr wykorzystanych w układzie sumatora/subtraktora po prawej stronie. Można manipulować określonymi kombinacjami za pomocą pól wyboru po lewej stronie (IN).



Rysunek 49: Użytkowanie aplikacji - makra

Identycznie działa sama symulacja subtraktora/sumatora. Na samej górze umieszczone są pola wyboru które możemy odpowiednio zaznaczać. Dodatkowo po prawej stronie widnieje bit odpowiedzialny za odejmowanie lub dodawanie.



Rysunek 50: Użytkowanie aplikacji - sumator/subtraktor

8 Podsumowanie

Procesor w dzisiejszym świecie jest stosowany w prawie każdym bardziej rozbudowanym urządzeniu elektronicznym. Sumator jest częścią jednostki ALU, a ALU jest częścią procesora. Układ kombinacyjny dodawania dwóch liczb dwójkowych posiada wiele ważnych funkcji, bez których niektóre urządzenia elektroniczne nie miałyby racji bytu.

Komputer stacjonarny, w sposób szczególny korzysta z sumatorów, ponieważ jest maszyną liczącą. Przykładem wykorzystania tego elementu w innym prostszym urządzeniu może być kalkulator, który także posiada jednostkę arytmetyczno logiczną i układy cyfrowe składające się z sumatorów.

Celem niniejszej pracy było stworzenie graficznej aplikacji symulującej działanie układu sumatora/subtraktora. Aplikacja została zrealizowana tak, aby użytkownik w pełni miał kontrolę nad programem zgodnie z założeniami architektury von Neumanna. Sama aplikacja pozwala prześledzić działanie dodawania lub odejmowania liczb binarnych na 4 bitach. Co więcej, aby użytkownik mógł bardziej zrozumieć samą ideę programu, wprowadzono piąty bit wyjściowy pokazujący “prawidłowy” wynik, nawet w sytuacji, w której występuje overflow. Zważywszy na przeprowadzone rozważania, zasadne jest stwierdzenie, że cel pracy został osiągnięty.

9 Bibliografia

- [1] Skorupski, A., 2000. Podstawy budowy i działania komputerów. Wydawnictwa Komunikacji i Łączności.
- [2] https://pl.wikipedia.org/wiki/John_von_Neumann
- [3] <https://budowakomputera.manifo.com/model-von-neumanna>
- [4] <https://www.bbc.co.uk/bitesize/guides/zhppfcw/revision/3>
- [5] <http://zsedabrowa.edu.pl/wp-content/uploads/2016/10/logiczny-model-komputera.pdf>
- [6] <http://www.korepetycjenowysacz.edu.pl/operacje-arytmetyczne-systemie-binarnym/>
- [7] <https://www.math.edu.pl/system-pozycyjny>
- [8] <https://botland.com.pl/blog/bramki-logiczne-jak-to-dziala/>
- [9] <https://whatis.techtarget.com/definition/processor>
- [10] <https://www.digitaltrends.com/computing/what-is-a-cpu/>
- [11] <https://www.computerhope.com/jargon/c/contunit.htm>
- [12] <https://www.geeksforgeeks.org/introduction-of-control-unit-and-its-design/>
- [13] <https://www.techtarget.com/whatis/definition/arithmetic-logic-unit-ALU/>
- [14] <https://pl.wikipedia.org/wiki/Subtraktor>
- [15] <http://pec.fizyka.amu.edu.pl/download/sumsub.pdf>
- [16] <https://digital-works.software.informer.com/3.0/>
- [17] <https://sourceforge.net/projects/cedarlogic/>
- [18] <http://www.hakasoft.com.au/winlogilab>
- [19] <https://sourceforge.net/projects/multimedialogic/>
- [20] <http://www.cburch.com/logisim/>

- [21] https://encyklopedia.biolog.pl/index.php?haslo=Rejestr_procesora
- [22] https://pl.wikipedia.org/wiki/Rejestr_procesora
- [23] https://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm
- [24] <https://www.geeksforgeeks.org/input-and-output-devices/>
- [25] https://pl.wikipedia.org/wiki/Magistrala_komunikacyjna
- [26] <https://www.geeksforgeeks.org/4-bit-binary-adder-subtractor/>
- [27] <http://www.ee.unb.ca/cgi-bin/tervo/alu.pl>
- [28] <https://docs.microsoft.com/en-us/dotnet/csharp/>
- [29] https://en.wikipedia.org/wiki/Adobe_Photoshop

10 Spis tablic/rysunków

Spis tablic

1	Wartości dodatnie dla 4 bitów - U2	7
2	Wartości ujemne dla 4 bitów - U2	7
3	Schemat - pozycyjne systemy liczbowe	8
4	Tabela prawdy - bramki logiczne	12
5	Najczęściej wykorzystywane rejestry	16
6	Tablica prawdy - FA	23
7	Wymagania funkcjonalne aplikacji	44
8	Wymagania нефункционалне aplikacji	44
9	Wartości oczekiwane	48
10	Wartości otrzymane w programie	48

Spis rysunków

1	Schemat dodawania liczb binarnych	6
2	Przykład dodawania liczb binarnych	6
3	Dodawanie liczb U2 - overflow	8
4	Bramka logiczna - NOT	10
5	Bramka logiczna - AND	10
6	Bramka logiczna - NAND	11
7	Bramka logiczna - OR	11
8	Bramka logiczna - NOR	11
9	Bramka logiczna - XOR	12
10	Typowy symbol - ALU	14
11	Magistrala systemowa	19
12	Sumator/Subtraktor - 4 bit	22
13	ALU Simulator - wprowadzanie wartości RB oraz RA	24
14	ALU Simulator - wynik operacji	24
15	ALU Simulator - flagi	25
16	Digital Works - panel wyboru	26
17	Digital Works - panel szybkiego wyboru	27
18	Digital Works - panel funkcyjny	27
19	Digital Works - panel symulacji	27
20	Digital Works	28
21	Cedar - panel wyboru	29
22	Cedar - panel szybkiego wyboru	29

23	Cedar - panel wyboru komponentów	29
24	Cedar - panel do tworzenia układów elektronicznych	30
25	Cedar Logic simulator	30
26	WinLogiLab	31
27	BaseCon	32
28	MultimediaLogic - panel wyboru	33
29	MultimediaLogic - panel szybkiego wyboru	33
30	MultimediaLogic - paleta różnych funkcji	34
31	MultimediaLogic - panel do tworzenia układów elektronicznych	35
32	MultimediaLogic - Program	35
33	Logisim - panel wyboru	36
34	Logisim - panel szybkiego wyboru	36
35	Logisim - paleta różnych funkcji	37
36	Logisim - panel do tworzenia układów elektronicznych	37
37	Logisim - Program	38
38	Okno programu Visual Studio	40
39	Photoshop	41
40	Aplikacja - Digital Works	42
41	Aplikacja sumatora/subtraktora w programie C#	43
42	CheckBox	45
43	CheckBox - użycie w kodzie	45
44	PictureBox	45
45	PictureBox - użycie w kodzie	46
46	TextBox	46
47	Pen	47
48	Pen - użycie w kodzie	47
49	Użytkowanie aplikacji - makra	49
50	Użytkowanie aplikacji - sumator/subtraktor	50