КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ВИСОКИХ ТЕХНОЛОГІЙ

В.о. завідувача кафедри квантової радіофізики
доц. Сергій Олександрович КОЛЄНОВ
Протокол №____ засідання кафедри
від “____” _____20___ р.

РОЗРОБКА ПРИСТРОЮ ДЛЯ ГАЛЬВАНІЧНОГО НАРОЩУВАННЯ ТОНКИХ МЕТАЛЕВИХ ПЛІВОК НА ОСНОВІ ПЛІС MAX10

Випускна кваліфікаційна робота бакалавра
студентки спеціальності
171 Електроніка
ОП «Електроніка (високі технології)»
Юрченко Дарини Олегівни

Науковий керівник
доцент кафедри
квантової радіофізики
к.ф.-м.н. **Колєнов Сергій Олександрович**

Оцінка захисту роботи
_____

Київ 2024

АНОТАЦІЯ

Юрченко Д. О. Розробка пристрою для гальванічного нарощування тонких металевих плівок на основі ПЛІС MAX10. – Випускна кваліфікаційна робота бакалавра за спеціальністю 171 Електроніка ОП «Електроніка (високі технології)».

У роботі розглянуто методи та особливості гальванічного нарощування тонких металевих плівок та розроблено прилад для контролю параметрів та автоматизації даного процесу. Розроблений прилад може бути використаний у виробництві електронних компонентів з тонкими плівками нікелю, міді, срібла або золота, або використаний задля дослідження впливу параметрів нарощування на структуру плівок та їх електрофізичні властивості.

Ключові слова: гальваностегія; імпульсне реверсивне покриття; тонкі металеві плівки; ПЛІС; електроніка; цифрова електроніка; аналогова електроніка; автоматизація.

ABSTRACT

In this paper the methods and features of galvanic deposition of thin metal films are analyzed, and a device for controlling the parameters and automating this process is developed. The developed device can be used in the production of electronic components with thin nickel, copper, silver or gold films, or used to study the effect of the plating parameters on the structure of the films and their electrophysical properties.

Keywords: electroplating; pulse reverse plating; thin metal films; FPGA; electrical engineering, digital electronics; analog electronics; automation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСМ – атомно-силовий мікроскоп

ІС – інтегральна схема

ПЛІС – програмована логічна інтегральна схема

СЕМ – сканувальний електронний мікроскоп

ТМП – тонкі металеві плівки

ШІМ – широтно-імпульсна модуляція

OLED – органічний світлодіод

PP (pulse plating) – імпульсне покриття

PRP (pulse reverse plating) – імпульсне реверсивне покриття

SPDT – single pole double throw

Вступ

Гальваностегія є однією з ключових технологій у багатьох сучасних галузях, таких як електроніка, оптика, нанотехнології, матеріалознавство та інших. Вона дозволяє нарощувати високоякісні тонкі плівки з різних металів на металізованих поверхнях, які завдяки своїм електрофізичним властивостям є важливими складовими багатьох пристроїв - від електрочутливих мікроелектронних компонентів до захисних покриттів на великих поверхнях. Нарощені тонкі плівки зазвичай виконують функцію збільшуючого електропровідність, зміцнюючого, захисного, або світловідбивного покриття [1]. З розвитком науки та техніки виникає потреба у вдосконаленні технологій виробництва електронних компонентів, тому вимоги до методів та приладів для гальванічного нарощування тонких плівок також постійно зростають.

Метою цієї роботи є розробка приладу для гальванічного нарощування тонких металевих плівок (ТМП), який би вираховував оптимальний час і параметри напилення за заданих вхідних параметрів; керував аналоговою периферією, яка регулює параметри нарощування; задовольняв сучасним вимогам щодо точності, однорідності та продуктивності процесу, при цьому будучи простим у використанні та легким і недорогим у виробництві. В якості основи приладу використовується програмована логічна інтегральна схема (ПЛІС) MAX10, завдяки якій досягається високих рівень синхронізації керуючих сигналів та можливість впровадження гнучкої в налаштуванні і діапазоні частот широтно-імпульсної модуляції (ШІМ). Напилені металеві плівки будуть розглядатися в контексті верхнього шару срібних електродів акустооптичного дефлектора та ряду технологічних процедур і вимог, дотримання яких грає ключову роль в отриманні якісних електродів.

У роботі розглянуто принципи технології гальванічного нарощування, основні технологічні параметри, що впливають на якість плівок, та наведено огляд сучасних методів і приладів, що використовуються при гальванічному

нарощувані ТМП. Актуальність роботи полягає в необхідності отримання високоякісних ТМП при виготовленні ряду електронних та акустооптичних компонентів, та можливості досліджувати вплив параметрів нарощування ТМП на їх характеристики. Метою роботи є розробка приладу для нарощування ТМП, за допомогою якого можна легко задавати параметри нарощування плівок в широкому діапазоні частот задля виявлення найбільш оптимальних параметрів нарощування ТМП в режимі імпульсного реверсивного покриття за даних умов та матеріалів.

Розділ 1
Аналіз предметної області

1.1 Тонкі плівки, їх параметри та властивості

ТМП називають плівки металів з товщиною в діапазоні від декількох нанометрів до близько десяти мікрометрів [1, 2]. Залежно від їх кінцевого призначення задані параметри плівок можуть бути різними, однак є ряд загальних характеристик і вимог, які властиві всім плівковим матеріалам. До них відносяться:

- рівномірність за товщиною;
- адгезія
- заданий хімічний і фазовий склад;
- розмір зерен та стан їхніх границь;
- орієнтація кристалітів [1, 2].

В залежності від товщини в ТМП спостерігаються різні розмірні ефекти – явища зміни фізичних та хімічних властивостей матеріалу, що змінюються в залежності від його розмірів. Розмірні ефекти зумовлені порушенням характерного для масивного матеріалу співвідношення між площею поверхні та об'ємом, та найбільш наглядно спостерігаються в плівках товщиною до 10 нм. Вони можуть бути різного характеру (квантові, механічні, магнітні, оптичні, тощо), бути зумовленими різними причинами (відмінним від масивних матеріалів співвідношенням площі поверхні та об'єму, відмінним від масивних матеріалів розміром кристалів, тощо) і, відповідно, по різному впливати на властивості плівок [3].

Існує ряд технологічних методів, які застосовують для напилення тонких плівок. Кожен з них має свої особливості, які визначають можливість та доцільність його застосування відповідно до кінцевого призначення напилених плівок.

1.2 Методи нанесення тонких металевих плівок

Основними трьома методами нанесення ТМП на підкладку є вакуумне термічне напилення, іонно-плазмове напилення та електрохімічне осадження (гальваностегія) [1]. Для нарощування ТМП методом електрохімічного осадження в більшості випадків потрібно використати один з перших двох методів для підготовки робочих зразків.

1.2.1 Вакуумне термічне напилення. Даний метод базується на випаровуванні матеріалу в умовах високого вакууму з подальшим конденсуванням пари на поверхні підкладки. Завдяки високому вакууму можна досягти високої чистоти і однорідності плівки, оскільки практично відсутні домішки газів і сторонніх часток. Також перевагою методу є його висока продуктивність. Недоліками методу є слабка адгезія плівки з підкладкою, через що плівки товщиною більше двох мікрометрів цим методом не наносять, низький коефіцієнт використання випаруваного матеріалу та складність отримання однорідної за товщиною плівки на підкладках складної форми [1, 2, 4].

1.2.2 Іонно-плазмове напилення. Це ряд високотехнологічних методів нанесення ТМП, що використовує плазму для прискорення іонів матеріалу і їх осадження на підкладку. Даний метод відрізняється високою енергією осаджуваних частинок, що дозволяє досягати щільних і міцних покриттів з високою адгезією до підкладки. Завдяки цим властивостям іонно-плазмове напилення широко застосовується в таких галузях як електроніка, оптоелектроніка та акустооптика.

Процес іонно-плазмового напилення складається з кількох ключових етапів. Спочатку в камеру з вакуумом вводиться газ аргон. Потім за допомогою високочастотного електричного поля газ іонізується, утворюючи плазму. Іони аргону бомбардують поверхню металевої мішені, з якої буде осаджуватися плівка, вибиваючи з неї атоми металу, які потім конденсуються на підкладці, утворюючи тонку плівку.

Перевагами іонно-плазмового напилення є однорідність плівок, гарна адгезія та можливість напилювати ТМП з різних матеріалів, включаючи тугоплавкі метали і складні сплави. Недоліками методу є складне обладнання та необхідність в ретельному контролі параметрів плазми, що може підвищувати вартість виробництва [1, 2, 4]. Магнетронне напилення, яке є різновидом іонно-плазмового напилення, буде використано в рамках цієї роботи для напилення тестових зразків.

1.3 Гальваностегія

Гальваностегія є одним з різновидів електролізу – електрохімічного процесу, в результаті якого, завдяки пропусканню через хімічні речовин струму, відбуваються розклади, яких неможливо досягти природнім шляхом. Речовина, яка проводить електричний струм (наприклад, розчин чи розплав солі) занурюють електроди: позитивно заряджений анод і від'ємно заряджений катод. Під впливом електричного струму відбувається окислення та відновлення речовин на електродах, що призводить до розщеплення сполук. Елементи чи сполуки, які виділяються на електродах, є продуктами електролізу. Прикладом використання електролізу є розщеплення води на водень та кисень (Рисунок 1.1).

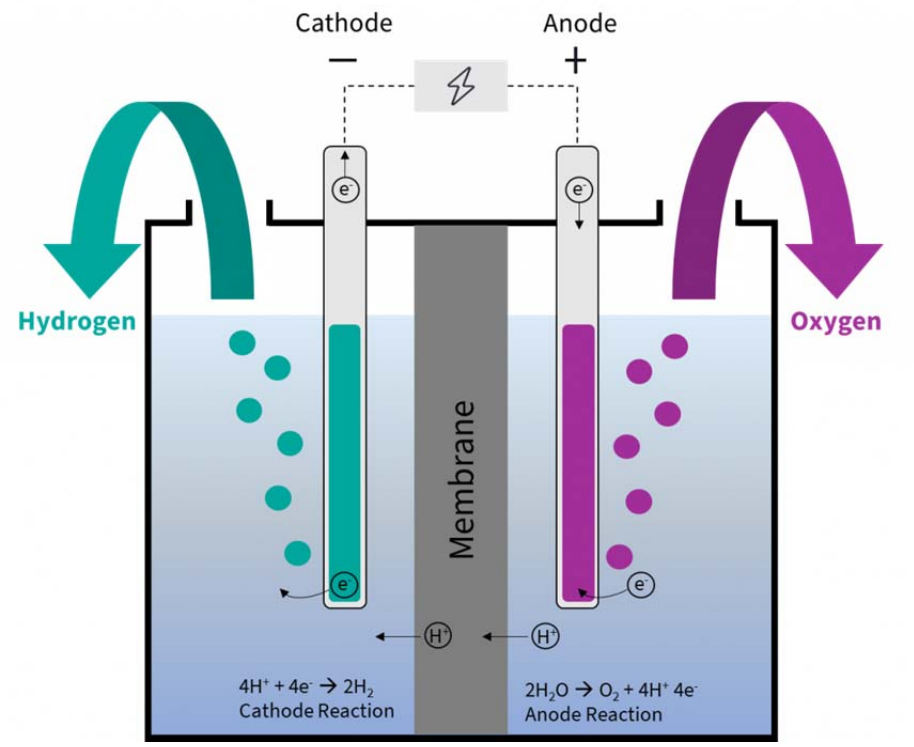

Рисунок 1.1. Схема розщеплення води на водень та кисень методом електролізу

Гальваностегія - це процес відновлення металів з їх іонів у вигляді твердого металевого осаду на електроді. Цей процес відбувається в електроліті, що містить іони металу, який нарощують. Під впливом електричного струму на аноді відбувається окислення металевих атомів, які переходять у стан іонів, а на катоді іони металу отримують електрони і відновлюються до металевого стану, утворюючи твердий осад. Таким чином на катоді формується шар металевої плівки [5].

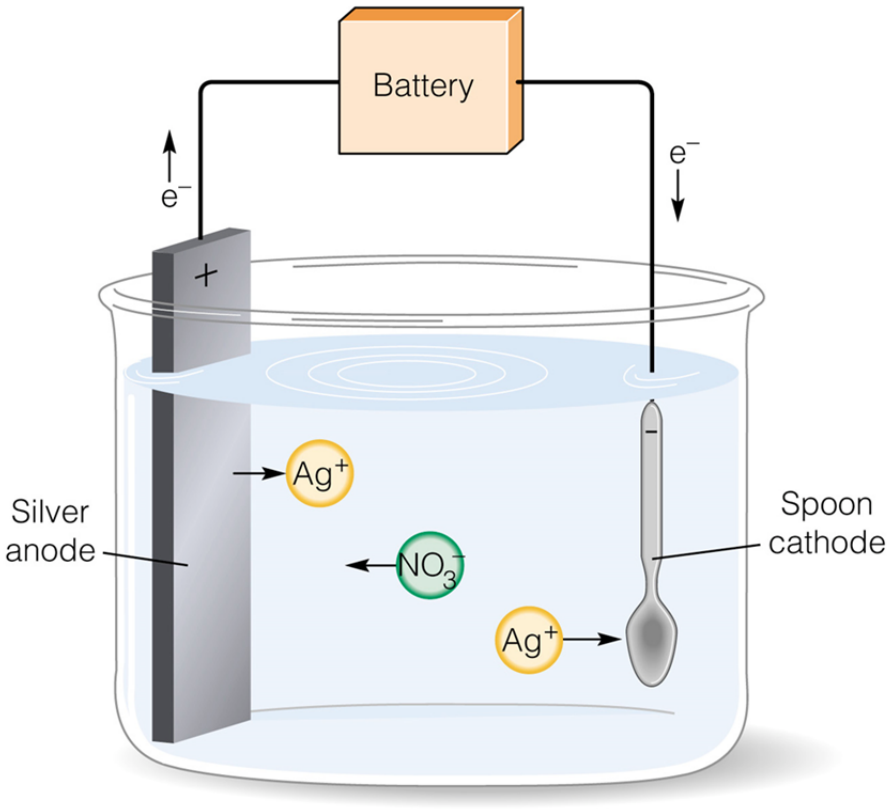

Рисунок 1.2. Схема гальванічного нарощування срібла

Нарощування металів проходить на молекулярному рівні, в мікро- та нанометровому діапазоні, дозволяючи наростити на деталі рівно стільки матеріалу, скільки потрібно для досягнення бажаних характеристик. Перевагами даного методу нарощування ТМП є фінансова ефективність, простота та висока швидкість процесу. Основними ж недоліками є необхідність в металевій основі та забруднення гальванічної ванни продуктами розпаду [2].

1.4 Закони Фарадея

Окрім самого отримання продуктів електролізу треба також розуміти, яку їх кількість та за який час вийде отримати. Закони Фарадея, також відомі як закони електролізу, описують залежність об'єму продуктів електролізу від кількості електричного заряду, який протікає через електроліт.

Формулювання першого закону Фарадея: маси речовин, що перетворюються на електродах під час електролізу, пропорційні кількості електрики, яка пройшла через електроліт.

$$m = k * Q \tag{1.1}$$

де m – маса продукту електролізу, k – електрохімічний еквівалент речовини, Q – повний електричний заряд, що пройшов через речовину.

Оскільки електрохімічний еквівалент речовини k означає масу речовини, що перетворюється при проходженні через електроліт 1 Кл заряду, то:

$$k = \frac{M_E}{F} \tag{1.2}$$

Підставляючи значення k та Q отримаємо:

$$k = \frac{M_E * I * t}{F} \tag{1.3}$$

Формулювання другого закону Фарадея: однакові кількості електрики виділяють при електролізі еквівалентні кількості будь-яких речовин, тому маси речовин, що перетворюються на електродах при проходженні однакової кількості електрики, пропорційні їх еквівалентним масам, а об'єми – еквівалентним об'ємам:

$$\frac{m_1}{m_2} = \frac{M_{E1}}{M_{E2}}, \qquad \frac{V_1}{V_2} = \frac{V_{E1}}{V_{E2}} \tag{1.4), (1.5}$$

Однак маса речовини, що виділяється при електролізі, фактично завжди менша від теоретично обрахованої за законами Фарадея. Це пов'язано з втратами струму та вторинними процесами, які відбуваються при електролізі. Вихід за струмом (η) є відношенням фактично одержаної при електролізі маси речовини ($m_{прак}$) до теоретично обчисленої ($m_{теор}$) і обчислюється за формулою [6]:

$$\eta = \frac{m_{прак}}{m_{теор}} * 100\% \tag{1.6}$$

Вихід за струмом при гальванічному нарощуванні більшості металів становить близько 90%, однак для таких процесів як нарощування хрому цей показник є значно меншим і в кращому випадку складає всього 20% [7].

Також при розрахунках часу нарощування для досягнення бажаних результатів треба враховувати валентність атомів металів, що використовуються, бо саме валентні електрони втрачаються в процесі окиснення. Наприклад, срібло, у власних сполуках, є одновалентним, а хром – шестивалентний. Для досягнення однакової товщини плівок обох металів доведеться або в 6 разів збільшити струм у випадку нарощування хрому, порівняно зі струмом при нарощувані срібла, або в стільки ж разів збільшити час нарощування для хрому.

1.5 Параметри гальваностегії

Гальванічне нарощування є складним процесом, в якому цілий ряд параметрів визначає якість та характеристики отриманих плівок. Контроль цих параметрів є критично важливим для досягнення плівок з бажаними характеристиками, таких як товщина, електропровідність, адгезія, тощо. Деякі з цих параметрів є загальновідомими і випливають з знань шкільної хімії та законів Фарадея. Це температура, склад електроліту, густина струму, час осадження. Проте навіть з контролем цих параметрів можуть виникнути певні складнощі. Наприклад, при нарощуванні плівок в звичайних кімнатних умовах, температура, за якою проходить реакція, в різні пори року буде різна; навіть за умови нарощування тільки одного виду металу постійно проводити нарощування в одному конкретному електроліті на протязі тривалого часу дуже складно. Однак в електрохімічній літературі висвітлюються більш технічні і комплексні параметри, розрахунком і описом яких займаються спеціалісти з галузі електрохімії. Їх розгляду присвячуються цілі підручники, тому в рамках цієї роботи розглядаються лише деякі з них.

1.5.1 Розсіювальна здатність (електроліту). В англомовній літературі позначається як «throwing power». Розсіювальна здатність визначає рівномірність розподілу металу на поверхні катода та часто вживається в контексті здатності зробити в процесі металізації

однорідний металевий шар на катоді складної форми. Розсіювальна здатність залежить в першу чергу від розподілу струму в гальванічній ванночці, але такі фактори як провідність електроліту, концентрація іонів в ньому, температура та провідність підкладки також впливають на цей параметр[6, 8]. Оксидні плівки на основі для нарощування плівок мають негативний ефект на розсіювальну здатність [8], а концентрація іонів металів часто обмежена їх розчинністю [6].

1.5.2 Покриваюча потужність. В англомовній літературі позначається як «covering power». Вона характеризує рівень металізації поверхні за дуже низької густини струму за умови існування заглиблень та поверхневих щілин. Цей термін передбачає здатність утворювати покриття, але не обов'язково однорідні, на відміну від розсіювальної здатності [8].

1.5.3 Розподіл струму. Відстань, за яку струм протікає в розчині від аноду до відповідних ділянок катоду, є критично важливою, оскільки навіть невелика різниця в надлишковому потенціалі може спричинити значну різницю в локальній швидкості реакції [6]. Декілька анодів може бути задіяно одночасно задля того, щоб зробити щільність струму більш рівномірною. Також варто на ранніх етапах проектування впевнитися, що підготовлена ванночка має оптимальний розмір для катоду та аноду гальванічного нарощування. На Рисунку 1.3 схематично зображено вплив розподілу струму на однорідність нарощування [6].

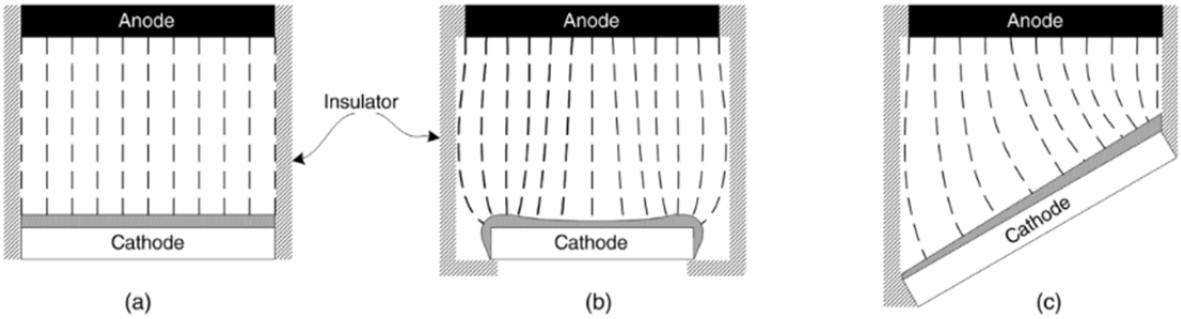

Рисунок1.3. Схематичне зображення впливу розподілу струму на однорідність нарощування

1.5.4 Густина струму та температура. Однорідність металевого осаду зменшується зі збільшенням середньої густини струму. Це відбувається тому, що омічний опір стає більшою частиною загального опору при більш високих густинах струму. Провідність та густина струму зростають зі збільшенням температури, що призводить до того, що, при заданій напрузі, з температурою зростає і густина струму. Отже, збільшення температури при постійній напрузі робить розподіл струму менш рівномірним. І навпаки - підвищення температури при постійній густині струму збільшує рівномірність осаду. Також слід зазначити, що з підвищенням температури зростає й швидкість побічних реакцій, що може призвести до більших втрат ефективності [6].

1.5.5 Побічні реакції. Побічні реакції - це реакції, які безпосередньо не сприяють утворенню бажаного осаду. Для осадження металів з водних розчинів найпоширенішою побічною реакцією є електроліз води або відновлення іонів водню, що призводить до виділення газоподібного водню при катодних потенціалах необхідних для осадження металів. Частина струму, яка призводить до виділення водню, є струмом, який не призводить до осадження металу, отже, являє собою втрату ефективності. Втрата ефективності є одним з основних наслідків побічних реакцій [6].

1.6 Методи гальванічного нарощування плівок

Виділяють три основні методи гальванічного нарощування плівок: метод нарощування при постійному струмі, метод імпульсного покриття (PP) та метод імпульсного реверсивного покриття (PRP). Вибір конкретного методу зазвичай залежить від галузі, в якій використовується напилення. Так, наприклад, у випадках, коли метою є отримання захисних або декоративних плівок, нарощування відбувається при постійному струмі, оскільки цей метод

є технічно найменш вибагливим та є оптимальним у випадках, коли розмір зерен нарощеного матеріалу та однорідність напиленої плівки не є критичними характеристиками. Натомість, в процесі виробництв електронних компонентів, коли важливою складовою нарощених плівок є досягнення певних значень деяких її параметрів, використання більш сучасних методів PP та PRP є більш доцільним.

1.6.1 Параметри методів імпульсних покриттів. Для охарактеризування імпульсів струму в методі PP потрібно знати три параметри: катодну пікову густину струму (j_c), тривалість катодного імпульсу (t_c) та інтервал між імпульсами (t_p), які зображено на Рисунку 1.4 [9].

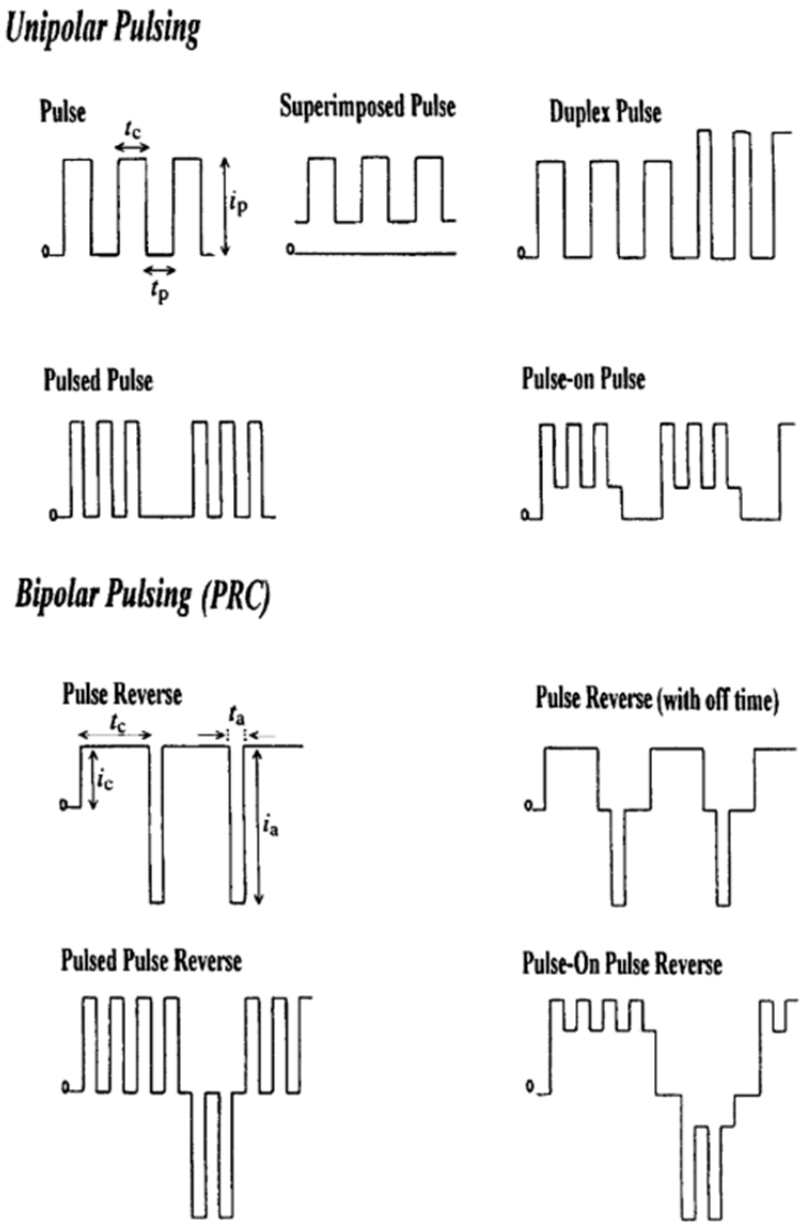

Рисунок 1.4. Форми сигналів за уніполярних та біполярних режимів нарощування ТМП

Середня густина струму в методі РР виражається формулою:

$$j_{AV} = \frac{(j_c * t_c)}{(t_c + t_p)} \tag{1.7}$$

Робочий цикл (Т), що представляє частину часу у кожному циклі, коли струм увімкнено, визначається наступним рівнянням:

$$T = \frac{t_c}{(t_c + t_p)} \tag{1.8}$$

Для охарактеризування імпульсів струму в методі PRP замість інтервалу між імпульсами додаються параметри анодної пікової густину струму (j_a) та тривалість анодного імпульсу (t_a) [9, 10]. Відтак рівняння середньої густини струму має вигляд:

$$j_{AV} = \frac{(j_c * t_c + j_a * t_a)}{(t_c + t_a)} \quad (1.9)$$

Робочий цикл для PRP визначається наступним рівнянням:

$$T = \frac{(j_c * t_c + j_a * t_a)}{(j_c * t_c)} \quad (1.10)$$

Зображені на Рисунку 1.4 форми сигналів є досить простими, однак проводилися і тестування впливу трикутних сигналів з включенням додаткових параметрів до форми сигналу в методі РР на якість нарощуваних плівок. Результати дослідження були позитивними, оскільки нарощування ТМП трикутною формою сигналу мало помітно кращі результати по параметру розсіювальної здатності порівняно з нарощуванням сигналом прямокутної форми [11].

1.6.2 Переваги застосування імпульсних покриттів. Уніполярний імпульсний струм став заміняти постійний завдяки здатності впливати на механізми електрокристалізації, що, в свою чергу, впливає на фізичні та механічні властивості гальванічно нарощеного металу. Оскільки швидкість нуклеації зростаючого електроосаду пропорційна густині струму, використання імпульсів високої густини струму дозволяє отримувати ТМП зі зниженою пористістю і, в більшості випадків, більш дрібними зернами[9, 10, 12, 15]. Контроль пористості нарощених плівок є досить проблематичним у разі використання постійного струму [12]. Отримання дрібнозернистої плівки на практиці залежить від того, що відбувається під час інтервалу між імпульсами, коли струм і переривається, оскільки це може сприяти десорбції домішок і стимулювати ренуклеацію з утворенням нових, менших кристалічних зерен [9].Зменшення розміру зерен сприяє збільшенню електропровідності та щільності плівки, а також зменшенню її пористості.

Зменшення пористості на практиці означає, що можна наносити більш тонкий, наприклад, золотий осад, і мати при цьому задовільні показники пористості [9].

Перевагою методу PRP є гальванічне полірування частини щойно нарощеного шару металу, що дозволяє прибирати можливі нерівності кожного циклу та зберігати гладку поверхню ТМП протягом всього процесу нарощування [9, 10, 15].

1.6.3 Порівняння плівок напилених при постійному струмі та імпульсному покритті. Виходячи з наведених вище показників, по яким метод PRP переважає над методом постійного струму, на Рисунках 1.5, 1.6 наведені порівняння нарощених за різних методів ТМП.

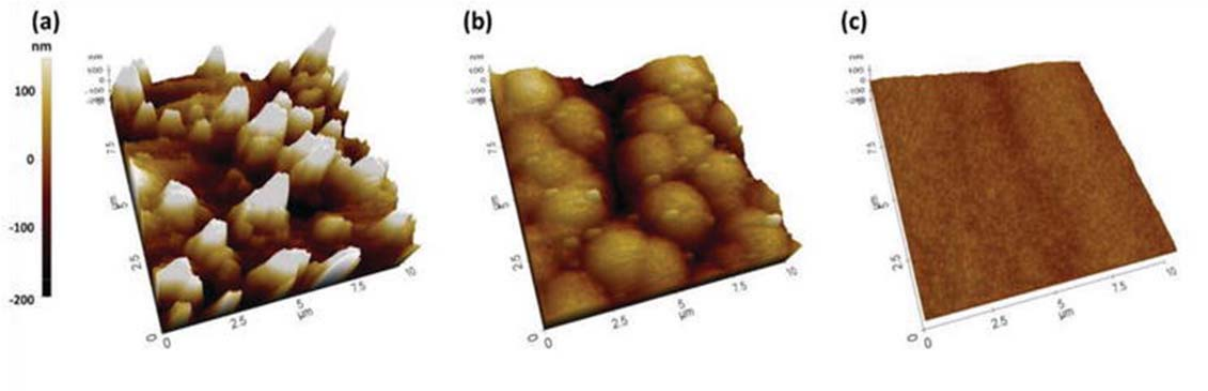

Рисунок 1.5. АСМ-мікрофотографії (а) поверхні золотої плівки, отриманої методом постійного струму в ціанідному електроліті, (б) поверхні золотої плівки, отриманої методами постійного струму та (в) РР в електроліті на основі сульфіту [13].

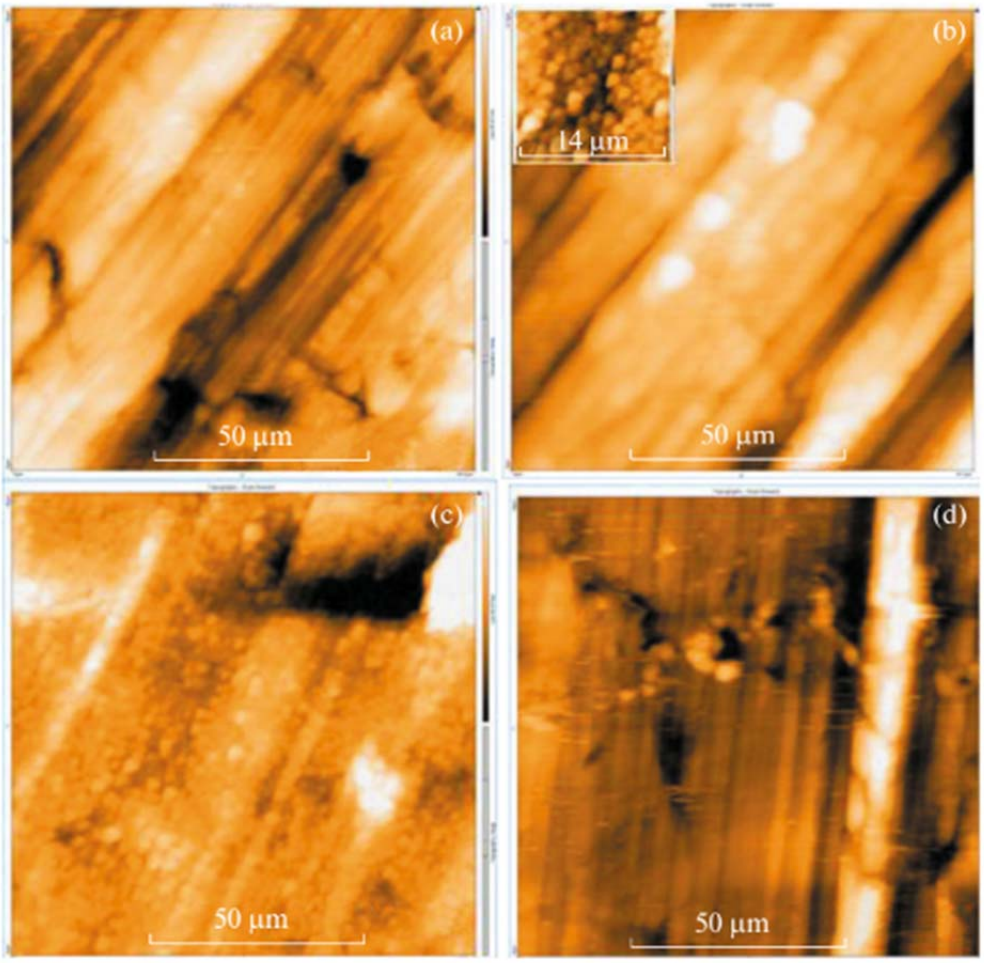

Рисунок 1.6. АСМ-зображення (а) чистого срібного сплаву, (б) Ag нарощеним методом PP, (в) Ag нарощеного методом PRP і (г) Ag нарощеного постійним струмом [14].

1.6.4 Обмеження методів імпульсних покриттів. В різних джерелах наводяться різні числові значення обмежень параметрів гальванічного нарощування плівок, але загальна концепція проблеми застосування високих частот при гальванічному нарощувані ТМП зберігається: при високих частотах анод і катод не встигають перезаряджатися при зміні напрямку струму, що має згладжуючий ефект на форму сигналу, яка починає наближатися до постійного струму. Це обмежує максимальну корисну частоту близько 500 Гц для більшості застосувань; однак, більш високі частоти можуть бути використані, коли застосовуються дуже високі щільності імпульсного струму [9, 10]. На Рисунку 1.7 зображено

нарощення ТМП за значно вищих за рекомендованих значень частоти імпульсів [16].

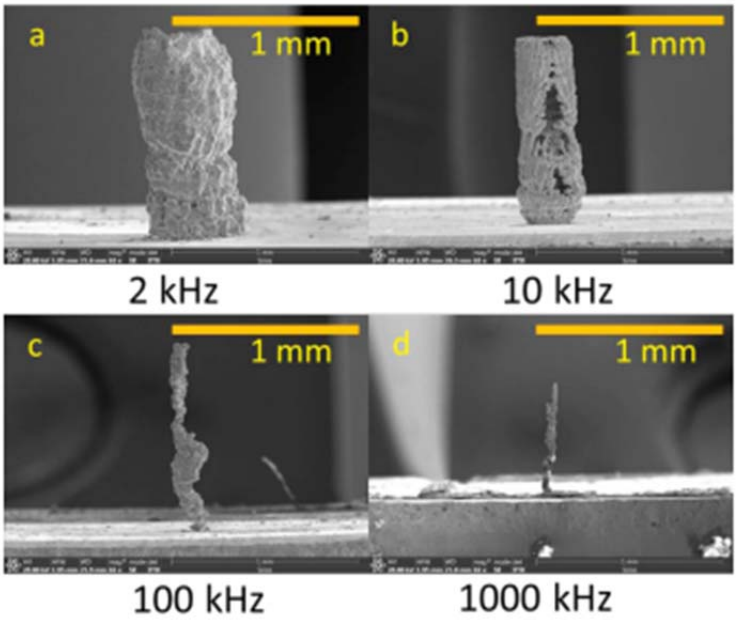

Рисунок 1.7. СЕМ-зображення нарощених стовпчиків під дією різних частот імпульсів

Існують також обмеження по значеннях робочого циклу. При збільшенні робочого циклу нарощування починає наближатися по характеристиках до постійного струму. За наявності достатньої резервної потужності випрямляча робочий цикл від 33 до 50%, ймовірно, є мінімальним практичним значенням [9]. На Рисунку 1.8 зображено вплив катодного робочого циклу на якість ТМП [17].

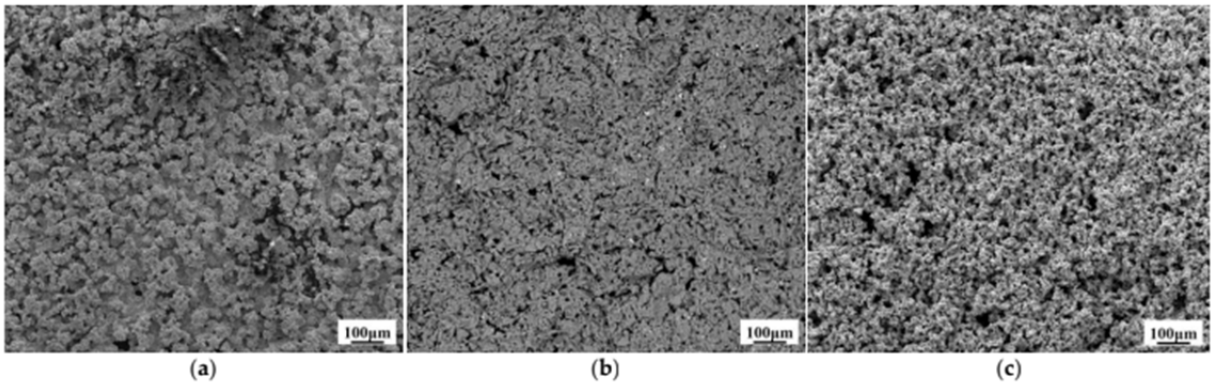

Рисунок 1.8. Зображення поверхневих СЕМ композитних покриттів W-Cu при різних катодних робочих циклах: (a) df = 20%, (b) df =40%, і (c) df = 60%, (J = 2 А/дм2, f = 1500 Гц, dr = 5%).

1.7 Сучасні прилади для гальванічного нарощування плівок

В залежності від сфери виду виробництва та вимог до напилених металевих плівок, використовується один з декількох видів обладнання, що забезпечують процес гальванічного нарощування плівок: джерело постійного або імпульсного струму, програмована установка для гальваніки [18].

Як було зазначено раніше, здебільшого гальванічне нарощування проводиться за використанням джерела постійного струму [6]. Імпульсне покриття, хоч і має ряд переваг, не є обов’язковим для досягнення бажаних характеристик нарощених ТМП для ряду задач. Саме з цієї причини більшість приладів для гальванічного нарощування плівок використовують постійний струм. Такі прилади переважно розраховані на промисловий масштаб робіт, від чого можуть видавати високу густину струму на високих напругах та мають гальванічні ванни великого об’єму. Більш дорогі моделі оснащені підігрівом ванни та механізмом, що повертає або переміщує анод, але багато базових моделей не мають функції розрахунку часу нарощування задля отримання бажаної товщини плівок. Приклад типового приладу для гальванічного нарощування ТМП наведений на Рисунку 1.9.

Рисунок 1.9. Прилад для гальванічного нарощування ТМП моделі Electroplating Unit Comfort II

Розділ 2
Класична технологія гальванічного нарощування тонких металевих плівок

2.1 Підготовка необхідних матеріалів

Опанування технології гальванічного нарощування відбувалося за методом нарощування плівок при використанні джерела постійного струму. Перед початком роботи був підготовлений ряд необхідних для неї компонентів, таких як: ванночка для гальваніки, срібний безцианістий електроліт, тестові зразки Хром-Срібло, припій POI-50, підставка під дроти і зразок, тримач зразку, посріблені дроти та срібний анод. Оскільки в даній роботі технологія гальванічного нарощування тонких плівок розглядається на прикладі напилення плівки срібла на електрод акустооптичного дефлектора, то вибір всіх описаних компонентів відбувався з дотриманням вимог до технології для нарощування плівок для даних компонентів:

- POI-50 є припоєм, що складається на 50% з олова та на 50% з індію, та має температуру плавлення близько 110° С; в даному процесі використовується саме він, оскільки підкладка акустооптичого дефлектора виготовляється з індію, що має температуру плавлення 160° С, тобто, якщо припаювати дротики до зразку більш звичним сплавом, наприклад ПОС-61, то при роботі з реальними акустооптичними компонентами їх підкладка розплавиться.
- Посріблені дроти з центральної жили коаксіального кабелю РКТФ-71 використовуються задля мінімізації опору електродів, що є складовою процесу гальванічного нарощування.

З обладнання були підготовані лабораторний блок живлення, паяльна станція, мультиметр та клеми з крокодилами на протилежних кінцях з м’якими дротами, щоб не перевішувати конструкцію зі зразком в ванночці. Практично все, що не входило в перелік основних хімічних компонентів,

робилося з підручних засобів. Зразки хром-срібло були нанесені методом магнетронного напилення на предметних скельцях для мікроскопії. Оскільки цей метод є дуже трудомістким і для отримання пари зразків потрібні значні фінансові та часові витрати, в даній роботі в цьому та розділі фінального тестування використовуються лише по декілька зразків на розділ.

2.2 Опрацювання технології в лабораторії

Процес гальванічного нарощування ТМП починається з фіксації зразку на підставці (Рисунок 2.1) і підготовці тонких срібних дротиків для напаювання по краям зразку, щоб зробити з нього катод.

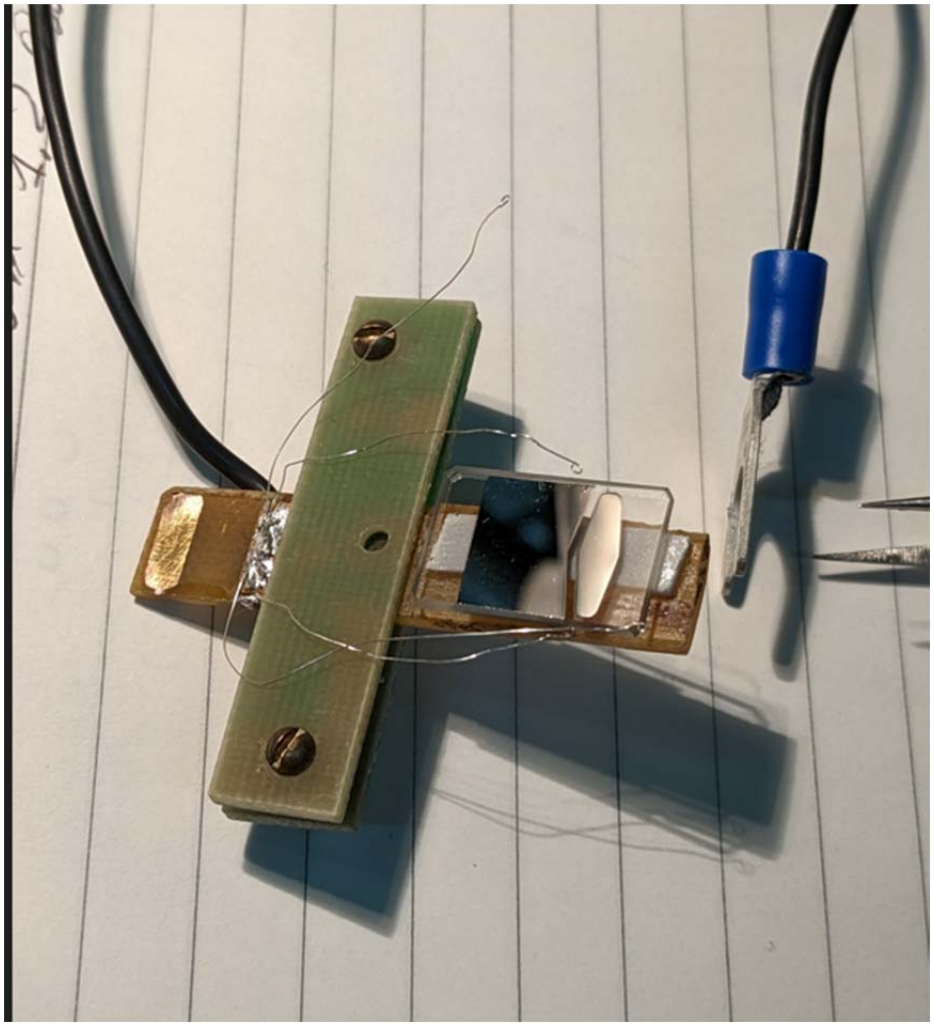

Рисунок 2.1. Зразок зафіксований на підставці

Дротики закручувалися на одному з кінців в петельку з одного витку (Рисунок 2.2), що значно полегшувало процес монтажу дротику до поверхні зразку.

Рисунок 2.2. Петелька на кінці дротику з центральної жили коаксіального кабелю РКТФ-71

Після того зразок на підставці з припаяними дротиками фіксувався тримачем на ванночці з електролітом. До місця з'єднання дротиків під'єднується негативний вивод блоку живлення, а до срібного елементу на іншому кінці ванночки – позитивний вивод. Блок живлення переводиться в режим стабілізації струму і виставляється на 5 мА – оптимальне значення для площі 1 $см^2$ металевої поверхні, яку і складає наш зразок. При включені блоку живлення і очікуванні 4 хвилини 35 секунд, які було встановлено як оптимальний час напилення за даних умов при попередніх нарощеннях, на зразок нарощується близько 1 мкм срібла (Рисунок 2.3).

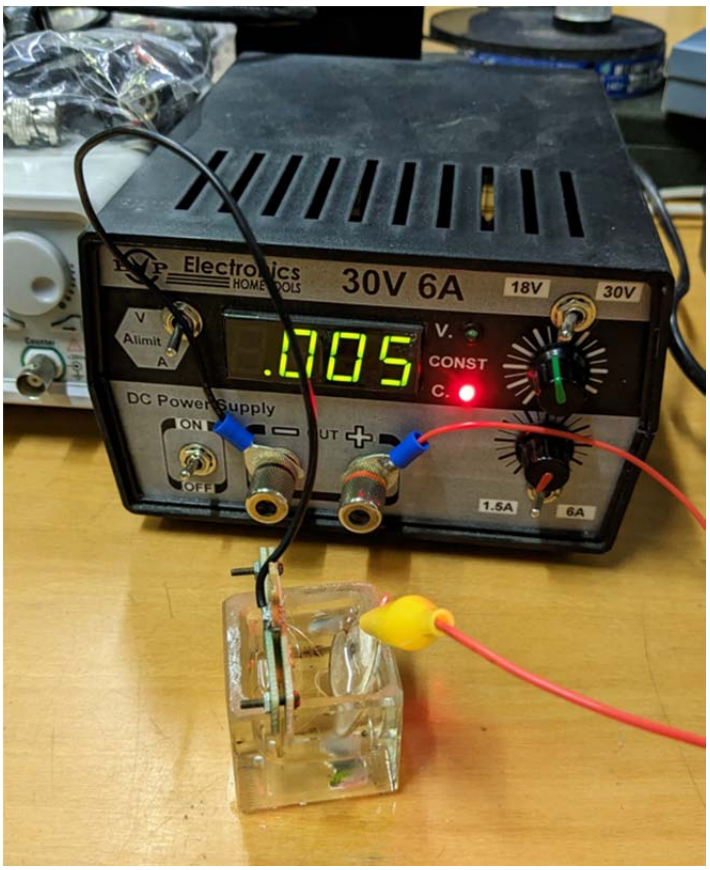

Рисунок 2.3. Зразок у ванночці з срібним елементом, підключені до лабораторного блоку живлення

В такому режимі було напилено декілька зразків та виміряно вольт-амперну характеристику процесу гальванічного нарощування за даних параметрів відстані від срібного елементу до зразку, особливостей електроліту, тощо. В Таблиці 2.1 наведені значення напруги при збільшені струму на блоці живлення.

Таблиця 2.1

Вольт-амперна характеристика процесу гальванічного нарощування

I	V
0.004	0.8
0.008	1.24
0.016	1.5
0.04	1.78

Графік, побудований з даних зображених в Таблиці 2.1, представлений на Рисунку 2.4.

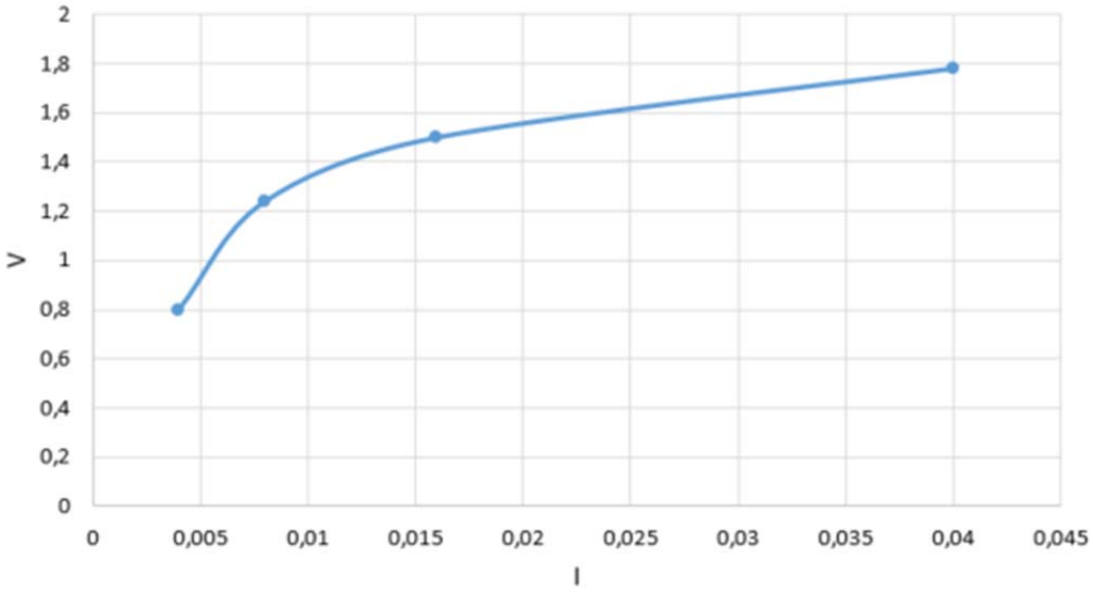

Рисунок 2.4. Графік залежності напруги від заданого струму

Два зразки, на які срібло нарощувалося протягом 4 хв 35 с при виставленому струмі 5 мА, зображено на Рисунку 2.5. Зразок зліва мав гарний контакт, в той час як від зразку справа в різні моменти протягом нарощування відпало два дротики.

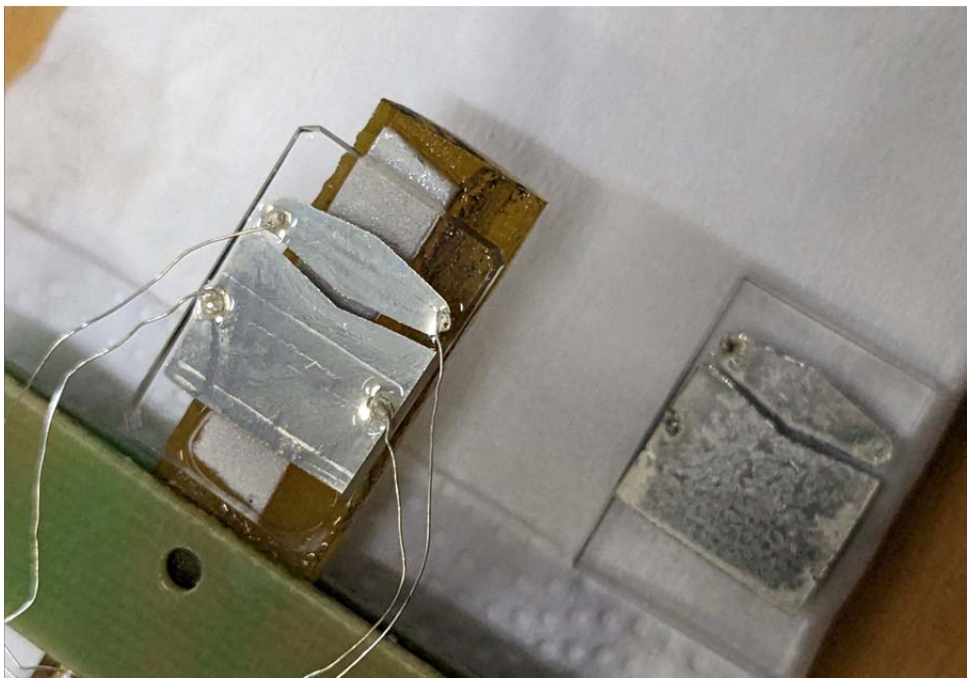

Рисунок 2.5. Два зразки з нарощеною плівкою товщиною в мікрон, але з різною якістю контакту з дротами-катодами.

Дане лабораторне дослідження з метою опанування технології гальванічного нарощування тонких плівок срібла з застосуванням джерела постійного струму не тільки наочно ознайомило з цією технологією, а й продемонструвало, що в подібних процесах можливі певні механічні складнощі і перешкоди, які не висвітлюються в статтях і наукових роботах. Такі речі, як складність надійної фіксації зразку у ванночці для запобігання від'єднання контактів від зразку буде прийнято до уваги і опрацьовано під час проектування приладу.

Розділ 3
Розробка приладу

3.1 Визначення основних параметрів приладу

Проектування приладу проходило з дотриманням наступних технічних вимог:

1. Прилад повинен живитися від 5 В через роз'єм micro USB.
2. Прилад повинен приймати значення від користувача:
   - матеріалу напилення (Ag, Cu, Au, Ni);
   - площі зразку, в діапазоні від 5 до 50 мм^2;
   - товщини плівки, в діапазоні від 0.01 до 30 мкм;
   - частоти катодного імпульсу, в діапазоні від 10 до 5000 Гц.
3. Прилад повинен розраховувати час нарощування плівок за законами Фарадея, та проводити налаштування і керування апаратної складової в процесі нарощення.

3.2 Вибір компонентів

За основу приладу, як і зазначалося раніше, була взята ПЛІС MAX10 на платі розробника від компанії Arrow. Ключовими аналоговими компонентами приладу було обрано декілька ІС виробництва Analog Devices, оскільки це якісний виробник з чудовою репутацією і вражаючим каталогом ІС з широким діапазоном значень параметрів. Кожна ІС підбиралася переважно за критерієм внутрішньої конфігурації, але також для всіх наведених нижче компонентів були деякі спільні критерії , а саме поєднання потрібної конфігурації з мінімальним опором та якомога меншим енергоспоживанням. Окрім цього було підібрано ряд периферійних складових у вигляді OLED дисплею, мембранних кнопок керування, індикаторних світлодіодів.

3.2.1 Плата MAX1000. Плата MAX1000 від компанії Arrow (Рисунок 3.1) спроектована навколо ПЛІС MAX10 від компанії Intel, і є логічною основою цієї роботи. Це потужний і універсальний інструмент для розробки цифрових систем що поєднує в собі високу продуктивність та простоту у використані і розробці пристроїв. Плата MAX1000 є однією з найбільш дешевих плат на основі ПЛІС, що має 8000 логічних комірок, майже 400 кБайт оперативної пам’яті та ряд периферійних компонентів, таких як акселерометр, SDRAM, тощо [19, 20].

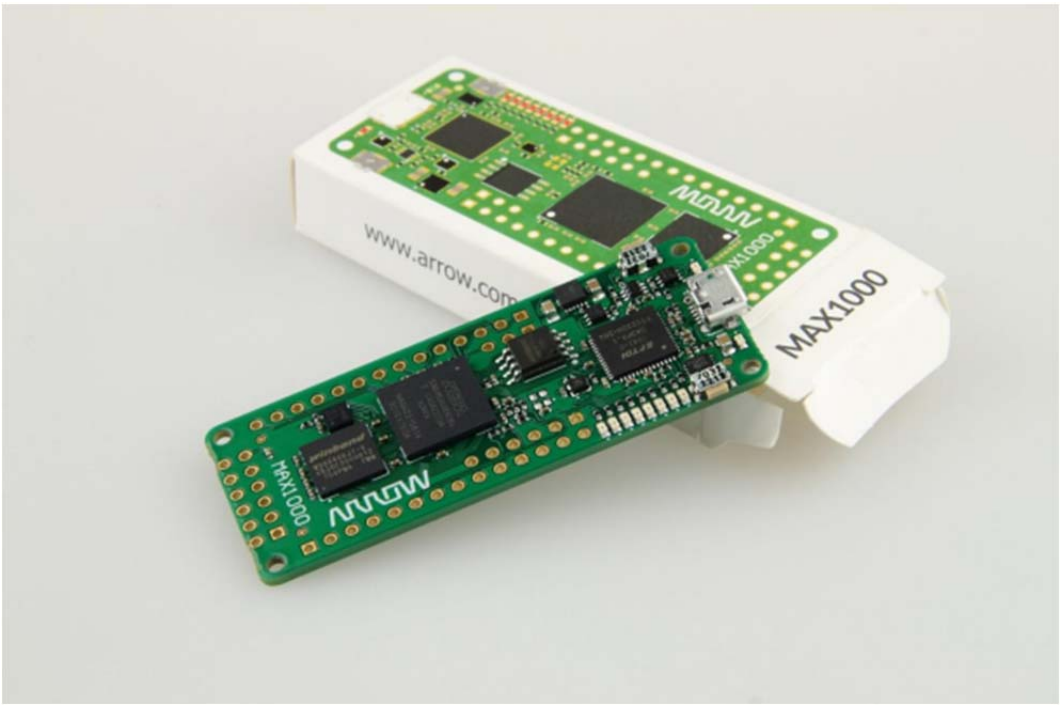

Рисунок 3.1. Плата MAX1000

3.2.2 Програмоване джерело струму LT3092. Дана ІС здатна видавати вихідний струм в діапазоні від 0.5 мА до 200 мА, який програмується за допомогою двох зовнішній резисторів, та має високі показники надійності та точності. Структурна схема ІС наведена на Рисунку 3.2 [21].

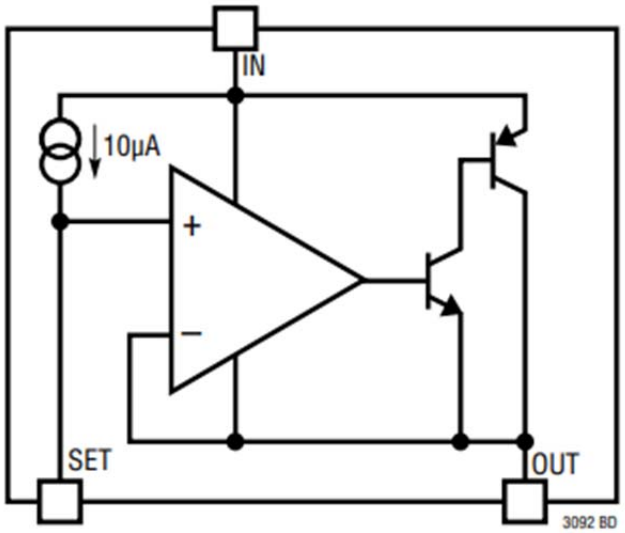

Рисунок 3.2. Структурна схема IC LT3092

3.2.3 Двоканальний перемикач ADG1636. IC представляє собою подвійний SPDT (Single Pole Double Throw) перемикач в мініатюрному корпусі з цифровим керуванням. ADG1636 має низький опір в увімкненому стані та високу швидкість перемикання контактів. Таких в даній роботі використовується два: один для організації швидкої зміни полярності струму, а другий задля переключенні резисторів на програмованому вході IC LT3092 для налаштування струму катодного і анодного імпульсів. На Рисунку 3.3 наведено структурну схему даного перемикача [22].

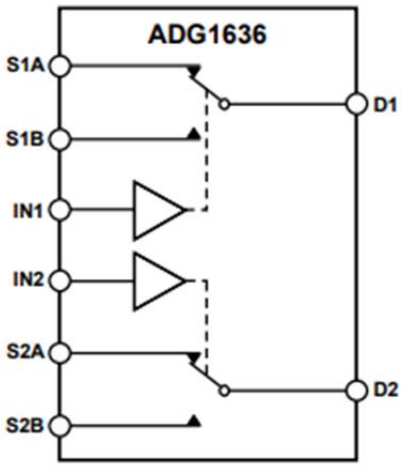

Рисунок 3.3. Структурна схема IC ADG1636

3.2.4 Цифровий потенціометр AD8402. IC AD8402 містить два цифрових потенціометри на 10 кОм з 256 кроками кожен, що забезпечує високу точність налаштування опору без впроваджень окремих механічних частин в будову приладу. Поточне значення потенціометру виставляється програмним чином, а саме передачею 8-бітного значення в IC по протоколу SPI. Один потенціометр буде підлаштований для отримання оптимального значення струму катодного імпульсу, а інший – для анодного. Структурна схема даного потенціометру наведена на Рисунку 3.4 [23].

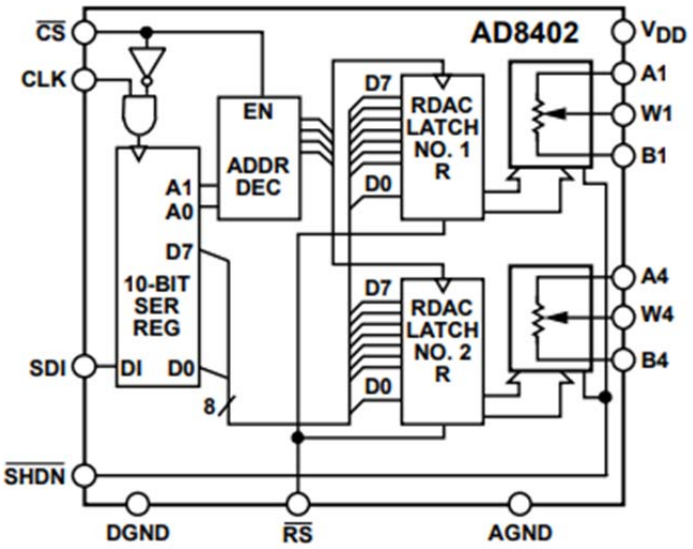

Рисунок 3.4. Структурна схема IC AD8402

3.2.5 Інші компоненти. Елементами взаємодії користувача з приладом є монохромний OLED дисплей, 5 мембранних кнопок та індикаторні світлодіоди червоного та зеленого кольорів. Перелічені компоненти зображені на Рисунку 3.5.

Рисунок 3.5. Компоненти взаємодії користувача з приладом

Обраний монохромний OLED дисплей має діагональ 2.42 дюйма та роздільну здатність 128*64 пікселі. На платі дисплею стоїть драйвер SSD1309, через який відбувається комунікація між мікроконтролером та дисплеєм по послідовному інтерфейсу I2C. Перевага OLED дисплею над більш звичним рідкокристалічним дисплеєм полягає в тому, що кожен піксель випромінює світло індивідуально, а не освітлюється джерелом світла. Завдяки цьому OLED дисплеї споживають менше електроенергії та формують більш контрастне зображення.

Мембранні кнопки в конфігурації панелі з чотирма стрілками і центральної кнопки задовольняють поставленим вимогам до елементу вводу користувачем робочих параметрів в програму пристрою. Індикаторні світлодіоди будуть вказувати в якому режимі перебуває пристрій: в режимі очікування вводу даних чи в процесі нарощування плівки.

3.3 Принципова схема та симуляція її роботи

З описаних вище компонентів була спроектована принципова схема аналогової частини приладу, яка наведена на Рисунку 3.6. OLED дисплей не наведений на цій схемі, але буде в подальшому підключений до виводів ПЛІС, відведених під I2C інтерфейс.

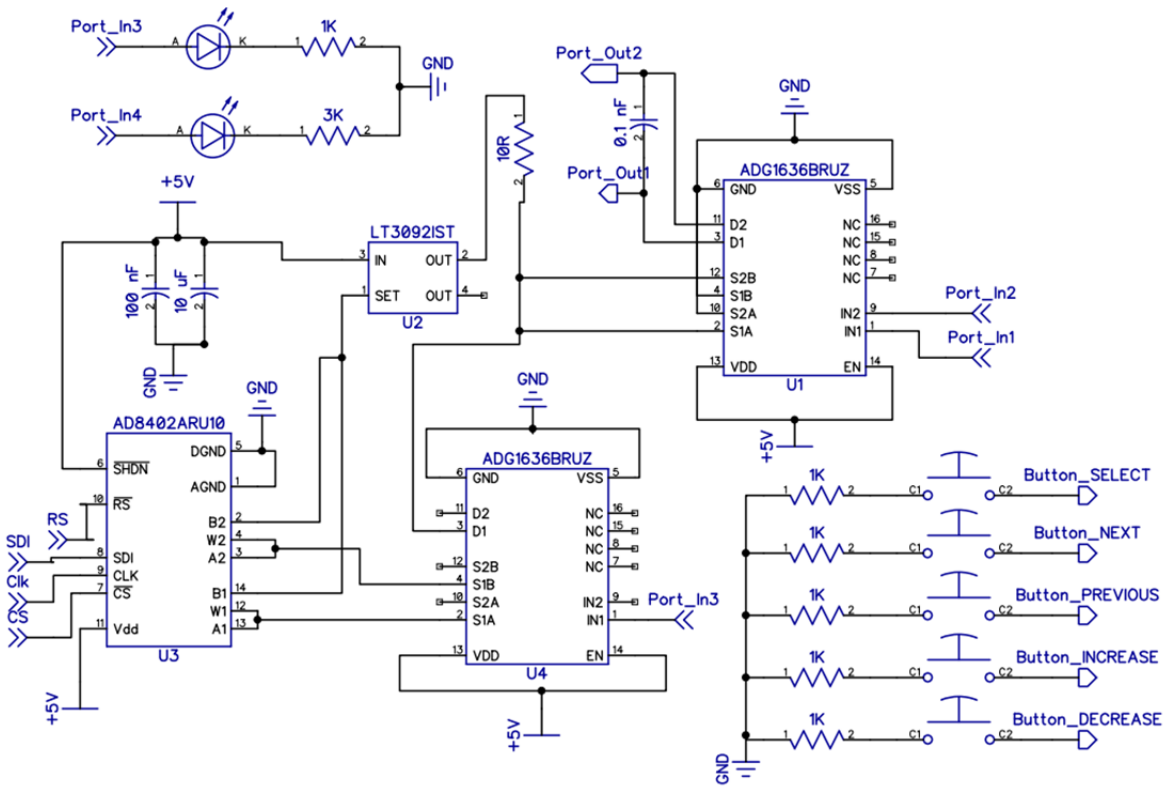

Рисунок 3.6. Принципова схема аналогової складової приладу

Дана схема живиться від 5 В виводу живлення на платі MAX1000. Серед компонентів схеми є:

- IC програмованого джерела струму LT3092, за допомогою якої реалізується генерація різного значення струму, в залежності від обраного матеріалу нарощування і виду електроду, до якого вона під'єднана;
- двоканальний цифровий потенціометр AD8402, два резистори якого виконують налаштування анодного та катодного струмів.

- дві двоканальні IC SPDT перемикачів, що міняють місцями контакти живлення на електродах, змінюючи напрям струму, та перемикаються між двома резисторами IC AD8402 для виставлення меншої густини струму на LT3092 на час катодних імпульсів, і збільшуючи густину струму для анодних імпульсів;
- мембранні кнопки для взаємодії користувача з приладом, з'єднані з живленням через підтягуючі резистори;
- світлодіоди для індикації поточного режиму роботи приладу;
- стабілізуючі конденсатори.

Симуляція даної схеми проводилася в програмному забезпеченні LTspice, в бібліотеках якого містяться всі використані в схемі IC, що полегшило процес дублювання схеми в програмне забезпечення. На Рисунку 3.7 зображену схему побудовану в LTspice.

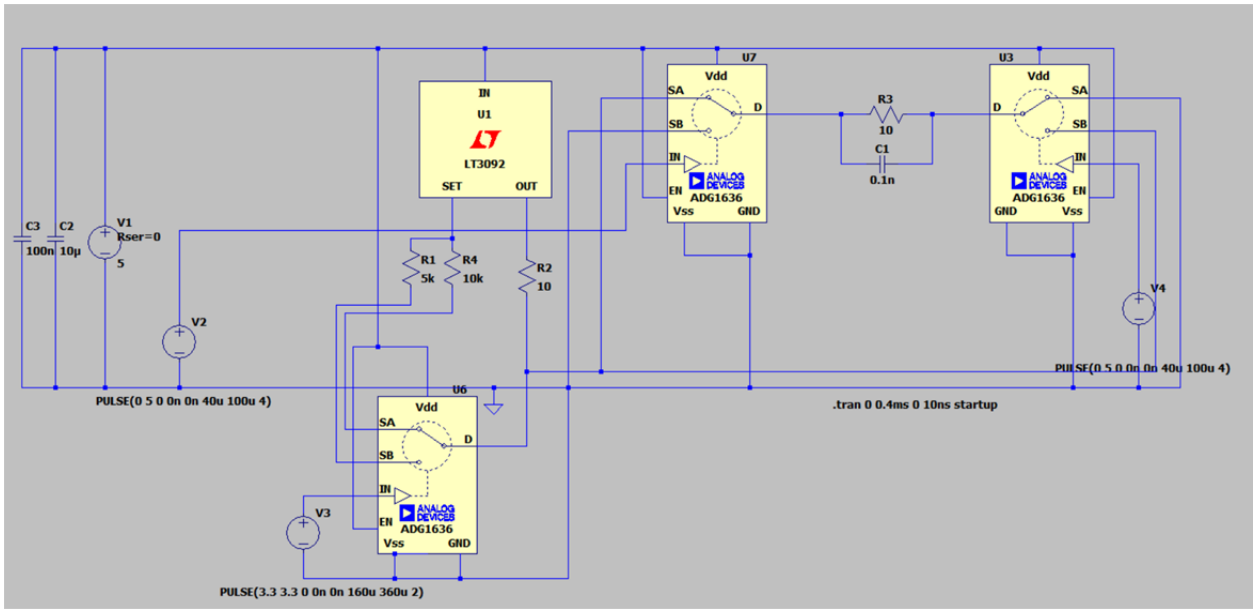

Рисунок 3.7. Принципова схема побудована в програмному забезпеченні LTspice

В даній симуляції не реалізована зміна струму для катодних і анодних імпульсів, а стоїть спільний час імпульсу 40 µs при часі циклу 100 µs. Основним завданням було перевірити форму струму, який протікає через

гальванічну ванночку, в умовах високочастотного перемикання контактів на аноді і катоді. Рисунок 3.8 демонструє форму струму в гальванічній ванночці, замість якої на схемі стоїть резистор.

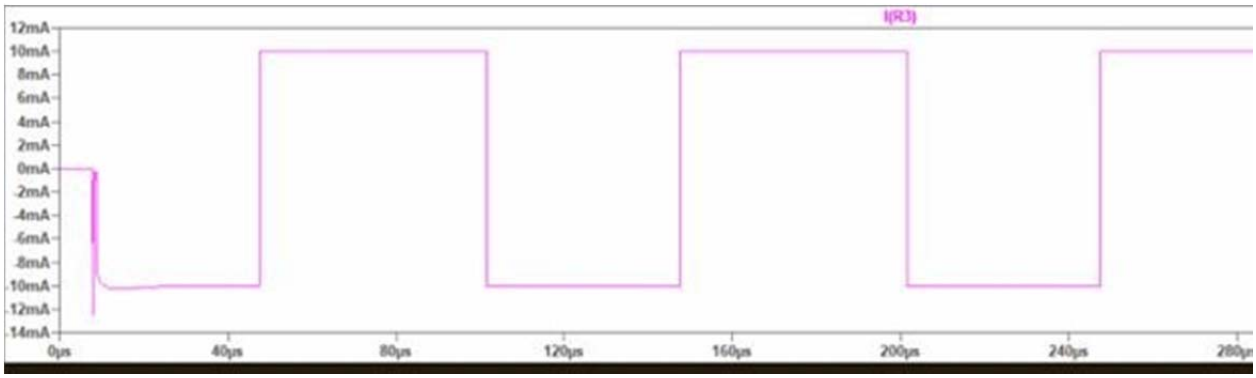

Рисунок 3.8. Графік струму, що протікає через резистор R3, що стоїть замість гальванічної ванночки в схемі симуляції.

З симуляції видно невеличкий спайк на початку форми який генерується в момент подачі живлення.. Згладжування струму в сторону форми постійного струму не спостерігається, форма прямокутних імпульсів чітка і має правильно форму.

3.4 Середовище розробки VHDPlus IDE

VHDPlus IDE є одним з найбільш новітніх середовищ розробки логічних систем на ПЛІС. Він поєднує в собі більшість тих можливостей, які пропонують середовища розробки, що монополізували сферу розробки пристроїв на ПЛІС останні десятиліття, таких як Intel Quartus Prime та Xilinx Vivado, і водночас з тим зручний інтерфейс та спрощену версію мови опису апаратури VHDP, що значно полегшує процес ознайомлення з цією сферою новачкам та любителям. Середовище використовує функціональні складові таких програмам як Quartus Prime та ModelSim задля надання користувачу потужних та надійних програмних інструментів в розробці систем на ПЛІС.

Попри можливість писати програми класичним мовами опису апаратури, такими як VHDL, Verilog та System Verilog, VHDPlus IDE підтримує мови C/C++ та пропонує завантажити бібліотеку Arduino, задля програмування процесора, синтезованого всередині ПЛІС. Однією з найбільш цікавих та унікальних характеристик цього програмного забезпечення є можливість легкого і швидкого додавання програмного процесора Nios II в свої проекти та робота з ним або на чистому C, або на C++, з можливістю використання бібліотек Arduino [24].

3.4.1 Nios II. Nios II - це програмований 32-бітний процесор архітектури RISC (Reduced Instruction Set Computing) [20], розроблений компанією Altera для використання в своїх ПЛІС. Nios II може бути імплементований в ПЛІС і використовуватися для вбудованого програмування, де програмний код збирається та завантажується безпосередньо в ПЛІС. Nios II є програмованим процесором, що дозволяє змінювати його конфігурацію та адаптувати під конкретні потреби. Процесор включає підтримку векторних операцій, що полегшує операції з масивами даних. Також в Nios II можна легко інтегрувати різноманітні порти вводу-виводу, таймери, UART, інтерфейси зовнішньої пам'яті та інші. Вікно конфігурації процесора в програмному забезпечені VHDPlus IDE показано на Рисунку 3.9.

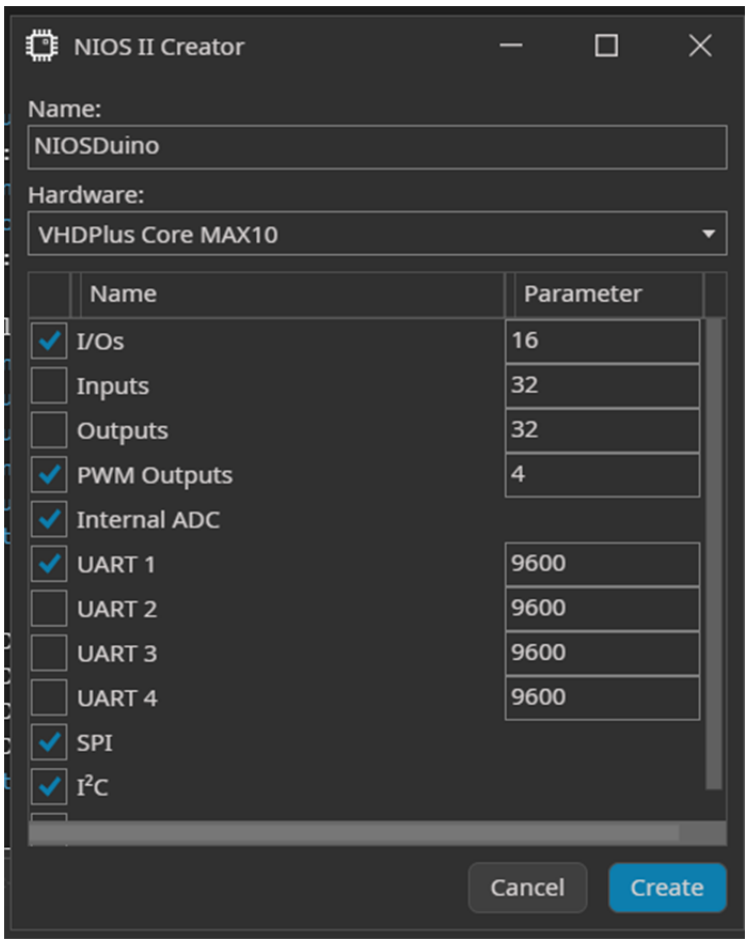

Рисунок 3.9. Вікно конфігурації NIOS II в VHDPlus IDE

Процесор може працювати з операційними системами вбудованого типу, такими як Micrium µC/OS-II, що робить його придатним для реалізації складних систем вбудованого програмування. Легкість інтегрування процесора в будь-який проект та можливість його реконфігурувати роблять NIOS II ідеальним для завдань, де критерії продуктивності, енергоефективності та гнучкості ставлять високі вимоги до програмованої конфігурації апаратних засобів.

3.4.2 Мова опису апаратури VHDP. Мову VHDP було створено на основі мови VHDL, хоча за структурою вона більше нагадує мову програмування високого рівня, оскільки включає опис апаратної логіки за допомогою високорівневих конструкцій. Основні конструкції мови включають модулі, сигнали, процеси і команди, які дозволяють описувати роботу апаратних компонентів. Основною одиницею в VHDP є модуль, який

описує окремий компонент апаратної системи. Мовою VHDP можна писати модулі паралельного, послідовного, та гібридного виконання. Окремої уваги заслуговує функція wait(), яка подібна за своєю роботою до функції delay() в Arduino та чекає вказану кількість циклів сигналу clock між двома операціями [25].

3.5 Розробка програми для мікроконтролера

В програмі мікроконтролера, що писалася на мові C++, використовуються бібліотеки Arduino, "OneButton" та "U8g2lib". Реалізація інтерфейсу у вигляді послідовних етапів меню вводу даних базується на використанні оператора switch (case) у всіх структурних складових програм

3.5.1 Бібліотека OneButton. Дана бібліотека була створена для полегшення використання однієї кнопки в якості багатофункціонального вводу в різноманітних проектах. Вона дозволяє легко реалізовувати різні події, такі як одиночне натискання, подвійне натискання і довге утримання кнопок приладу. Функції реакції на ці події є частиною цієї бібліотеки, все що треба зробити розробнику це обрати, які саме з доступних подій він хоче реалізувати в своєму проекті, та створити функцію, на яку буде посилатися функція реакції на події. Також варто не забувати використовувати в основній частині коду функцію опитування натискання кнопок, інакше програма не дізнається, що відбулося натискання кнопки [26].

3.5.2 Бібліотека U8g2lib. Бібліотека U8g2 є графічною бібліотекою для монохромних дисплеїв, розробленою для вбудованих систем. Вона підтримує широкий спектр OLED і LCD дисплеїв на драйверах, таких як SSD1306, SH1106, ST7565, і багато інших. Бібліотека складається з двох частин U8g2 і U8x8 та включає графічні процедури для малювання ліній, прямокутників, кіл, а також підтримує багато шрифтів. Бібліотека вимагає певного обсягу пам'яті мікроконтролера для рендерингу дисплея. При роботі з графічними об’єктами треба пам’ятати, що якщо на дисплеї

щось повинно, наприклад, перейти з одного кута дисплею в інший, то для подібної простої анімації треба спочатку видалити початкове положення об’єкту на дисплеї шляхом виводу на екран цього об’єкту «вимкненими» пікселями, і тільки після цього виводити його на дисплей в бажаному місці [27]

3.5.3 Структурна схема програми. Як зазначалося раніше, програма для мікроконтролеру переважно складається з детального опису кожного з запланованих значень оператора switch і подій всередині нього. На рисунку 3.10 зображено структурну схему написаної програми, де в прямокутних рамках позначені етапи, про які знає користувач; в кружках позначені етапи, які відбуваються всередині мікроконтролеру, та написами на біля стрілок вказано умову переходу у вигляді певного вводу, для переходу в той чи інший етап програми. Код програми для мікроконтролера наведено в Додатку А.

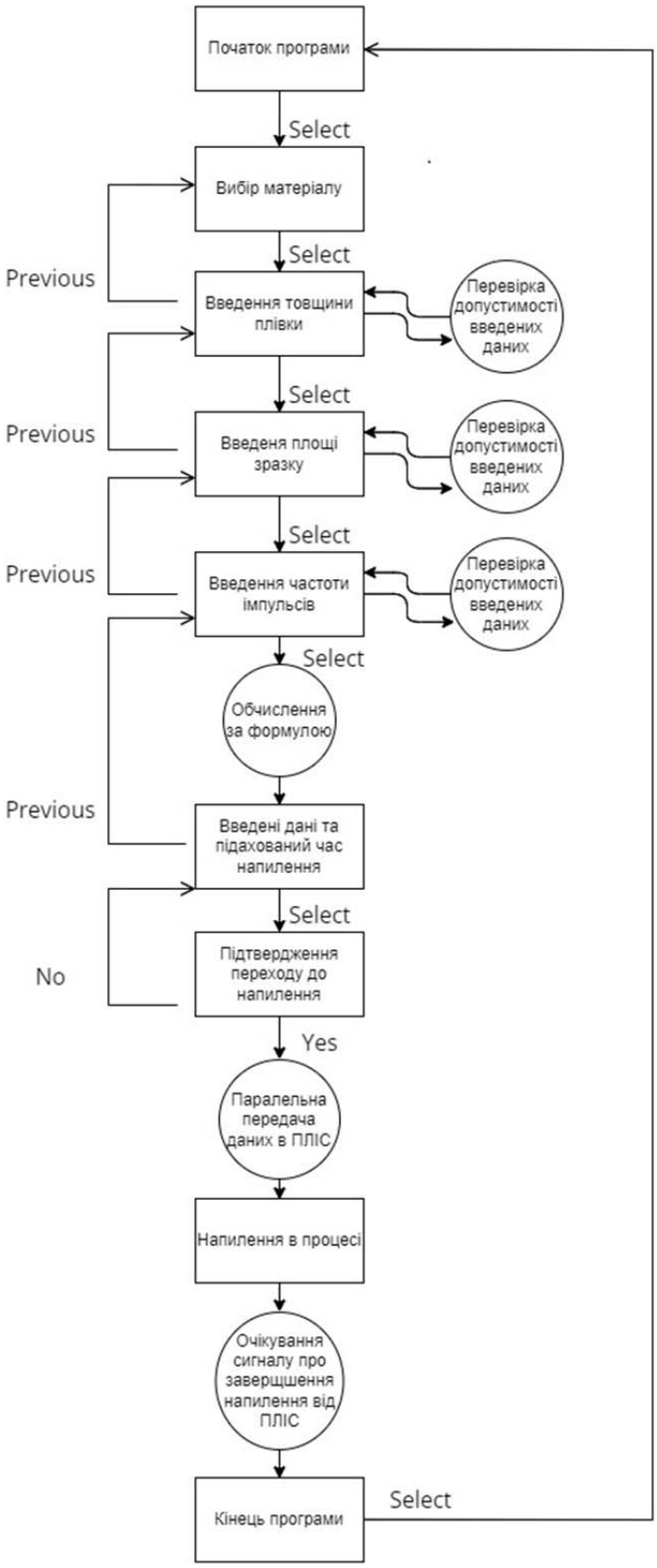

Рисунок 3.10. Структурна схема програми

3.5.4 Структурна схема взаємодії різних елементів програми. На Рисунку 3.11 зображено структурну схему взаємодії коду з бібліотек з основним кодом програми. В рамках з білим фоном вказані елементи основного коду, з жовтим – елементи графічної бібліотеки U8g2lib, з зеленим – бібліотеки роботи з кнопками OneButton.

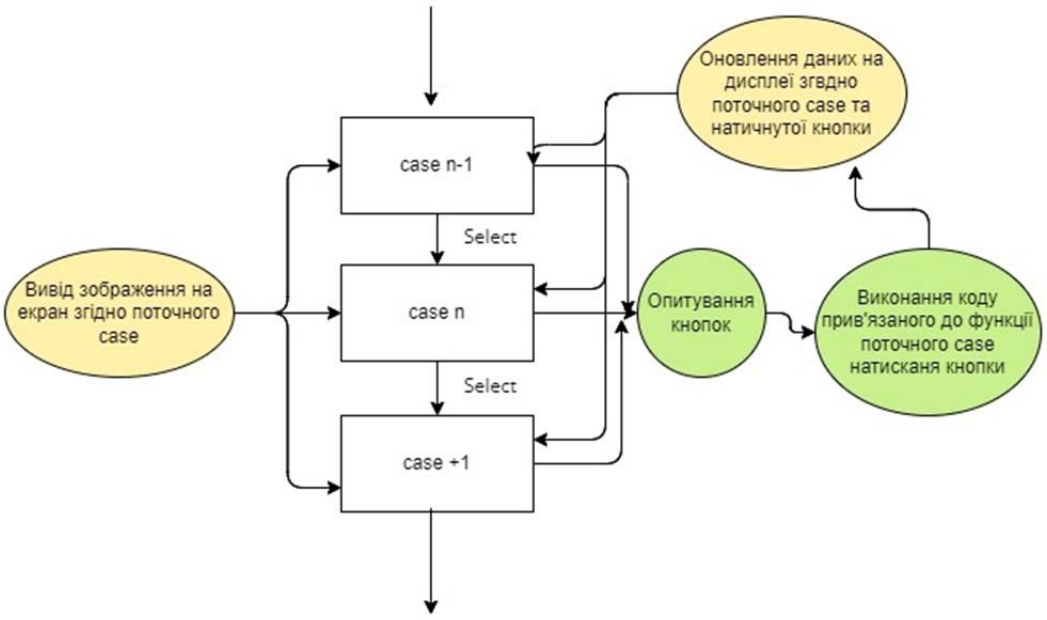

Рисунок 3.11. Структурна схема взаємодії різних елементів програми між собою

3.6 Розробка програми для ПЛІС

Код програми для ПЛІС складається з синтезованого програмного процесора NIOS II, в якому виконується код для мікроконтролера, і невеликого окремого процесу керування аналоговими компонентами приладу, написаному на мові VHDPlus. Процес очікує сигналу передачі керування ПЛІС від NIOS II, після чого паралельно зчитує значення матеріалу нарощування, часу напилення та частоти з ряду внутрішніх виводів

процесора. Після отримання даних програма рахує який опір треба виставити на цифрових потенціометрах, щоб співвідношення катодного струму до анодного становило 1:3, та виставляє значення резисторів 8-бітним значенням, яке передається в IC AD8402 по протоколу SPI. Керування SPDT перемикачами здійснюється завдяки гібридному написанню коду, в якому деякі операції виконуються паралельно, а деякі - послідовно. Також використовується вже згадана функція wait(). Структурну схему коду для ПЛІС зображено на Рисунку 3.12. Код програми для ПЛІС наведено в Додатку Б.

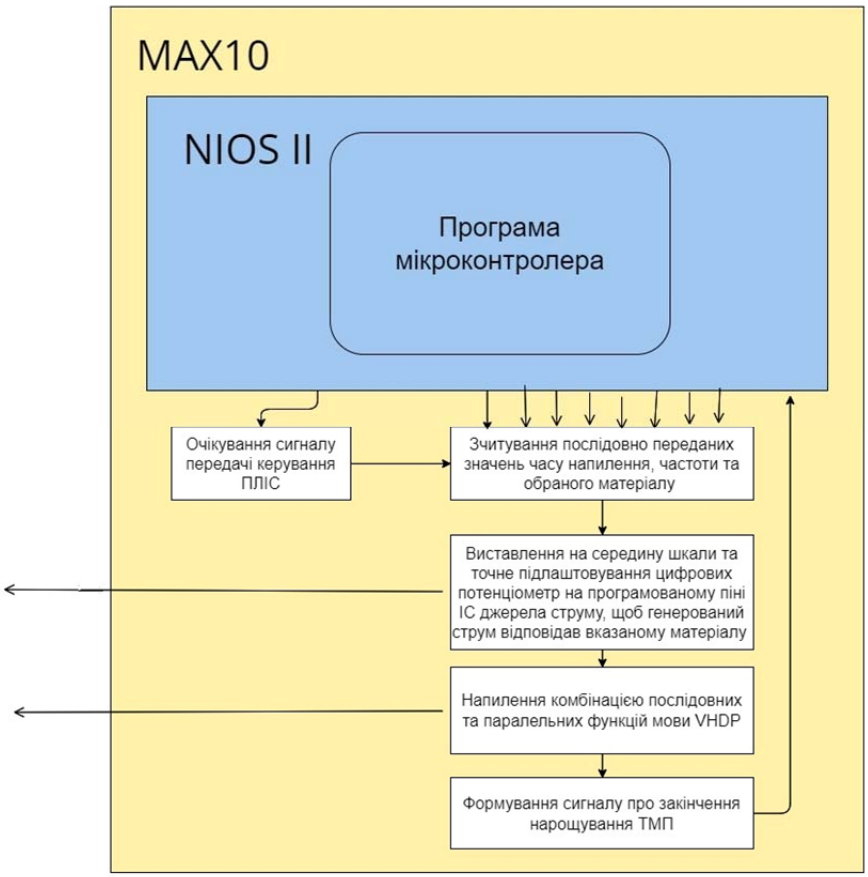

Рисунок 3.12. Структурна схема коду для ПЛІС

Висновки

Розглянуто вплив різних методів гальванічного нарощування тонких плівок на мікроструктуру і механічні властивості плівок, з чого зроблено висновок про створення пристрою для реалізації саме PRP методу.

Використання мостової схеми живлення гальванічної ванни дозволило реалізувати PRP метод в пристрої з однополярним живленням.

Використання ПЛІС MAX10 із вбудованим процесором дозволяє одночасно реалізувати переваги ПЛІС як пристрою реального часу і процесора як пристрою, орієнтованого на інтерфейс користувача.

Для досягнення найкращих результатів важливо вдосконалити пристрій для автоматичного балансування частоти з іншими параметрами імпульсу, що потребує подальших експериментальних досліджень з реальними зразками.

Список використаних джерел

1. Рогуль Т.Г. Металеві тонкі плівки. *Енциклопедія Сучасної України*. URL: https://esu.com.ua/article-66660 (дата звернення: 20.05.2024).
2. Солован М.М., Мостовий А.І.. Тонкоплівкова електроніка. Чернівці: Чернівецький нац. ун-т, 2021. 128 с.
3. Стасюк З.В., Лопатинський А.І. Розмірні кінетичні явища в тонких плівках металів. Класичні ефекти (огляд). *Фізика і хімія твердого тіла*. 2001. Т. 2, № 4. С. 521–542,
4. Калинушкін Є.П., Федоркова Н.М., Синиціна Ю.П. та ін. Тонкоплівкові матеріали та технології їх одержання: Навч. посібник. – Дніпропетровськ: НМетАУ, 2009. – 175 с.
5. Плачкова С., Плачков І. Книга 2. Пізнання й досвід – шлях до сучасної енергетики. *Енергетика: історія, сучасність і майбутнє*. 2012 URL: http://energetika.in.ua/ua/books/book-2/part-3/section-10/10-1 (дата звернення: 10.03.2024).
6. Шевчук М. В. Закони електролізу (закони Фарадея). Електронний посібник з дисципліни: Хімія. Луцьк, 2016. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/хымыя%20гот%20овий/page74.html (дата звернення: 08.12.2023).
7. Fuller T. F., Harb J. N. Electrochemical Engineering. Wiley & Sons, Incorporated, John, 2018. 448 с.
8. de Bruijn W. Throwing Power and Covering Power in Electroplating Solutions. *Transactions of the IMF*. 1950. Т. 27, № 1. С. 1–11. URL: https://doi.org/10.1080/00202967.1950.11869549 (дата звернення: 01.06.2024).
9. Mandich N. V. Pulse and pulse-reverse electroplating. *Metal Finishing*. 1999. Vol. 97, no. 1. P. 375–380. URL: https://doi.org/10.1016/s0026-0576(00)83097-4 (дата звернення: 04.04.2024).

10. Chandrasekar M. S., Pushpavanam M. Pulse and pulse reverse plating–Conceptual, advantages and applications. *Electrochimica Acta*. 2008. Т. 53, № 8. С. 3313–3322.
11. Royal Circuits Solutions. DC vs. Pulse Plating for Beginners. *Royal Circuits Solutions*. URL: https://www.royalcircuits.com/2019/01/15/dc-vs-pulse-plating-for-beginners/ (дата звернення: 26.05.2024).
12. Yung K.C.та ін. The effect of waveform for pulse plating on copper plating distribution of microvia in PCB manufacture . *The International Journal of Advanced Manufacturing Technology*. 2004. Т. 23, № 3-4. С. 245–248. URL: https://doi.org/10.1007/s00170-003-1667-1 (дата звернення: 01.06.2024).
13. Novel Metal Electrodeposition and the Recent Application / ред.: M. Sone, K. Masu. IntechOpen, 2019. URL: https://doi.org/10.5772/intechopen.73382 (дата звернення: 01.06.2024).
14. Kalaivani V., Shanthi C. Pulse Reverse Plating of Silver on Silver Alloy. *Protection of Metals and Physical Chemistry of Surfaces*. 2018. Vol. 54, no. 4. P. 668–672. URL: https://doi.org/10.1134/s2070205118040202 (дата звернення: 04.04.2024).
15. Sharretts Plating Company. The Effects of Periodic Reverse Current in Electroplating. *Sharretts Plating Company*. URL: https://www.sharrettsplating.com/blog/effects-periodic-reverse-current-electroplating/ (дата звернення: 20.05.2024).
16. Kamaraj A., Reed N., Sundaram M. Effect of Ultra-High Pulse Frequency on the resolution in the Electrochemical Deposition of Nickel. *Procedia Manufacturing*. 2021. Т. 53. С. 59–63. URL: https://doi.org/10.1016/j.promfg.2021.06.010 (дата звернення: 01.06.2024).
17. Zhao Y. та ін Influence of Pulse Current Forward-Reverse Duty Cycle on Structure and Performance of Electroplated W–Cu Composite Coatings/ *Materials*. 2021. Т. 14, № 5. С. 1233.

18. Kehong Equipment. Types of Electroplating Power Supply. *Medium*. URL: https://medium.com/@kehonghaman/types-of-electroplating-power-supply-ea75d1bf7443 (дата звернення: 01.06.2024).
19. Beier L. MAX1000. *VHDPlus*. URL: https://vhdplus.com/docs/components/max1000/ (дата звернення: 01.06.2024).
20. Arrow. MAX1000 User Guide. *Arrow*. URL: https://www.arrow.com/en/products/max1000/arrow-development-tools (дата звернення:: 07.03.2024).
21. Analog Devices. LT3092 Datasheet and Product Info. *Analog Devices*. URL: https://www.analog.com/en/products/lt3092.html (дата звернення: 02.04.2024).
22. Analog Devices. ADG1636 Datasheet and Product Info. *Analog Devices*. URL: https://www.analog.com/en/products/lt3092.html (дата звернення: 02.04.2024).
23. Analog Devices. AD8402 Datasheet and Product Info. *Analog Devices*. URL: https://www.analog.com/en/products/lt3092.html (дата звернення: 02.04.2024).
24. Mennen H. Program Software on your FPGA. *VHDPlus*. URL: https://vhdplus.com/docs/guides/nios2/ (дата звернення: 01.06.2024).
25. Beier L. Wait. *VHDPlus*. URL: https://vhdplus.com/docs/vhdp/procedureoperations/wait/ (дата звернення: 01.06.2024).
26. Hertel M. Arduino OneButton Library. *GitHub*. URL: https://github.com/mathertel/OneButton (дата звернення: 28.04.2024).
27. Oliver. U8g2_Arduino: Arduino Monochrome Graphics Library. *GitHub*. URL: https://github.com/olikraus/U8g2_Arduino (дата звернення: 28.04.2024).

Додаток А

Програма для процесору NIOS II написана на мові C++

```
#include <Arduino.h>
#include <OneButton.h>
#include <U8g2lib.h>
#include <Wire.h>

U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

//-----------------------MENU STAGES AND VARIABLES-----------------------------
----------
int input_stage;
int menu_option;
int material;
unsigned int thickness;
unsigned int area;
unsigned int frequency;
int result;
int previous_menu_option;
int previous_num;

String material_list[] = {"Ag", "Cu", "Au", "Ni"};
int digit_coordinate[] = {34, 46, 70, 83};
bool select_pressed = false;
bool input_from_fpga = false;
String curr_str;
String materials;
String prev_str;
```

```
//------------------------------FUNCTIONS-------------------------------------

void clickIncrease();
void longPressIncrease();
void clickDecrease();
void longPressDecrease();
void clickNext();
void clickBefore();
void clickSelect();
//button form for testing
void clickFPGA();

void ifNumOverLimit(unsigned int& n);
void ifNumBelowLimit(unsigned int& n);
void numIncrease(unsigned int& n);
void numDecrease(unsigned int& n);
void menuNext(int max_option);
void menuPrevious(int min_option);
void setVariablesToZero();

//--------------------------DISPLAY FUNCTIONS-----------------------------------
void drawMenuStage();
void invertSelect();
void invertPrevious();
void drawFourDigits(int num);
void drawFrequency();
void updateActiveOption();
```

```
//----------------------------BUTTONS-----------------------------------
//Pin on the right of the module is gnd
#define pin_select 0 //green, second from the right
#define pin_decrease 1
#define pin_increase 2
#define pin_before 3
#define pin_next 4 // brown, first from the left
#define pin_input_from_fpga 5 // brown, first from the left
#define led_red 6 //
#define led_green 7 //

OneButton button_select(pin_select, true);
OneButton button_decrease(pin_decrease, true);
OneButton button_increase(pin_increase, true);
OneButton button_before(pin_before, true);
OneButton button_next(pin_next, true);
//button form for testing
OneButton button_fpga(pin_input_from_fpga, true);

void setup() {

  //serial
  Serial.begin(9600);
  Serial.println("--------Start--------");
  input_stage = 8;
  //display
  u8g2.begin();
  u8g2.setContrast(15);
  //leds
  pinMode(led_red, OUTPUT);
```

```
    pinMode(led_green, OUTPUT);
    //button functions
    button_increase.attachClick(clickIncrease);
    button_increase.attachDuringLongPress(longPressIncrease);
    button_decrease.attachClick(clickDecrease);
    button_decrease.attachDuringLongPress(longPressDecrease);
    button_next.attachClick(clickNext);
    button_before.attachClick(clickBefore);
    button_select.attachClick(clickSelect);
    //button form for testing
    button_fpga.attachClick(clickFPGA);

    button_increase.setLongPressIntervalMs(300);
    button_decrease.setLongPressIntervalMs(300);
}

void loop() {

    if (input_stage == 5) {
        digitalWrite(led_green, LOW);
        digitalWrite(led_red, HIGH);
    }
    else {
        digitalWrite(led_red, LOW);
        digitalWrite(led_green, HIGH);
    }
    //test case
    switch (input_stage) {
            case 0: Serial.println("-----Choose material: ");
                    menu_option = material;
```

```
        previous_menu_option = material;
        drawMenuStage();
        while (1) {
          button_next.tick();
          button_before.tick();
          button_select.tick();

          if (select_pressed == true) {
            material = menu_option;
            Serial.println("Material = ");
            Serial.println(material);
            input_stage++;
            select_pressed = false;
            invertSelect();
            break;
          }
        }
        menu_option = 0;
        previous_menu_option = 0;
        delay(10);
        break;

case 1: Serial.println("-----Choose thickness: ");
        drawMenuStage();
        while (1) {
          button_next.tick();
          button_before.tick();
          button_increase.tick();
          button_decrease.tick();
          button_select.tick();
```

```
            if (select_pressed == true) {
                if (menu_option == -1) {
                    select_pressed = false;
                    input_stage--;
                    invertPrevious();
                    break;
                }
                else {
                    Serial.println("Thickness = ");
                    Serial.println(thickness);
                    select_pressed = false;
                    input_stage++;
                    invertSelect();
                    break;
                }
            }
        }
        menu_option = 0;
        previous_menu_option = 0;
        delay(10);
        break;

case 2: Serial.println("-----Choose area");
        drawMenuStage();
        while (1) {
            button_next.tick();
            button_before.tick();
            button_increase.tick();
            button_decrease.tick();
```

```
            button_select.tick();

            if (select_pressed == true) {
               if (menu_option == -1) {
                  select_pressed = false;
                  input_stage--;
                  invertPrevious();
                  break;
               }
               else {
                  Serial.println("Area = ");
                  Serial.println(area);
                  select_pressed = false;
                  input_stage++;
                  invertSelect();
                  break;
               }
            }
         }
         menu_option = 0;
         previous_menu_option = 0;
         delay(10);
         break;

case 3: Serial.println("-----Choose frequency");
         drawMenuStage();
         while (1) {
            button_next.tick();
            button_before.tick();
            button_increase.tick();
```

```
            button_decrease.tick();
            button_select.tick();

            if (select_pressed == true) {
               if (menu_option == -1) {
                  select_pressed = false;
                  input_stage--;
                  invertPrevious();
                  break;
               }
               else {
                  Serial.println("Freq = ");
                  Serial.println(frequency);
                  select_pressed = false;
                  input_stage++;
                  invertSelect();
                  break;
               }
            }
         }
         menu_option = 0;
         previous_menu_option = 0;
         delay(10);
         break;

case 4: result = material + thickness; // for testing purposes
         drawMenuStage();

         Serial.println("-------Results:");
         Serial.println("Material: ");
```

```
Serial.println(material);
Serial.println("Thickness: ");
Serial.println(float(thickness)/100);
Serial.println("Area: ");
Serial.println(float(area)/100);
Serial.println("Frequency: ");
Serial.println(frequency);
Serial.println("Calculated time: ");
Serial.println(result);

while (1) {
   button_next.tick();
   button_before.tick();
   button_select.tick();

   if (select_pressed == true) {
      if (menu_option == -1) {
         select_pressed = false;
         input_stage--;
         invertPrevious();
         break;
      }
      else {
         select_pressed = false;
         input_stage++;
         invertSelect();
         break;
      }
   }
}
```

```
        menu_option = 0;
        previous_menu_option = 0;
        delay(10);
        break;

case 5: Serial.println("-----Are you sure?");
        drawMenuStage();
        while (1) {
           button_next.tick();
           button_before.tick();
           button_select.tick();

           if (select_pressed == true) {
              if (menu_option == 0) {
                 select_pressed = false;
                 input_stage--;
                 break;
              }
              else {
                 select_pressed = false;
                 input_stage++;
                 break;
              }
           }
        }
        menu_option = 0;
        previous_menu_option = 0;
        delay(10);
        break;
```

```
        case 6: Serial.println("!!!!Start procces!!!!");
                drawMenuStage();
                while(1) {
                   button_fpga.tick();
                   if (input_from_fpga == true) {
                      input_from_fpga = false;
                      input_stage++;
                      break;
                   }
                   delay(50);
                }
                break;

        case 7: Serial.println("-----FINISHED");
                drawMenuStage();
                while (1) {
                   button_select.tick();

                   if (select_pressed == true) {
                      select_pressed = false;
                      input_stage = 0;
                      u8g2.drawButtonUTF8(98,  59,  U8G2_BTN_INV,  0,    4,    1,
"Start" );
                      u8g2.updateDisplay();
                      delay(100);
                      break;
                   }
                }
                setVariablesToZero();
                delay(10);
```

```
                break;

        case 8: Serial.println("-----Press SELECT to start");
                drawMenuStage();
                while (1) {
                  button_select.tick();

                  if (select_pressed == true) {
                    select_pressed = false;
                    input_stage = 0;
                    u8g2.drawButtonUTF8(98, 59, U8G2_BTN_INV, 0,  4,  1,
"Start" );
                    u8g2.updateDisplay();
                    delay(100);
                    break;
                  }
                }
                setVariablesToZero();
                delay(10);
                break;

        default: Serial.println("Out of case bound");
                 input_stage = 0;
                 break;
    }
    delay(10);
}

void clickIncrease() {
    switch (input_stage) {
```

```
        //case 0: return 1;
        case 1: numIncrease(thickness);
                Serial.println(thickness);
                drawFourDigits(thickness);
                break;

        case 2: numIncrease(area);
                Serial.println(area);
                drawFourDigits(area);
                break;

        case 3: numIncrease(frequency);
                Serial.println(frequency);
                drawFourDigits(frequency);
                break;
    }
    updateActiveOption();
}

void clickDecrease() {
    switch (input_stage) {
        //case 0: return 0;
        case 1: numDecrease(thickness);
                Serial.println(thickness);
                drawFourDigits(thickness);
                break;

        case 2: numDecrease(area);
                Serial.println(area);
                drawFourDigits(area);
```

```
            break;

        case 3: numDecrease(frequency);
            Serial.println(frequency);
            drawFourDigits(frequency);
            break;
    }
    updateActiveOption();
}

void clickNext() {
    switch (input_stage) {
        // scrolling through material options
        case 0: if (menu_option < 3) {
                menu_option++;
                Serial.println(menu_option);
                material = menu_option;
            }
            break;
        // previous step + 4 digits
        case 1: menuNext(3);
            break;
        // previous step + 4 digits
        case 2: menuNext(3);
            break;
        // previous step + 4 digits
        case 3: menuNext(3);
            break;
        // previous step + OK
        case 4: menuNext(0);
```

```
            break;
        // previous step + NO/YES
        case 5: menuNext(1);
            break;
    }
    updateActiveOption();
}

void clickBefore() {
    switch (input_stage) {
        // scrolling through material options
        case 0: if (menu_option > 0) {
                menu_option--;
                Serial.println(menu_option);
                material--;
            }
            break;
        // previous step + 4 digits
        case 1: menuPrevious(-1);
            break;
        // previous step + 4 digits
        case 2: menuPrevious(-1);
            break;
        // previous step + 4 digits
        case 3: menuPrevious(-1);
            break;
        // previous step + OK
        case 4: menuPrevious(-1);
            break;
        // previous step + NO/YES
```

```
            case 5: menuPrevious(0);
                    break;
    }
    updateActiveOption();
}

void clickSelect() {
    select_pressed = true;
}

void clickFPGA() {
    input_from_fpga = true;
}

void longPressIncrease() {
    switch (input_stage) {
            //case 0: return 1;
            case 1: numIncrease(thickness);
                    Serial.println(thickness);
                    drawFourDigits(thickness);
                    break;

            case 2: numIncrease(area);
                    Serial.println(area);
                    drawFourDigits(area);
                    break;

            case 3: numIncrease(frequency);
                    Serial.println(frequency);
                    drawFrequency();
```

```
            break;
    }
    updateActiveOption();
}

void longPressDecrease() {
    switch (input_stage) {
        //case 0: return 0;
        case 1: numDecrease(thickness);
            Serial.println(thickness);
            drawFourDigits(thickness);
            break;

        case 2: numDecrease(area);
            Serial.println(area);
            drawFourDigits(area);
            break;

        case 3: numDecrease(frequency);
            Serial.println(frequency);
            drawFrequency();
            break;
    }
    updateActiveOption();
}

void numIncrease(unsigned int& n) {
    switch (menu_option) {
        case 0: n = n + 1000;
            ifNumOverLimit(n);
```

```
            break;

        case 1: n = n + 100;
            ifNumOverLimit(n);
            break;

        case 2: n = n + 10;
            ifNumOverLimit(n);
            break;

        case 3: n = n + 1;
            ifNumOverLimit(n);
            break;
    }
}

void numDecrease(unsigned int& n) {
    switch (menu_option) {
        case 0: n = n - 1000;
            ifNumBelowLimit(n);
            break;

        case 1: n = n - 100;
            ifNumBelowLimit(n);
            break;

        case 2: n = n - 10;
            ifNumBelowLimit(n);
            break;
```

```
            case 3: n = n - 1;
                    ifNumBelowLimit(n);
                    break;
    }
}

void ifNumOverLimit(unsigned int& n) {
    if (n > 9999) {
        n = 0;
    }
}

void ifNumBelowLimit(unsigned int& n) {
    if (n > 9999) {
        n = 9999;
    }
}

void menuNext(int max_option) {
    if (menu_option < max_option) {
        menu_option++;
        Serial.println(menu_option);
    }
}

void menuPrevious(int min_option) {
    if (menu_option > min_option) {
        menu_option--;
        Serial.println(menu_option);
    }
```

```
}

void setVariablesToZero() {
   input_stage = 0;
   menu_option = 0;
   previous_menu_option = 0;
   material = 0;
   thickness = 0;
   area = 0;
   frequency = 0;
   result = 0;
   previous_num = 0;
   curr_str = "";
}

void drawMenuStage() {
   u8g2.clearBuffer();
   switch (input_stage) {
         case 0: u8g2.setFont(u8g2_font_nokiafc22_tf          );
               u8g2.drawStr(20,10,"Choose  material");
               u8g2.drawHLine(0, 12, 124);
               u8g2.setFont(u8g2_font_ncenR12_tr          );
               materials = material_list[0] + "   " + material_list[1] + "   " +
material_list[2] + "  " + material_list[3];
               u8g2.drawStr(10,40,materials.c_str());
               u8g2.drawRFrame(material*29 + 8,25,26,21,3);
               u8g2.setFont(u8g2_font_5x7_tf          );
               u8g2.drawButtonUTF8(102, 59, U8G2_BTN_BW1, 0,  4,  1, "Next" );
               u8g2.drawButtonUTF8(56, 59, U8G2_BTN_BW0, 0,  0,  1, "1/5" );
               break;
```

```
case 1: u8g2.setFont(u8g2_font_nokiafc22_tf              );
        u8g2.drawStr(23,10,"Enter  thickness");
        u8g2.drawHLine(0, 12, 124);
        u8g2.setFont(u8g2_font_profont22_tf             );
        previous_num = thickness;
        drawFourDigits(thickness);
        u8g2.setFont(u8g2_font_DigitalDisco_tf          );
        u8g2.drawStr(8,40,"T:");
        u8g2.setFont(u8g2_font_nokiafc22_tf          );
        u8g2.drawStr(100,40,"mkm");
        u8g2.drawHLine(digit_coordinate[menu_option], 23, 12);
        u8g2.drawHLine(digit_coordinate[menu_option], 43, 12);
        u8g2.setFont(u8g2_font_5x7_tf               );
        u8g2.drawButtonUTF8(102, 59, U8G2_BTN_BW1, 0,  4,  1, "Next" );
        u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
        u8g2.drawButtonUTF8(56, 59, U8G2_BTN_BW0, 0,  0,  1, "2/5" );
        break;

case 2: u8g2.setFont(u8g2_font_nokiafc22_tf          );
        u8g2.drawStr(35,10,"Enter  area");
        u8g2.drawHLine(0, 12, 124);
        u8g2.setFont(u8g2_font_profont22_tf             );
        previous_num = area;
        drawFourDigits(area);
        u8g2.setFont(u8g2_font_DigitalDisco_tf          );
        u8g2.drawStr(8,40,"A:");
        u8g2.setFont(u8g2_font_nokiafc22_tf          );
        u8g2.drawStr(100,40,"mm2");
        u8g2.drawHLine(digit_coordinate[menu_option], 23, 12);
```

```
        u8g2.drawHLine(digit_coordinate[menu_option], 43, 12);
        u8g2.setFont(u8g2_font_5x7_tf              );
        u8g2.drawButtonUTF8(102, 59, U8G2_BTN_BW1, 0,  4,  1, "Next" );
        u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
        u8g2.drawButtonUTF8(56, 59, U8G2_BTN_BW0, 0,  0,  1, "3/5" );
        break;

case 3: u8g2.setFont(u8g2_font_nokiafc22_tf          );
        u8g2.drawStr(30,10,"Enter  frequency");
        u8g2.drawHLine(0, 12, 124);
        u8g2.setFont(u8g2_font_profont22_tf            );
        previous_num = frequency;
        drawFrequency();
        u8g2.setFont(u8g2_font_DigitalDisco_tf         );
        u8g2.drawStr(8,40,"f:");
        u8g2.setFont(u8g2_font_nokiafc22_tf          );
        u8g2.drawStr(100,40,"Hz");
        u8g2.drawHLine(digit_coordinate[menu_option], 23, 12);
        u8g2.drawHLine(digit_coordinate[menu_option], 43, 12);
        u8g2.setFont(u8g2_font_5x7_tf              );
        u8g2.drawButtonUTF8(102, 59, U8G2_BTN_BW1, 0,  4,  1, "Next" );
        u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
        u8g2.drawButtonUTF8(56, 59, U8G2_BTN_BW0, 0,  0,  1, "4/5" );
        break;

case 4: u8g2.clearBuffer();
        u8g2.setFont(u8g2_font_nokiafc22_tf            );
        u8g2.drawStr(35,10,"Your input");
        u8g2.drawHLine(0, 12, 124);
        u8g2.setFont(u8g2_font_5x7_tf              );
```

```
        u8g2.drawStr(5,20,"Material");
        curr_str = material_list[material];
        u8g2.drawStr(70,20,curr_str.c_str());
        u8g2.drawStr(4,27,"Thickness");
        curr_str = String(float(thickness)/100, 2);
        u8g2.drawStr(70,27,curr_str.c_str());
        u8g2.drawStr(5,34,"Area");
        u8g2.drawStr(70,34,curr_str.c_str());
        curr_str = String(float(area)/100, 2);
        u8g2.drawStr(5,41,"Frequency");
        curr_str = String(frequency);
        u8g2.drawStr(70,41,curr_str.c_str());
        u8g2.drawStr(5,48,"Calc. time");
        u8g2.drawButtonUTF8(102, 59, U8G2_BTN_BW1, 0,  4,  1, "Next" );
        u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
        u8g2.drawButtonUTF8(56, 60, U8G2_BTN_BW0, 0,  0,  1, "5/5" );
        u8g2.sendBuffer();
        break;

case 5: u8g2.clearBuffer();
        u8g2.setFont(u8g2_font_nokiafc22_tf              );
        u8g2.drawStr(27,20,"Are you sure?");
        u8g2.drawFrame(10,8,104,46);
        u8g2.setFont(u8g2_font_open_iconic_check_2x_t            );
        u8g2.drawGlyph(30,46,68);
        u8g2.drawRFrame(28,27,20,20, 3);
        u8g2.drawGlyph(75,46,64);
        u8g2.sendBuffer();
        break;
```

```
        case 6: u8g2.clearBuffer();
                u8g2.setFont(u8g2_font_nokiafc22_tf          );
                u8g2.drawStr(35,35,"*In Progres*");
                u8g2.sendBuffer();
                break;

        case 7: u8g2.clearBuffer();
                u8g2.setFont(u8g2_font_nokiafc22_tf          );
                u8g2.drawStr(40,35,"FINISHED");
                u8g2.drawButtonUTF8(98, 59, U8G2_BTN_BW1, 0,  4,  1, "Start" );
                u8g2.sendBuffer();
                break;

        case 8: u8g2.clearBuffer();
                u8g2.setFont(u8g2_font_nokiafc22_tf          );
                u8g2.drawStr(30,25,"Press SELECT");
                u8g2.drawStr(15,40,"to start the program");
                u8g2.drawButtonUTF8(98, 59, U8G2_BTN_BW1, 0,  4,  1, "Start" );
                u8g2.sendBuffer();
                break;
    }
    u8g2.sendBuffer();
}

void invertSelect(){
    u8g2.drawButtonUTF8(102, 59, U8G2_BTN_INV, 0,  4,  1, "Next" );
    u8g2.updateDisplay();
    delay(100);
}
```

```
void invertPrevious() {
   u8g2.drawButtonUTF8(5, 59, U8G2_BTN_INV, 0,  4,  1, "Prev" );
   u8g2.updateDisplay();
   delay(100);
}

void drawFourDigits(int num) {
   u8g2.setFont(u8g2_font_profont22_tf              );
   u8g2.setDrawColor(0);
   prev_str = String(float(previous_num)/100, 2);

   if (previous_num < 1000) {
      prev_str = "0" + prev_str;
   }

   if ((previous_num % 100 == 0) && (previous_num % 1000 == 0)) {
      prev_str = prev_str + ".0";
   }

   if (previous_num % 1000 == 0) {
      prev_str = prev_str + "0";
   }

   u8g2.drawStr(35,40, prev_str.c_str());
   curr_str = String(float(num)/100, 2);
   u8g2.setDrawColor(1);

   if (num < 1000) {
      curr_str = "0" + curr_str;
   }
```

```
    u8g2.drawStr(35,40, curr_str.c_str());
    u8g2.updateDisplay();
    previous_num = num;
}

void drawFrequency() {
    u8g2.setFont(u8g2_font_profont22_tf          );
    u8g2.setDrawColor(0);
    prev_str = String(frequency);

    if (previous_num < 1000) {
        prev_str = "0" + prev_str;
    }

    if (previous_num < 100) {
        prev_str = "0" + prev_str;
    }

    if (previous_num < 10) {
        prev_str ="0" + prev_str;
    }

    u8g2.drawStr(35,40, prev_str.c_str());
    curr_str = String(frequency);
    u8g2.setDrawColor(1);

    if (frequency < 1000) {
        curr_str = "0" + curr_str;
    }
    u8g2.drawStr(35,40, curr_str.c_str());
```

```
    u8g2.updateDisplay();
    previous_num = frequency;
}

void updateActiveOption() {
    u8g2.setFont(u8g2_font_5x7_tf   );
    switch (input_stage) {
        case 0: u8g2.setDrawColor(0);
            u8g2.drawRFrame((previous_menu_option*29 + 8),25,26,21,3);
            u8g2.setDrawColor(1);
            u8g2.drawRFrame((menu_option*29 + 8),25,26,21,3);
            previous_menu_option = menu_option;
            u8g2.updateDisplay();
            break;

        case 1: if (menu_option >= 0) {
                u8g2.setDrawColor(0);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 23, 12);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 43, 12);
                u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                u8g2.setDrawColor(1);
                u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                u8g2.drawHLine(digit_coordinate[menu_option], 23, 12);
                u8g2.drawHLine(digit_coordinate[menu_option], 43, 12);
                previous_menu_option = menu_option;
            }
            else {
                u8g2.setDrawColor(0);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 23, 12);
```

```
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 43, 12);
                u8g2.setDrawColor(1);
                u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
            }
            u8g2.updateDisplay();
            break;

        case 2: if (menu_option >= 0) {
                u8g2.setDrawColor(0);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 23, 12);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 43, 12);
                u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                u8g2.setDrawColor(1);
                u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                u8g2.drawHLine(digit_coordinate[menu_option], 23, 12);
                u8g2.drawHLine(digit_coordinate[menu_option], 43, 12);
                previous_menu_option = menu_option;
            }
            else {
                u8g2.setDrawColor(0);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 23, 12);
                u8g2.drawHLine(digit_coordinate[previous_menu_option], 43, 12);
                u8g2.setDrawColor(1);
                u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
            }
            u8g2.updateDisplay();
            break;
```

```
        case 3: if (menu_option >= 0) {
                    u8g2.setDrawColor(0);
                    u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                    u8g2.setDrawColor(1);
                    u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                }
                else {
                    u8g2.setDrawColor(1);
                    u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                }
                u8g2.updateDisplay();
                break;

        case 4: if (menu_option >= 0) {
                    u8g2.setDrawColor(0);
                    u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                    u8g2.setDrawColor(1);
                    u8g2.drawButtonUTF8(5, 59, U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                }
                else {
                    u8g2.setDrawColor(1);
                    u8g2.drawButtonUTF8(5,                                    59,
U8G2_BTN_SHADOW1|U8G2_BTN_BW1, 0,  4,  1, "Prev" );
                }
                u8g2.updateDisplay();
                break;
```

```
        case 5: if (menu_option > 0) {
                    u8g2.setDrawColor(0);
                    u8g2.drawRFrame(28,27,20,20, 3);
                    u8g2.setDrawColor(1);
                    u8g2.drawRFrame(73,27,20,20, 3);
                }
                else {
                    u8g2.setDrawColor(0);
                    u8g2.drawRFrame(73,27,20,20, 3);
                    u8g2.setDrawColor(1);
                    u8g2.drawRFrame(28,27,20,20, 3);
                }
                u8g2.updateDisplay();
                break;

        case 7: u8g2.drawButtonUTF8(98, 60, U8G2_BTN_INV, 0,  4,  1, "Start"
);
                u8g2.updateDisplay();
                break;
    }
}
```

Додаток Б

Програма для ПЛІС написана на мові VHDP

```
Main
(
   Reset: in STD_LOGIC := '1';
   sdram_addr: out STD_LOGIC_VECTOR(11 downto 0);
   sdram_ba: out STD_LOGIC_VECTOR(1 downto 0);
   sdram_cas_n: out STD_LOGIC;
   sdram_cke: out STD_LOGIC;
   sdram_cs_n: out STD_LOGIC;
   sdram_dq: inout STD_LOGIC_VECTOR(15 downto 0) := (others => 'X');
   sdram_dqm: out STD_LOGIC_VECTOR(1 downto 0);
   sdram_ras_n: out STD_LOGIC;
   sdram_we_n: out STD_LOGIC;
   sdram_clk_clk: out STD_LOGIC;
   pio_export: inout STD_LOGIC_VECTOR(19 downto 0) := (others => 'X');
   i2c_sda: inout STD_LOGIC := 'Z';
   i2c_scl: inout STD_LOGIC := 'Z';

   prog_switch: out STD_LOGIC_VECTOR(2 downto 0);
   control_back: out STD_LOGIC := 0;
   MISO: in STD_LOGIC := '0';
   SCLK: buffer STD_LOGIC := '0';
   SS: buffer STD_LOGIC := '1';
   MOSI: out STD_LOGIC := '0';

)
{
   Process
   (
```

```
    VARIABLE param : STD_LOGIC_VECTOR (10 downto 0);
    VARIABLE count : NATURAL range 0 to 5000 := '0';
    VARIABLE flag: STD_LOGIC := '0';
)
{
    If (pio_export(19) = '1' AND flag = '0')
    {
        param(10 downto 0) := pio_export(18 downto 8);
        flag := '1';
    }

    If (pio_export(19) = '1' AND flag = '1')
    {
        Thread
        {
            avalon_pwm_pwm_out <= "500";
            prog_switch(0) <= '1';
            Step {prog_switch(1) <= '1'}
        }

    }
}

Process()
{
    Thread
    {
        RW <= '0';
        Address <= "00";
        Data_OUT <= "00111010";
```

```
        SPI_Master_Enable <= '1';
        Step { SPI_Master_Enable <= '0'; }
        While(SPI_Master_Busy = '1') {}

    }
}

SIGNAL RW   : STD_LOGIC := '0';
SIGNAL Address : STD_LOGIC_VECTOR(1 downto 0);
SIGNAL Data_IN : STD_LOGIC_VECTOR(7 downto 0);
SIGNAL Data_OUT : STD_LOGIC_VECTOR(7 downto 0);
SPI_Master_TX_Data(15) <= RW;
SPI_Master_TX_Data(14) <= '0';
SPI_Master_TX_Data(13 downto 8) <= Address;
SPI_Master_TX_Data(7 downto 0) <= Data_OUT;


CONSTANT SPI_Master_Bits: NATURAL := 8;
SIGNAL SPI_Master_Reset: STD_LOGIC := '0';
SIGNAL SPI_Master_TX_Data: STD_LOGIC_VECTOR(SPI_Master_Bits - 1
downto 0) := (others => '0');
SIGNAL SPI_Master_Enable: STD_LOGIC := '0';
SIGNAL SPI_Master_Cont: STD_LOGIC := '0';
SIGNAL SPI_Master_Busy: STD_LOGIC := '0';
SIGNAL SPI_Master_RX_Data: STD_LOGIC_VECTOR(SPI_Master_Bits - 1
downto 0) := (others => '0');
SIGNAL avalon_pwm_clk: STD_LOGIC;
SIGNAL avalon_pwm_reset_n: STD_LOGIC;
SIGNAL avalon_pwm_chipselect: STD_LOGIC;
SIGNAL avalon_pwm_address: STD_LOGIC_VECTOR;
```

```
    SIGNAL avalon_pwm_write: STD_LOGIC;
    SIGNAL avalon_pwm_writedata: STD_LOGIC_VECTOR;
    SIGNAL avalon_pwm_read: STD_LOGIC;
    SIGNAL avalon_pwm_readdata: STD_LOGIC_VECTOR;
    SIGNAL avalon_pwm_irq: STD_LOGIC;
    SIGNAL avalon_pwm_pwm_out: STD_LOGIC_VECTOR;

    NewComponent avalon_pwm
    (
        CLK_PRESCALER_WIDTH                                          =>
avalon_pwm_CLK_PRESCALER_WIDTH,
        PWM_COUNTER_WIDTH   => avalon_pwm_PWM_COUNTER_WIDTH,
        PWM_OUTPUTS_COUNT   => avalon_pwm_PWM_OUTPUTS_COUNT,
        PRELOAD_REGS        => avalon_pwm_PRELOAD_REGS,
        CONSTANT_MAX        => avalon_pwm_CONSTANT_MAX,
        PULSE_DITHER        => avalon_pwm_PULSE_DITHER,
        clk                 => avalon_pwm_clk,
        reset_n             => avalon_pwm_reset_n,
        chipselect          => avalon_pwm_chipselect,
        address             => avalon_pwm_address,
        write               => avalon_pwm_write,
        writedata           => avalon_pwm_writedata,
        read                => avalon_pwm_read,
        readdata            => avalon_pwm_readdata,
        irq                 => avalon_pwm_irq,
        pwm_out             => avalon_pwm_pwm_out,
    );

    NewComponent SPI_Master
    (
```

```
    CLK_Frequency => 12000000,
    SPI_CLK       => 1000000,
    Bits          => SPI_Master_Bits,
    CPol          => '1',
    CPha          => '1',

    Reset         => SPI_Master_Reset,
    MISO          => MISO,
    SCLK          => SCLK,
    SS            => SS,
    MOSI          => MOSI,
    TX_Data       => SPI_Master_TX_Data,
    Enable        => SPI_Master_Enable,
    Cont          => SPI_Master_Cont,
    Busy          => SPI_Master_Busy,
    RX_Data       => SPI_Master_RX_Data,
);
NewComponent NIOSDuino_Processor
(
    Reset         => Reset,
    sdram_addr    => sdram_addr,
    sdram_ba      => sdram_ba,
    sdram_cas_n   => sdram_cas_n,
    sdram_cke     => sdram_cke,
    sdram_cs_n    => sdram_cs_n,
    sdram_dq      => sdram_dq,
    sdram_dqm     => sdram_dqm,
    sdram_ras_n   => sdram_ras_n,
    sdram_we_n    => sdram_we_n,
    sdram_clk_clk => sdram_clk_clk,
```

```
        pio_export    => pio_export,
        i2c_sda       => i2c_sda,
        i2c_scl       => i2c_scl,
    );
}
```