ЗМІСТ

[ВСТУП 4](#_Toc9413075)

[РОЗДІЛ 1 ТЕОРІЯ ЗВУКУ ТА МУЗИКИ 5](#_Toc9413076)

[1.1. Звук як фізичне явище 5](#_Toc9413077)

[1.2. Спектральний склад звуку 6](#_Toc9413078)

[1.3. Музичні терміни та правила 7](#_Toc9413079)

[1.4. Методи та основи цифрового синтезу 8](#_Toc9413080)

[1.4.1. Форми хвиль, що використовуються у музиці 9](#_Toc9413081)

[1.4.2. Архітектура синтезатора 10](#_Toc9413082)

[ВИСНОВКИ ДО РОЗДІЛУ 1 13](#_Toc9413083)

[РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ДЛЯ ІНТЕГРУВАННЯ ТА МОЖЛИВИХ РЕАЛІЗАЦІЙ ПРОЕКТУ 14](#_Toc9413084)

[2.1 Перелік можливих систем для інтеграції, їх аналіз, недоліки та переваги 14](#_Toc9413085)

[2.1.1 Системи на мікроконтролері 14](#_Toc9413086)

[2.1.2 Програмний додаток на ПК(синтез на ЦП) 16](#_Toc9413087)

[2.1.3 Схема синтезатора складена з окремих мікро чипів, пам’яті, периферії 16](#_Toc9413088)

[2.1.4 ПЛІС(FPGA) 18](#_Toc9413089)

[2.2 Огляд архітектури FPGA на прикладі Intel Cyclone IV 25](#_Toc9413090)

[ВИСНОВКИ ДО РОЗДІЛУ 2 27](#_Toc9413091)

[РОЗДІЛ 3 СХЕМА СИНТЕЗУ ЗВУКУ. ОСНОВНІ ЧАСТИНИ СПРОЕКТОВАНОГО СИНТЕЗАТОРА 28](#_Toc9413092)

[3.1 GEN 28](#_Toc9413093)

[3.2 OSCILL 30](#_Toc9413094)

[3.3 CONV\_INT\_TO\_FLOAT 31](#_Toc9413095)

[3.4 CONV\_FLOAT\_TO\_INT 32](#_Toc9413096)

[3.5 TIMER 33](#_Toc9413097)

[3.6 MUL 34](#_Toc9413098)

[3.7 ADSR 35](#_Toc9413099)

[3.8 BLOCK 36](#_Toc9413100)

[3.9 ADD 37](#_Toc9413101)

[3.10 UART\_RX 38](#_Toc9413102)

[3.11 BUTTON\_PROC 38](#_Toc9413103)

[3.12 UART\_PROC 39](#_Toc9413104)

[3.13 ЦАП ЗА СХЕМОЮ R2R 40](#_Toc9413105)

[3.14 РЕЗУЛЬТАТИ СИМУЛЯЦІЙ 42](#_Toc9413106)

[ВИСНОВКИ ДО РОЗДІЛУ 3 47](#_Toc9413107)

[ВИСНОВКИ 48](#_Toc9413108)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 49](#_Toc9413109)

[Додаток А 50](#_Toc9413110)

[Додаток Б 51](#_Toc9413111)

# ВСТУП

# РОЗДІЛ 1 ТЕОРІЯ ЗВУКУ ТА МУЗИКИ

* 1. **Звук як фізичне явище**

Як фізичне явище, звук представляє собою механічне коливання у деякому середовищі. Як і хвиля, звук характеризується амплітудою, частотою, тиском та складом.

Частота або ж висота звуку – фізична величина, що характеризує кількість повторів певної події за 1 секунду часу. Але через те, що звук є складним процесом і найчастіше являє собою суму кількох гармонік, то висотою звуку називають частоту найбільш помітної гармоніки(основного тону).

Амплітуда – фізична величина, що характеризує найбільше зміщення сигналу, що періодично змінюється. Гучність залежить не лише від амплітуди, а й від частоти та спектрального складу.

Гучність звуку є відносною величиною і дорівнює рівню звукового тиску, створюваного синусоїдальним тоном з частотою 1кГц. Тобто частоти з однаковою амплітудою створюють різним за рівнем тиск в залежності від частоти. На рисунку 1.1 наведено графік рівня звукового тиску.

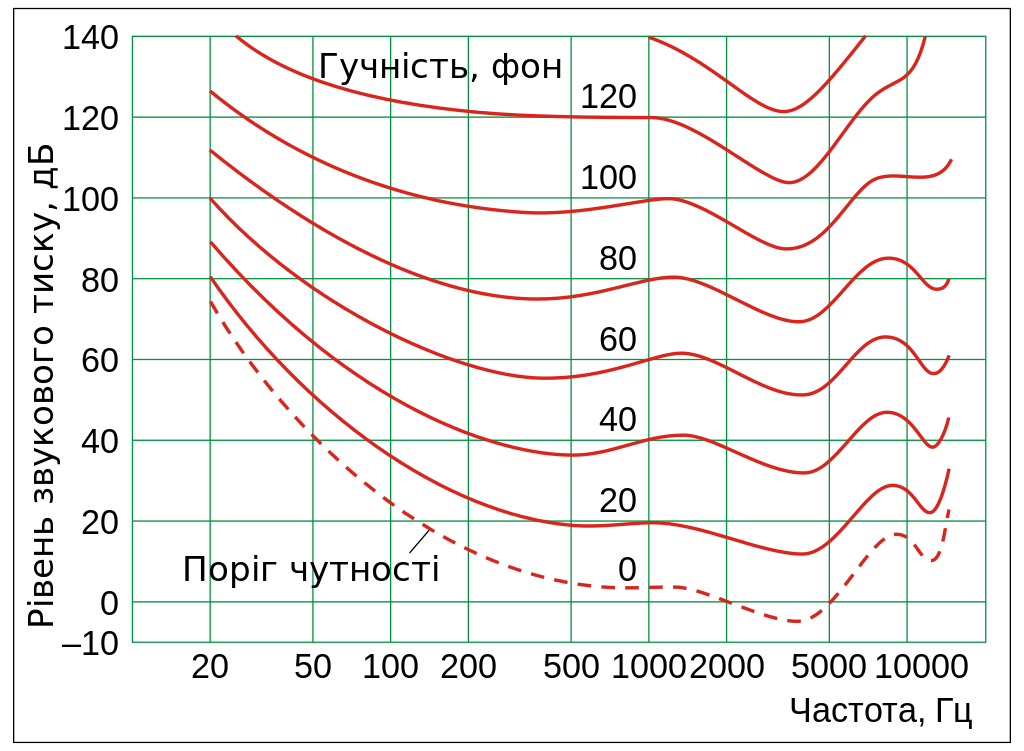


Рисунок 1.1 Криві рівнів гучності

* 1. **Спектральний склад звуку**

Отже будь-який звук, як в музиці так і в природі, є складним звуком, тобто складається з багатьох гармонік різної частоти та амплітуди.

Але при цьому можна виділити основний фон з найбільшою амплітудою. Інші з гармоніки називаються обертонами. Так для кожного інструменту є приклади його спектрального складу або ж як кажуть у музиці – тембру. На рисунку 1.2 наведено тембри декількох інструментів.

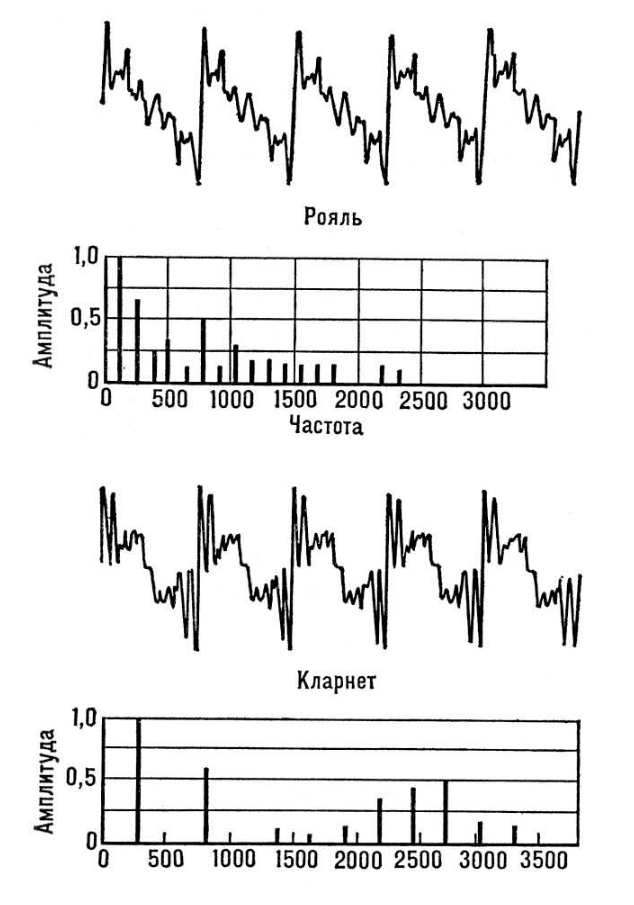


Рисунок 1.2 Тембр

* 1. **Музичні терміни та правила**

Нота – умовний символ, що характеризує висоту звуку.

Октава – відстань між двома тонами, частоти яких відрізняються у 2 рази, наприклад нота До 2-ї октави у 2 рази вище за ноту До 1-ї. Також октавою називають 7 основних тонів та 5 полу тонів. На рисунку 1.3 зображена одна октава, що містить 12 тонів. На рисунку 1.4 зображені всі існуючі октави.

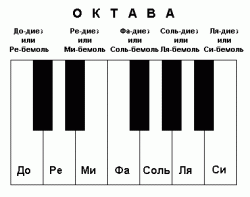


Рисунок 1.3 Ноти



Рисунок 1.4 Перелік октав, що містяться у фортепіано

Частота ж між двома сусідніми нотами в одній октаві відрізняється у 1,059463 рази (21/12). У таблиці 1.1 та 1.2 представлені частоти усіх нот.

Табл. 1.1 – Частоти у герцах усіх нот звукоряду

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ДО | РЕ | МИ | ФА | СОЛЬ | ЛЯ | СИ |
| Субконтр | 16,352 | 18,354 | 20,602 | 21,827 | 24,5 | 27,5 | 30,868 |
| Контр | 32,704 | 36,708 | 41,204 | 43,654 | 49 | 55 | 61,736 |
| Велика | 65,408 | 73,416 | 82,408 | 87,308 | 98 | 110 | 123,472 |
| Мала | 130,816 | 146,832 | 164,816 | 174,616 | 196 | 220 | 246,944 |
| 1 | 261,632 | 293,664 | 329,632 | 349,232 | 392 | 440 | 493,888 |
| 2 | 523,264 | 587,328 | 659,264 | 698,464 | 784 | 880 | 987,776 |
| 3 | 1046,528 | 1174,656 | 1318,528 | 1396,928 | 1568 | 1760 | 1975,552 |
| 4 | 2093,056 | 2349,312 | 2637,056 | 2793,856 | 3136 | 3520 | 3951,104 |
| 5 | 4186,112 | 4698,624 | 5274,112 | 5587,712 | 6272 | 7040 | 7902,208 |
| 6 | 8372,224 | 9397,248 | 10548,22 | 11175,42 | 12544 | 14080 | 15804,42 |

Табл. 1.2 – Частоти у герцах усіх нот диез звукоряду

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ДО ДИЕЗ | РЕ ДИЕЗ | ФА ДИЕЗ | СОЛЬ ДИЕЗ | ЛЯ ДИЕЗ |
| Субконтр | 17,324 | 19,445 | 23,125 | 25,957 | 29,135 |
| Контр | 34,648 | 38,89 | 46,25 | 51,914 | 58,27 |
| Велика | 69,296 | 77,78 | 92,5 | 103,828 | 116,54 |
| Мала | 138,592 | 155,56 | 185 | 207,656 | 233,08 |
| 1 | 277,184 | 311,12 | 370 | 415,312 | 466,16 |
| 2 | 554,368 | 622,24 | 740 | 830,624 | 932,32 |
| 3 | 1108,736 | 1244,48 | 1480 | 1661,248 | 1864,64 |
| 4 | 2217,472 | 2488,96 | 2960 | 3322,496 | 3729,28 |
| 5 | 4434,944 | 4977,92 | 5920 | 6644,992 | 7458,56 |
| 6 | 8869,888 | 9955,84 | 11840 | 13289,98 | 14917,12 |

* 1. **Методи та основи цифрового синтезу**

Існує багато способів синтезувати звук :

* Адитивний. Даний метод синтезу полягає у додаванні до основного тону ще декількох гармонік кратної частоти. Прикладом адитивного синтезу є орган.
* Субтрактивний. Цей метод є зворотнім до адитивного. Заснований на «відніманні» від сигналу його складових, тобто заснований на фільтрах.
* Семплювання. Цей метод заснований на отримані звуку з записаного файлу та його подальшій обробці. Схожий на wavetable.
* FM/AM синтез. Вхідний тембр звуку спотворюється завдяки іншому(моделюючому) сигналу, що впливає на його форму.
* Фізичне моделювання. Використовує математичні моделі для отримання звуку.
* Wavetable синтез
* Інші
  + 1. **Форми хвиль, що використовуються у музиці**

Раніше синтезатори могли синтезувати тільки синусоїдальні сигнали(аналогові синтезатори). З приходом транзисторів з’явилася можливість синтезувати трикутні, прямокутні, пилу та імпульсні сигнали. На рисунку 1.5 наведені ці форми сигналів.

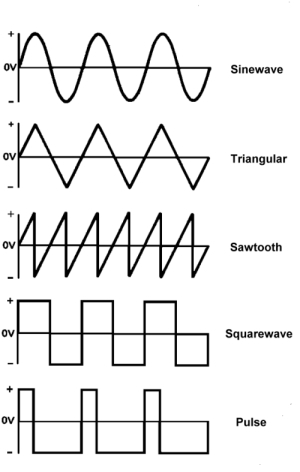


Рисунок 1.5 Форми сигналів

Синусоїдальні або як їх ще називають прості, адже з ним отримується складний звук.

Прямокутні(меандр) – така форма сигналу використовується для імітації басових звуків і духовних інструментів.

Пиловидні – використовуються для смичкових інструментів.

Трикутні сигнали – їх звучання найбільш схоже на звук труби органу.

Імпульсні сигналі є прямокутними з певною скважністю. Скважність К означає, що сигнал включає всі гармоніки, крім гармонік кратних К.

* + 1. **Архітектура синтезатора**

На рисунку 1.6 зображена архітектура синтезатора. Далі описані функції усіх цих модулів.

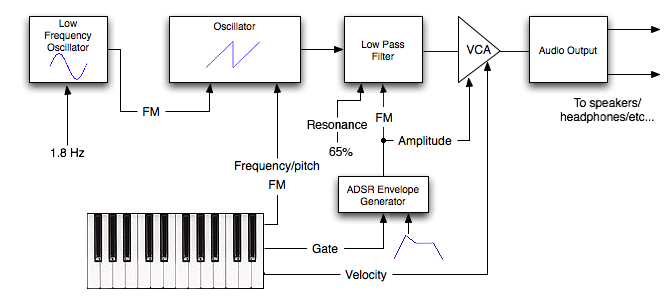


Рисунок 1.6 Архітектура модульного синтезатора

* + - 1. **Осцилятор**

В класичних синтезаторах усі форми звуку генеруються осцилятором(VCO) і далі вже йдуть на обробку. Генерація відбувається безперервно.

* + - 1. **Зміна гучності**

В аналогових синтезаторах обробка гучності відбувається у VCA – підсилювач контрольований напругою. В цифровому синтезі це можна реалізувати за допомогою помножувача на певний коефіцієнт.

* + - 1. **Фільтр**

Використовується для обмеження чи контрастності певного діапазону частот. Як правило це фільтр низьких частот, фільтр високих частот та смуговий фільтр.

* + - 1. **Генератор обвідної хвилі**

Генератором обвідної хвилі може бути низькочастотний фільтр з можливістю вибору форми або генератор ADSR, що є більш складним. За допомогою цього генератора вихідна гучність змінюється за кривою, що задана у генераторі. Таким чином можна досягти звучання схожого на певний інструмент. На рисунку 1.7 можна побачити обвідні, що характерні деяким інструментам.

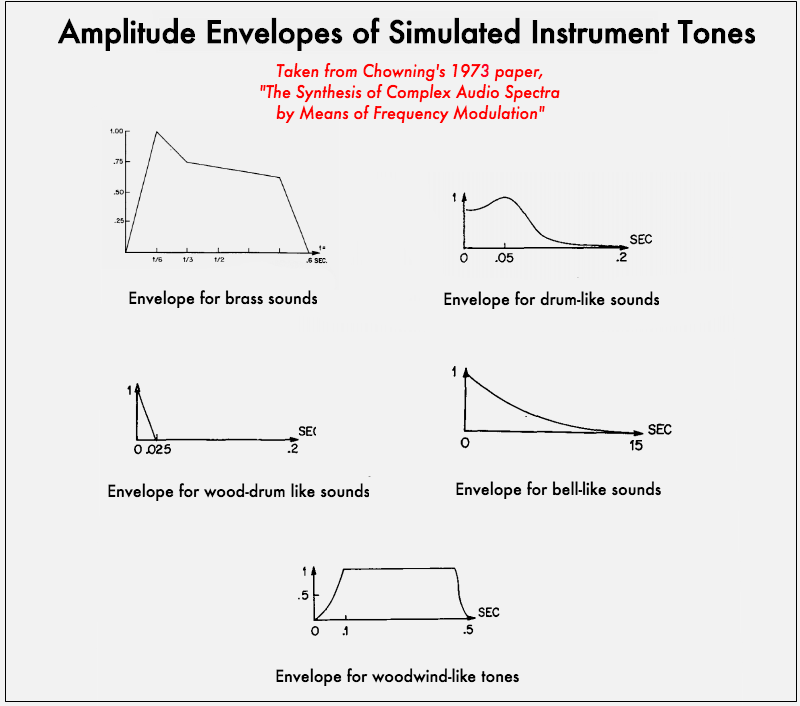


Рисунок 1.7 Графіки обвідних деяких інструментів

Як видно з рисунку ADSR складається з 4 частин:

* Attack(атака) – гучність з 0 збільшується до максимуму.
* Decay(спад) – гучність зменшується до рівня утримування
* Sustain(утримування або підтримка) – тривалість цієї ділянки не залежить від параметрів, а триває доки натиснута клавіша. Рівень на якому утримується сигнал є 80-90 %.
* Release(відпускання) – ця ділянка починається після відпускання клавіші. Гучність падає з рівня підтримування до нуля.
  + - 1. **LFO. Низькочастотний осцилятор**

Блок генерує сигнал малої частоти від 1 до 2 Гц певної форми. Цей сигнал можна використовувати як обвідну амплітуди або для осцилятора для корегування частоти сигналу.

## ВИСНОВКИ ДО РОЗДІЛУ 1

В цьому розділі описана усі основні поняття музики та стадії синтезу. Отриманих знань достатньо для подальшого проектування обраного продукту. Виконання усіх вищенаведених модулів забезпечить повноцінну роботу продукту на рівні існуючих конкурентних рішень. У наступному розділі буде проведено аналіз можливих реалізацій та обґрунтування обраного мною способу.

# РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ДЛЯ ІНТЕГРУВАННЯ ТА МОЖЛИВИХ РЕАЛІЗАЦІЙ ПРОЕКТУ

* 1. **Перелік можливих систем для інтеграції, їх аналіз, недоліки та переваги**
     1. **Системи на мікроконтролері**

Почнемо з огляду та основних характеристик сучасних систем.

Архітектура ARM представлена мікроконтролерами Cortex STM32. Дані мікроконтролери досить розвинені, найбільш продуктивна версія цього мікроконтролеру Cortex-M4. Наразі найпопулярніші рішення мають 32-бітну архітектуру, постійну пам’ять, оперативну пам’ять, таймери, інтерфейси обміну даними, швидку обробку переривань та працюють на частотах до 168 МГц.

На рисунку 2.1 зображена архітектура Cortex-M4.

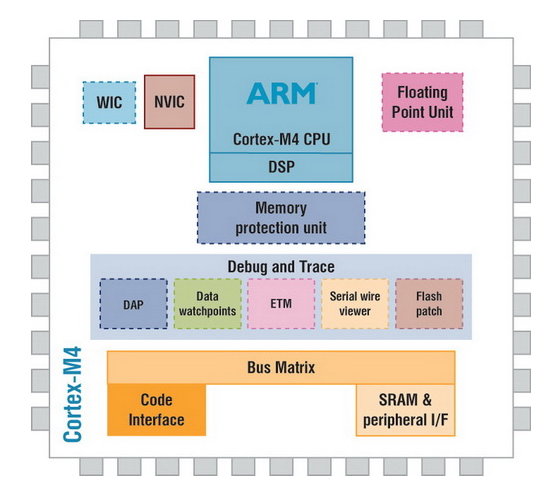


Рисунок 2.1

Компанія розміщує на кристалі мікроконтролер та DSP. Завдяки окремому блоку DSP, що виконує операції з плаваючою комою, цей мікроконтролер можна використовувати в нестандартних для нього проектах. Ядро підтримує множення з накопленням (МАС) за 1 такт, обчислювальний блок FPU, що виконує операції за 1 такт та SIMD.

Структура також вражає, мікроконтролер має безліч інтерфейсів( I2C, SPI, USART, USB, camera interface, Ethernet) та аналогових засобів (АЦП, ЦАП). На рисунку 2.2 зображена структурна схема ядра.

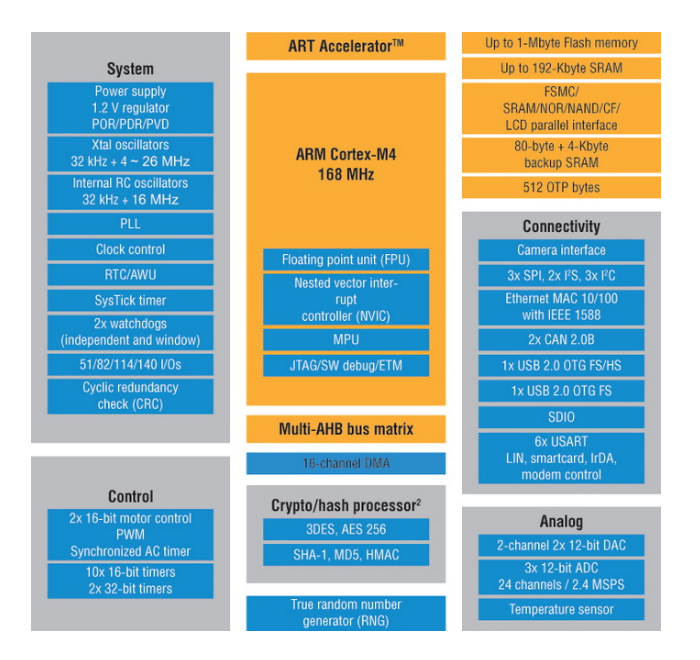


Рисунок 2.2

Дійсно це сімейство мікроконтролерів, вражає та розширює ринок їх використання. Необхідність у використанні окремих чипів DSP зникла, а багата периферія дозволяє реалізувати проекти значної складності.

Але мікроконтролер виконує послідовність команд, тому його можливості не безграничні. У разі досить довгого ланцюга фільтрів та обробок, час реакції на зовнішні події збільшиться, і звісно досягти паралельного генерування не вдасться, що призведе знову ж таки до неприпустимих затримок.

* + 1. **Програмний додаток на ПК(синтез на ЦП)**

Звичайно ЦП комп’ютера як і мікроконтролер виконує послідовність команд, але його архітектура набагато розвиненіша за мікроконтролер.

Перевагою процесора є наявність спеціалізованих блоків та інструкцій що виконують швидкі перетворення над числами великої розрядності. Також неможна забувати про наявність, як мінімум 2-х ядр, що працюють паралельно. Тому ця система підходить для реалізації даного проекту. Це підтверджено багатьма програмними доданками, що існують у даний час. Але існують і недоліки:

* Вартість такої системи перевищує вартість окремих пристроїв, адже необхідний досить потужний, а отже і коштовний процесор та коштовне програмне забезпечення
* Вимагає окремого комп’ютера для роботи, наприклад у студії або на концерті
* Вимагає окремої клавіатури, що приєднується до ПК
* Навіть на швидкісних процесорах виникають деякі затримки
  + 1. **Схема синтезатора складена з окремих мікро чипів, пам’яті, периферії**

На даний момент, це найпопулярніший варіант синтезатора. Його переваги полягають у спеціалізований чипах, що реалізують цифрову обробку сигналів (DSP), де виконується усе генерування та обробка ефектами. Наявний процесор виконує функції керування екраном та усіма клавішами налаштувань, що наявні на корпусі, також реалізує зв’язок з оперативною пам’яттю. Хоча і сам чип DSP має підключення до постійної пам’яті. На рисунку 2.3 наведена схема такого синтезатора.

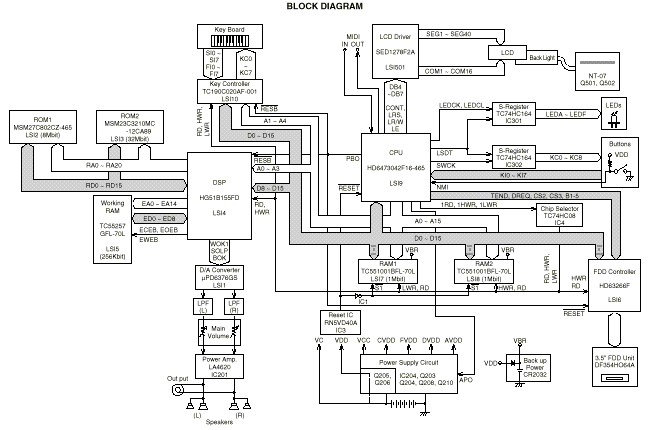


Рисунок 2.3

Найбільшу увагу треба звернути саме на мікросхему DSP. Ці процесори виконують операції над числами з фіксованою, або плаваючою комою, найчастіше мають архітектуру VLIW або SIMD і працюють на частотах до 1 ГГц. Нижче представлений опис найбільш популярних процесорів з фіксованою комою: сімейство Blackfin від компанії ADI, MSC81XX від Freescale, TMS320C64X від Texas Instruments. Усі ці процесори являються 16-бітними, використовують архітектуру VLIW та мають декілька блоків MAC(multiply-accumulate), що виконують функцію множення і додавання. Середня ціна таких процесорів 30 доларів. Але існують і багатоядерні рішення з масовим паралелізмом, що мають нижчу продуктивність і меншу ціну. Найкращим же рішенням є процесори з плаваючою комою від виробників Texas Instruments та Analog Devices. Побудовані на базі 8-тактної архітектури VLIW, мають набір інструкцій з фіксованою комою, але й спеціалізовані окремі блоки для роботи з плаваючою комою, можуть виконувати дві 32-бітні МАС-операції на частоті 350МГц. Ціна такого процесора близька до 200 доларів.

* + 1. **ПЛІС(FPGA)**

До цього усі можливі рішення були засновані на процесорах, спеціалізованих процесорах, швидкому виконанні послідовності команд.Але тепер розглянемо **ПЛІС (**Програмовану логічну інтегральну схему).



Рисунок 2.4

ПЛІС не виконує команди, а за допомогою мови описуапаратури, наприклад VHDL або Verilog, дозволяє написати такі блоки, що виконують певну дію безперервно в режимі конвеєру. Використовуючи ПЛІС ми не прив’язані до кількості ядр, що можуть одночасно виконати обмежену кількість операцій, ми можемо встановити певну кількість блоків, що будуть працювати одночасно і незалежно. Сучасні архітектури ПЛІС навіть не вимагають додаткової периферії такої як інтерфейс обміну даними з пам’яттю або комп’ютером, адже будь-яку схему можна реалізувати всередині кристалу. Сучасні ПЛІС вже мають вбудовану пам'ять, швидкісні інтерфейси вводу\виводу(PCI Express), спеціалізовані блоки множення, блоки PLL, що дозволяють отримати надвисокі частоти роботи. І все це починаючи від 10 доларів за кристал з 6 тисячами логічних комірок. В цьому випадку найбільшою проблемою є саме проектування схеми. І так як уся схема знаходиться на одному кристалі і звільняє нас від з’єднань між різними мікросхемами, то надійність такою системи вища. Як результат, FPGA може бути імплемінтована в будь-який проект:

* пристрої комунікації(маршрутизатори)
* обробка картинок та відео
* прискорення математичних розрахунків
* інші швидкісні схеми

Інший варіант SoC, що поєднує мікропроцесор та ділянку FPGA, дозволяє поєднувати 2 способи роботи, мікрокоманди процесора та паралельні блоки на FPGA.

Існують багато технологій за якими виробляють програмовані логічні схеми схеми. На рисунку 2.5 наведені основні типи ПЛІС.

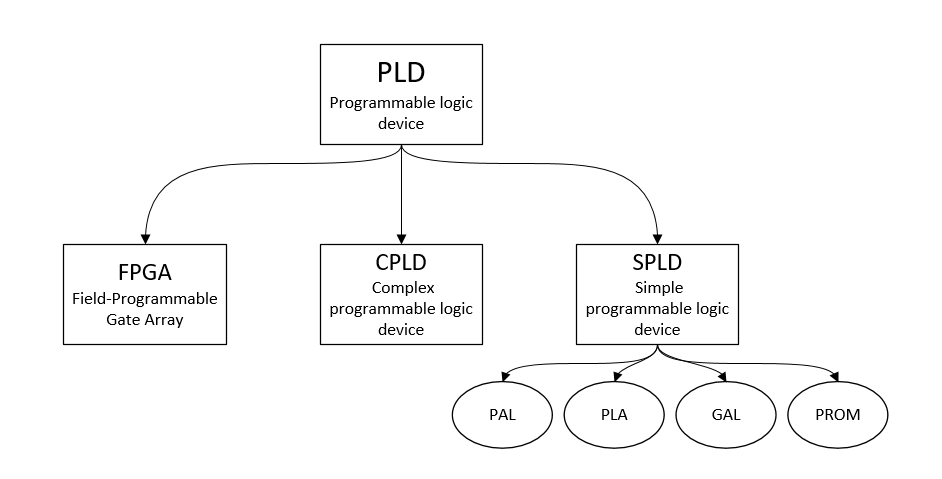


Рисунок 2.5

Кожне з цих сімейств відрізняється шляхом програмування, способом з’єднання ліній та багато чим іншим. Старші сімейства починалися з плавких контактів, тобто схему можна було прошити лише раз, інші мали енергонезалежну пам’ять, яку потрібно було прошивати. І лише з приходом CPLD та FPGA вдалося отримати енергонезалежну схему з безмежною кількістю перепрограмувань, що відбувалися дуже легко. На рисунку 2.6 показані види контактів, що «програмуються».

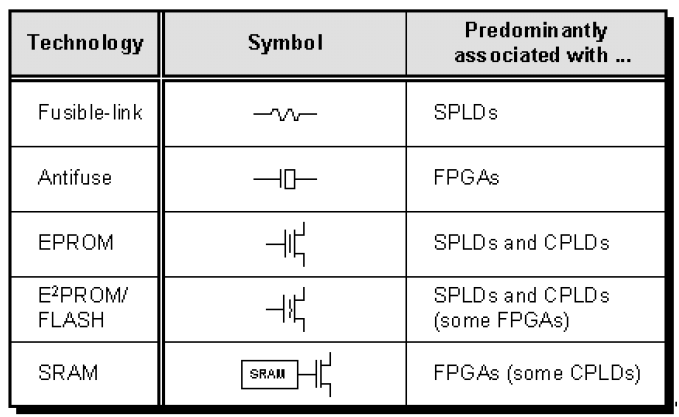


Рисунок 2.6

Fusible-link – легкоплавке з’єднання, за допомогою спеціального програматора, непотрібні зв’язки руйнувалися, перепрограмування неможливе.

Antifuse – як і легкоплавке з’єднання antifuse є одноразовим, в початковому стані antifuse є розімкненим, і лише при протіканні току – з’єднується. Фізика процесу зворотня до запобіжника.

EPROM транзистор – до його виникнення у 1971 році усі зв1язки програмувалися лише раз. Тому створення транзистора було обов’язковим. Плаваючий затвор повністю ізольований від усіх інших частин і тому може зберігати свій стан та придатний до стирання за допомогою ультрафіолету. За допомогою більш високої напруги проходив запис у транзистор, що відповідав одному біти. Так програмування інтегральної схеми звелося до очистки пам’яті ультрафіолетом та прошиванню чипа на програматорі.

EEPROM – ця пам’ять стала наступним кроком. Комірка пам’яті тепер могла бути очищена напругою.

SRAM – постійна оперативна пам’ять. Відрізнялась зворотнім зв’язком, що звільнило від періодичних перезаписів динамічної пам’яті. Енергозалежна пам’ять, але найпростіший спосіб читання\запису. Стирання пам’яті при вимкнені живлення призводить до встановлення Flash-пам’яті. При кожному вмиканні FPGA прошиває сама себе.

FPGA від різних виробників можуть відрізнятися архітектурою і структурними частинами, але все ж мають однакову базу. Концепція полягає у програмованих блоках, що складаються з LUT( look-up table)(таблиця істинності), тригера та мультиплексора. FPGA складається з багатьох блоків, починаючи від декількох тисяч, закінчуючи мільйонами.

Та логічні блоки також бувають різних типів. На рисунку 2.7 зображений MUX-блок, що реалізує необхідну функцію за допомогою багатьох мультиплексорів.

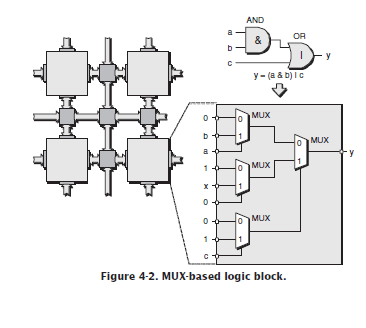


Рисунок 2.7

Альтернативою і основним типом логічних блоків є LUT. На рисунку 2.8 зображена реалізація тої самої функції. Ця таблиця істинності програмується у елемент. Для забезпечення гнучкості FPGA має логічні елементи з різної кількістю входів, тобто LUT, що реалізує функцію від різної кількості змінних.

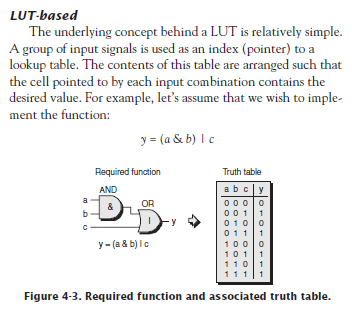


Рисунок 2.8

Деякі функції, наприклад множення, ділення, підведення до степені, займають багато місця і працюють повільно якщо вони реалізовані на логічних елементах. Саме через це створили жорсткі блоки всередині ПЛІС – вбудовані помножувачі. Вони розподілені по всьому кристалу, розміщуються поряд с блоками RAM. Деякі ПЛІС мають виділені блоки суматорів, бо однією з популярних операцій DSP є MAC(multiply-and-accumulate). На рисунку 2.9 зображена ця операція.

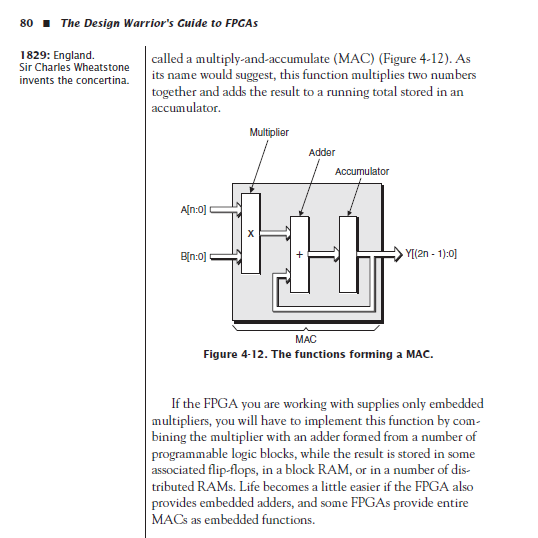


Рисунок 2.9 МАС

Коли написаний код перетворюється у схему, важливо розуміти як саме компілятор це зробить. Адже однакові на перший погляд ділянки коду можуть мати різну швидкість та займати більше ресурсів. Далі наведено приклади різниці між послідовним та паралельним способами опису схеми.

На рисунку 2.10 Зображена згенерована схема, як видно з коду схема реалізує мультиплексор, але конструкція if-then-else призвела до послідовного поєднання багатьох мульплексорів, саме така поведінка призведе до затримки та займе більше логічних елементів.

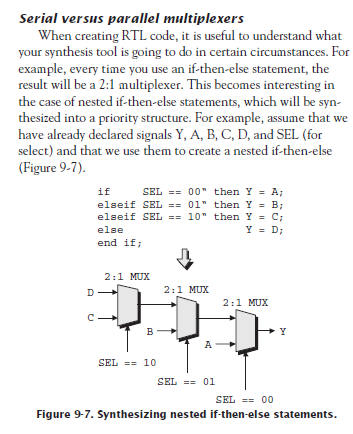


Рисунок 2.10

Правильною поведінкою буде схема з одного мультиплексора з 4 входами. На рисунку 2.11 зображена ця схема, що реалізована за конструкцією case.

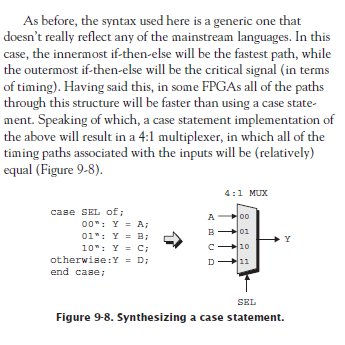


Рисунок 2.11

Можна зробити висновок, що будь-який проект можна реалізувати сотнями способів, що будуть працювати абсолютно по-різному, мати різну конфігурацію, будуть використовувати різну кількість логічних блоків та спеціалізованих. Часто при використанні IP cores, користувачу дається вибір чи використовувати спеціалізовані блоки чи будувати схему на логічних елементах, чи акцентувати увагу на швидкість чи на економію місця. На рисунку 2.12 наведений приклад, що показує 2 різні реалізації однієї функції.

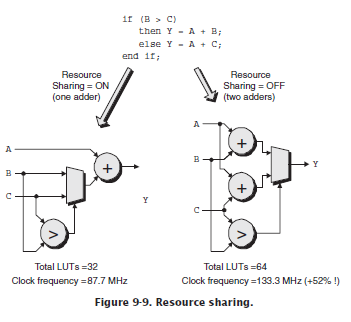


Рисунок 2.12

Як видно з рисунку, перший варіант реалізовано послідовно. Спочатку відбудеться порівняння, результат його потрапить на мультиплексор і вже його результат потрапить на суматор. Цей варіант займає 32 логічні елементи, але через те що в ланцюгу стоїть три блоки, максимальна частота такої схеми 87 МГц. Інший варіант спрямований на швидкість. Додавання в цьому варіанті відбувається в обох випадках і паралельно з цим відбувається порівняння B і C. В цьому ланцюгу послідовність з 2 блоків, тому частота цієї схеми може бути 133 МГц, що в півтора рази більше.

* 1. **Огляд архітектури FPGA на прикладі Intel Cyclone IV**

Логічні елементи(LЕs) є найменшими блоками в архітектурі ПЛІС. Кожен ЛЕ має :

* LUT. Чотирьох входовий блок, що реалізує функцію за таблицею істинності.
* Програмований регістр
* Вихід регістру
* Вихід функції LUT

Регістр можна запрограмувати на D,RS,T,JK тригер. Кожен регістр має порти вхідних даних, синхронізації, дозволу синхронізації, очистки.

Кожен LE має три виходи, які керують локальними, рядковими та колонковими ресурсами маршрутизації. На рисунку 2.13 зображений один логічний елемент.

Кожна LAB(logic array block) складається з таких функцій:

* 16 ЛЕ
* Сигнали керування LAB
* ЛЕ-сполучення
* Регістрові сполучення
* Місцеві з’єднання між ЛЕ

Локальні сполучення передають сигнали між ЛЕ в одній LAB. Регістрові сполучення передають сигнали між регістрами ЛЕ в одній LAB. Cyclone IV включає в собі вбудовані модулі пам’яті M9K, що налаштовуються на різні функції, такі як: оперативна пам’ять, регістри зсуву, ROM, FIFO. На рисунку 2.14 зображений кристал Cyclone IV, розміщення основних частин.

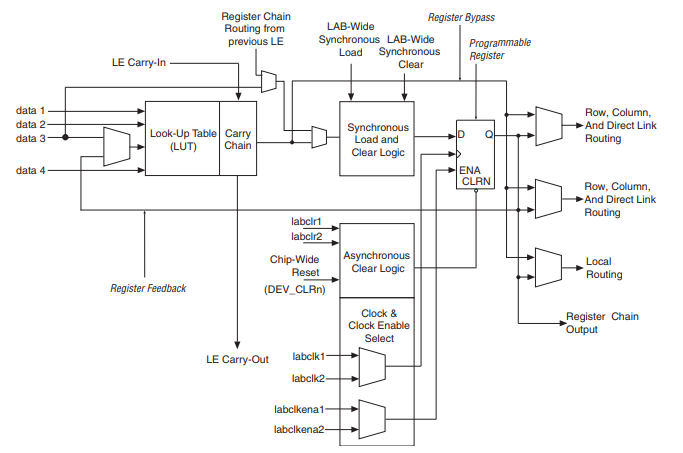


Рисунок 2.13 Логічний елемент

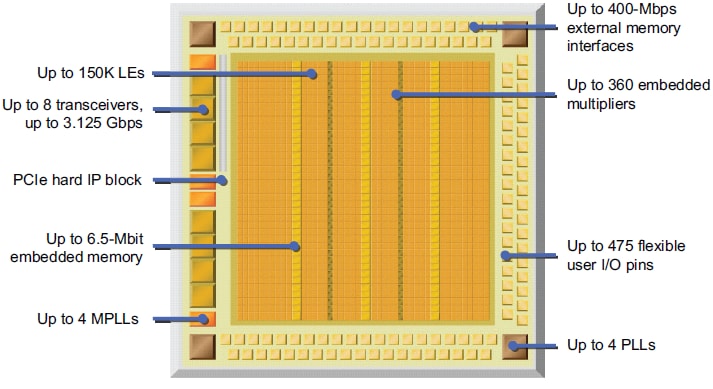


Рисунок 2.14 Топологія кристалу Cyclone IV

## ВИСНОВКИ ДО РОЗДІЛУ 2

Переглянувши усі доступні системи реалізації цифрового синтезатора, оптимальний вибір падає на ПЛІС. Вартість такого рішення набагато менша за вартість спеціалізованих процесорів, а швидкодія і часові затримки набагато менші ніж у мікроконтролерах. Гнучкість ПЛІС дозволяє проектувати різні фільтри, обробки сигналу, інтерфейси зв’язку з ПК та багато іншого. При тому маючи продукт його покращення не потребує замін деталей, корпусу та всього іншого, а полягає лише у зміні налаштувань ПЛІС. Це дозволяє робити унікальні продукти для кожного користувача в залежності від його потреб. В наступному розділі будуть описані модулі, що були розроблені для синтезатора на мові опису апаратури VHDL.

**РОЗДІЛ 3 СХЕМА СИНТЕЗУ ЗВУКУ. ОСНОВНІ ЧАСТИНИ СПРОЕКТОВАНОГО СИНТЕЗАТОРА**

* 1. **GEN**

Модуль складається з накопичувального суматора. Доданком є число, що вираховується в залежності від тактової частоти схеми, необхідної частоти на виході та розміру акумулятору.

*(1)*

*(2)*

Старші біти акумулятора є адресом комірки пам’яті або джерелом сигналу типу «Пила», старший біт є джерелом сигналу «Меандр». На рисунку 3.2 зображена симуляція цього модуля. Для забезпечення якості сигналу було вибрано 32-бітний акумулятор, що дозволяє редагувати вихідну частоту з точністю *FREQCLOCK*/232 = 0,0116 Гц. Для забезпечення плавності вихідного сигналу було обрано довжину пам’яті 4096 комірок, що дорівнює 12 бітам адреси. Ширина пам’яті є другим параметром плавності сигналу, але використання ЦАП великої розрядності призведе до великих похибок. Для досягнення відмінного сигналу бажано використовувати спеціалізовані мікросхеми ЦАП. Обрана ширина пам’яті дорівнює 12 бітам, тому ширина шини як адреси, так і даних дорівнює 12 бітів. На рисунку 3.1 наведене графічне представлення даного модуля.

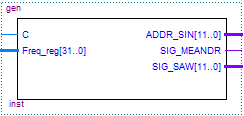


Рисунок 3.1 Модуль GEN

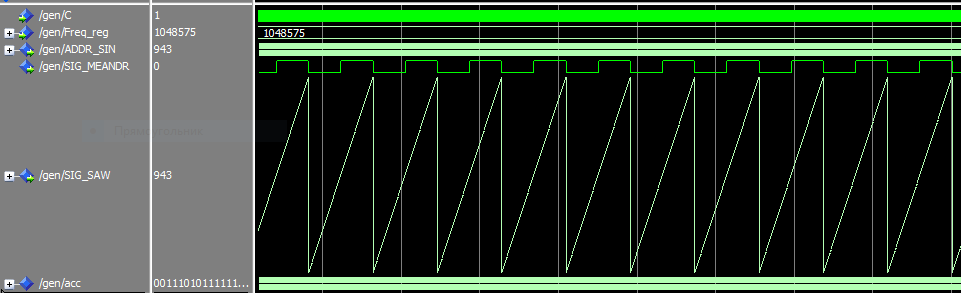


Рисунок 3.2 Симуляція GEN

У таблиці 3.1 наведено значення доданку для отримання певної частоти звуку. У пам’яті ПЛІС зберігаються лише значення частот для субконтроктави, що є найпершою октавою. Інші значення отримуються зсувом вліво.

Таблиця 3.1 – Значення доданків акумулятора для кожного значення частоти

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Октава | Клавіші | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 1405 | 1488 | 1577 | 1670 | 1770 | 1875 | 1986 | 2105 | 2230 | 2362 | 2503 | 2652 |
| 2 | 2809 | 2976 | 3153 | 3341 | 3539 | 3750 | 3973 | 4209 | 4459 | 4724 | 5005 | 5303 |
| 3 | 5619 | 5952 | 6306 | 6681 | 7079 | 7500 | 7946 | 8418 | 8919 | 9449 | 10011 | 10606 |
| 4 | 11237 | 11905 | 12613 | 13363 | 14158 | 14999 | 15891 | 16836 | 17838 | 18898 | 20021 | 21212 |
| 5 | 22474 | 23810 | 25226 | 26725 | 28315 | 29999 | 31783 | 33673 | 35675 | 37796 | 40043 | 42425 |
| 6 | 44948 | 47620 | 50451 | 53450 | 56630 | 59998 | 63566 | 67345 | 71350 | 75591 | 80086 | 84849 |
| 7 | 89896 | 95240 | 100902 | 106900 | 113261 | 119995 | 127131 | 134690 | 142700 | 151183 | 160171 | 169699 |
| 8 | 179792 | 190479 | 201804 | 213800 | 226521 | 239990 | 254262 | 269380 | 285400 | 302366 | 320343 | 339397 |
| 9 | 359584 | 380959 | 403609 | 427600 | 453043 | 479981 | 508524 | 538761 | 570800 | 604731 | 640685 | 678794 |

* 1. **OSCILL**

Модуль є оболонкою для модуля GEN. Адреса отримана з GEN потрапляє до пам’яті, дані з пам’яті виводяться через порт. Наявність мультиплексора забезпечує вибір вихідного сигналу(синусоїда з пам’яті, меандр і пила з GEN). На рисунку 3.4 наведена симуляція з усіма вихідними сигналами. На рисунку 3.3 зображено графічне представлення модуля. Вхід MULT\_OUT служить для вибору вихідного сигналу :

* «00» - X «000»;
* «01» - меандр;
* «10» - пила;
* «11» - дані з пам’яті (синусоїда).

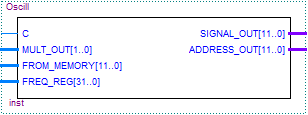


Рисунок 3.3 Модуль OSCILL

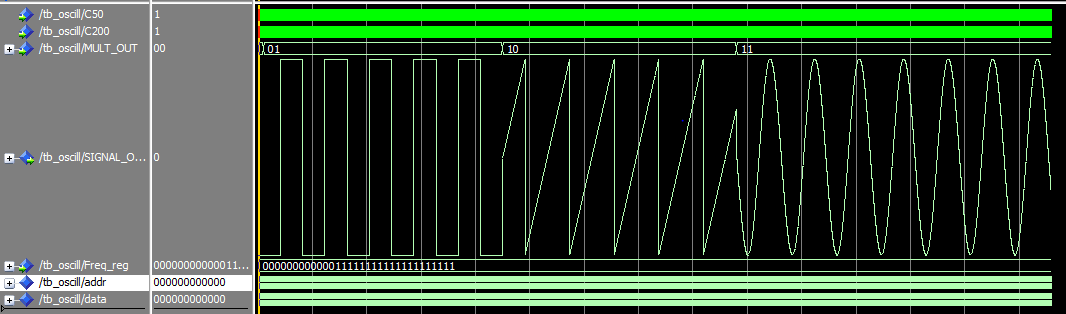


Рисунок 3.4 Симуляція OSCILL

* 1. **CONV\_INT\_TO\_FLOAT**

Модуль виконує перетворення цілого числа від 0 до 4095 у число з плаваючою комою для подальших перетворень амплітуди сигналу.

Перетворення виконується у форматі IEEE754. На рисунку 3. Зображене графічне представлення модуля. Даний модуль працює синхронно, про що свідчить вхід CLOCK, модуль має часову затримку між зміною даних на вході та видачою результату на виході. Ця затримка дорівнює 6 тактам, для зменшення часової затримки частоту роботи задано максимально можливою – 200 МГц. На рисунку 3.3 наведена симуляція цього блоку.

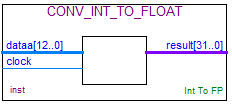


Рисунок 3.5 Модуль CONV\_INT\_TO\_FLOAT



Рисунок 3.6 Симуляція CONV\_INT\_TO\_FLOAT

* 1. **CONV\_FLOAT\_TO\_INT**

Модуль виконує перетворення числа з плаваючою комою у ціле число для передачі числа на суматор і ЦАП.

На рисунку 3. Зображене графічне представлення модуля. Даний модуль працює синхронно, про що свідчить вхід CLOCK, модуль має часову затримку між зміною даних на вході та видачою результату на виході. Ця затримка дорівнює 6 тактам, для зменшення часової затримки частоту роботи задано максимально можливою – 200 МГц.

Перетворення виконується у форматі IEEE754. На рисунку 3.4 наведена симуляція цього блоку.

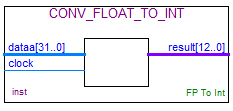


Рисунок 3.7 Модуль CONV\_FLOAT\_TO\_INT



Рисунок 3.8 Симуляція CONV\_FLOAT\_TO\_INT

* 1. **TIMER**

Модуль чекає на сигнал GATE, що свідчить про натискання клавіші синтезатора. Лічильник рахує кількість відпрацьованих тактів та переводить автомат у певний стан. Стан автомату у реальному часі передається у модуль ADSR. Користувач задає довжину кожного стану, кількість тактів кожного режиму задаються на відповідних входах модуля. На рисунку 3.5 наведено модуляцію цього блоку. Звісно від заданої частоти цього модуля залежить плавність зміни рівня гучності. Чим більше частота модуля, тим більше треба робити значення ATTACK\_TIME, DECAY\_TIME, RELEASE\_TIME, та робити менше значення відповідної дельти. Максимальне значення часу кожного стану дорівнє 231, що є границею типу natural. Тобто при частоті 200МГц довжина стану може бути 10 секунд, при частоті 10МГц – 214 секунд і так далі. Мінімальний числом дельти у форматі IEEE754 може бути 1.4Е-45(1.4\*10-45).

Перед тим як змінити стан автомату модуль видає імпульс сигналу CLEARADD, що приходить на вхід суматора, що очищує усі дані всередині нього, так як суматор має затримку результату, і навіть після зміни стану продовжить видавати дані минулого стану.

Так як по сигналу GATE не можна перевірити зайнятий Block чи ні, було введено сигнал BUSY, що дорівнює 1 поки TIMER знаходиться у робочих станах.

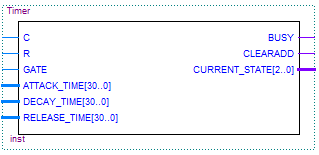


Рисунок 3.9 Модуль TIMER

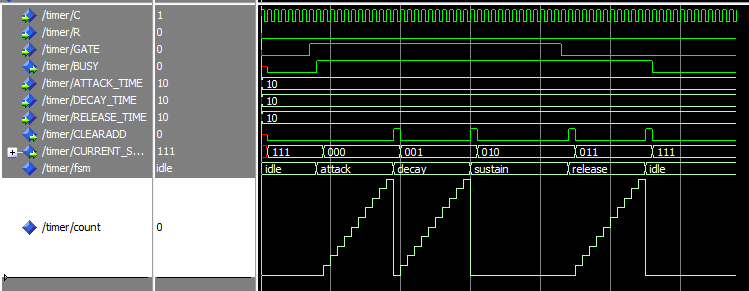


Рисунок 3.10 Симуляція TIMER

* 1. **MUL**

Модуль виконує множення чисел з плаваючою комою. Реалізує стадію обробку сигналу VCA, де вхідний сигнал з максимальною амплітудою обмежується коефіцієнтом, що отриманий від модуля ADSR. На рисунку 3.6 наведена симуляція даного блоку, де видно затримку у 6 тактів. Через це рекомендовано обирати максимально допустиму частоту роботи модуля – 200 МГц.

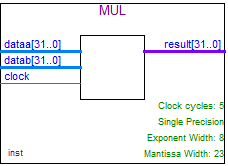


Рисунок 3.10 Модуль MUL

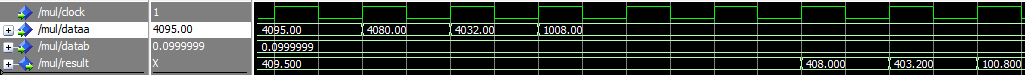


Рисунок 3.11 Симуляція MUL

* 1. **ADSR**

Модуль складається з TIMER та ADDSUB. Суматор працює в режимі накоплення, тобто одним з входів суматора є його вихід. Значення числа на іншому вході залежить від поточного стану, що приходить з TIMER. ADSR приймає на вхід три числа з плаваючою комою, ці числа є доданками, що подаються на вхід суматора. Так на виході ADSR виникає число з плаваючою комою, що росте з 0 до 1 в режимі ATTACK, спадає у режимі DECAY та RELEASE знову до 0. Це значення потрапляє на MUL, цим саме регулюється амплітуда вихідного сигналу. Нижче на рисунку 3.7 наведено симуляцію роботи. Завдяки зміні параметрів можна досягти різних звучань однієї і тої самої ноти.

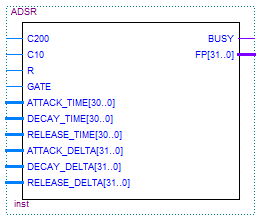


Рисунок 3.12 Модуль ADSR

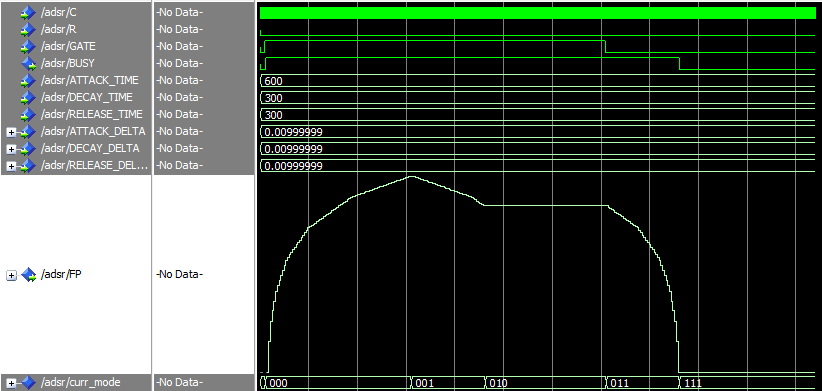


Рисунок 3.13 Симуляція ADSR

* 1. **BLOCK**

Цей модуль поєднує усі блоки, що були описані вище. Один блок може оброблювати одну натиснуту клавішу. Для повноцінної роботи синтезатор має мати як мінімум 5 таких блоків. Модуль приймає на свої входи такі параметри: натиск клавіші, що запускає таймер, число, необхідне GEN для генерації сигналу та параметри для блоку ADSR.

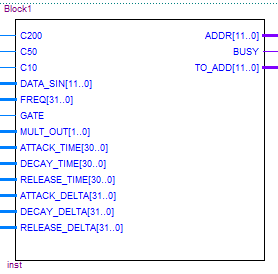


Рисунок 3.14 Модуль Block

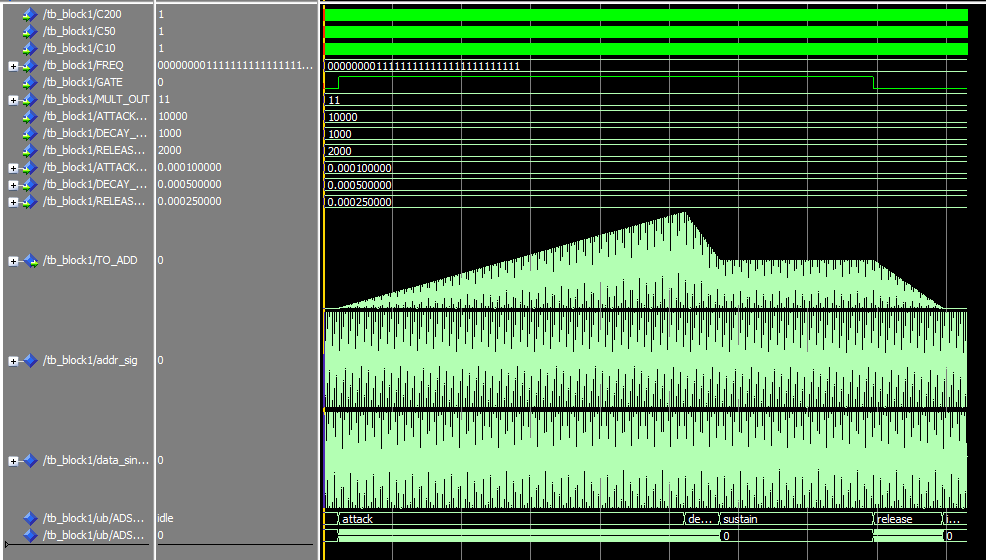


Рисунок 3.15 Симуляція BLOCK

* 1. **ADD**

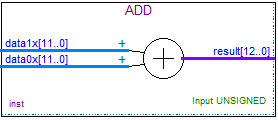
Виконує поєднання сигналів, що виходять з модулів Block. Реалізований на паралельному суматорі з IP Core від компанії Altera. На рисунку 3.16 зображене графічне позначення модуля. На рисунку 3.17 зображена симуляція цього модуля.

Рисунок 3.16 Модуль ADD



Рисунок 3.17 Модуль ADD

* 1. **UART\_RX**

UART – універсальний асинхронний приймач\передавач, виконує побітову передачу. Часто використовується для з’єднання комп’ютера з периферією. Усі сучасні мікроконтролери мають вбудований UART.

Модуль UARTRX реалізує лиш приймач від комп’ютера, для отримання параметрів синтезатора. UART використовує зсувний регістр як для отримання, так і для передачі.

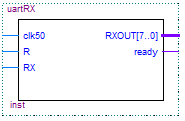


Рисунок 3.18 Модуль UART\_RX

* 1. **BUTTON\_PROC**

Так як ресурси ПЛІС обмежені, то встановлення 12 модулів Block для паралельної роботи усіх 12 клавіш неможливе. Це змушує робити модуль арбітра, що реалізований у BUTTON\_PROC. Цей модуль отримує сигнал BUSY від кожного Block’у. Якщо є вільний Block та була натиснута клавіша, то модуль починає її відтворювати, якщо ж усі Block’и зайняті, то реакції на натиск клавіші не буде. Використана мною ПЛІС має лише 6 тисяч логічних елементів, тому може вмістити 2 або 3 модуля Block. Також в залежності від натиснутої клавіші та поточного значення октави, Button\_Proc дасть потрібне значення частоти для генерування у модулі Oscill.

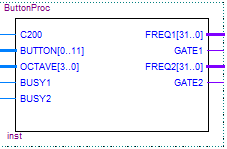


Рисунок 3.19 Модуль BUTTON\_PROC

* 1. **UART\_PROC**

Дані що приходять від комп’ютера на плату, складаються в байти у модулі UART\_RX. Ці байти потрапляють до даного модуля. Так як параметрів, що приходять багато, то треба реалізувати процедуру отримання адреси і далі вже самих даних, що складаються з 4 байтів. UART\_PROC працює за принципом автомату і формує адресу і дані, що далі потрапляють на потрібний вихід модуля. Синтезатор не обов’язково мусить бути підключений до ПК, так як цей модуль має стандартні налаштування , та 2 клавіші для ще 2 режимів налаштувань.

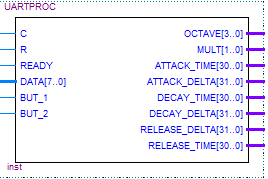


Рисунок 3.20 Модуль UART\_PROC

* 1. **ЦАП ЗА СХЕМОЮ R2R**

Так як ПЛІС працює з логічними сигналами, то отримання аналогового сигналу на його неможливе. Для цього використовується схема ЦАП (цифро-аналоговий перетворювач). ЦАП має декілька варіантів роботи: з паралельною входом і з послідовним входом. Як правило, послідовні ЦАП працюють за схемою дельта-сігма модуляції, де шириною імпульсу, регулюється вихідна напруга. Паралельний ЦАП можна реалізувати за допомогою матриці резисторів, де опір кожного резистора визначає вагу розряду. На рисунку 3.9 зображено приклад схема ЦАП на матриці R2R.

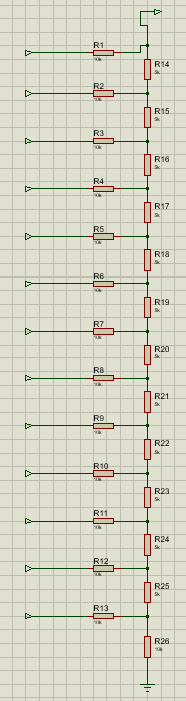
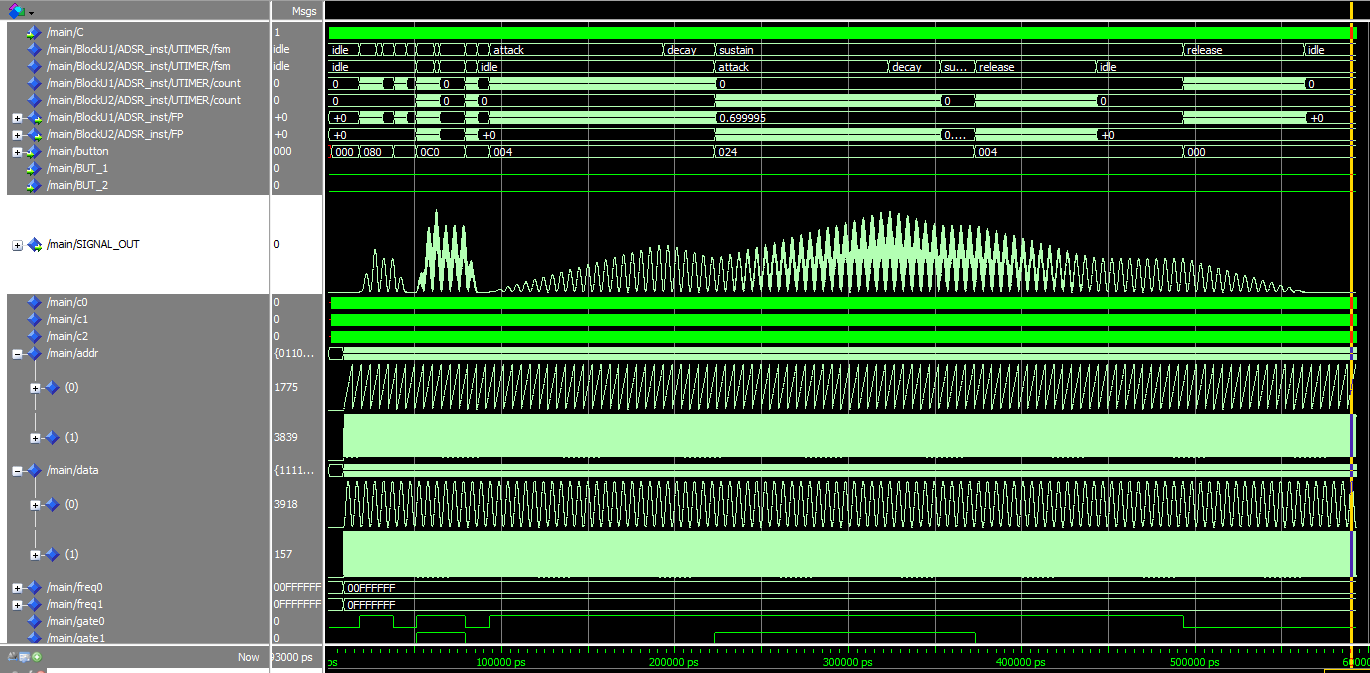
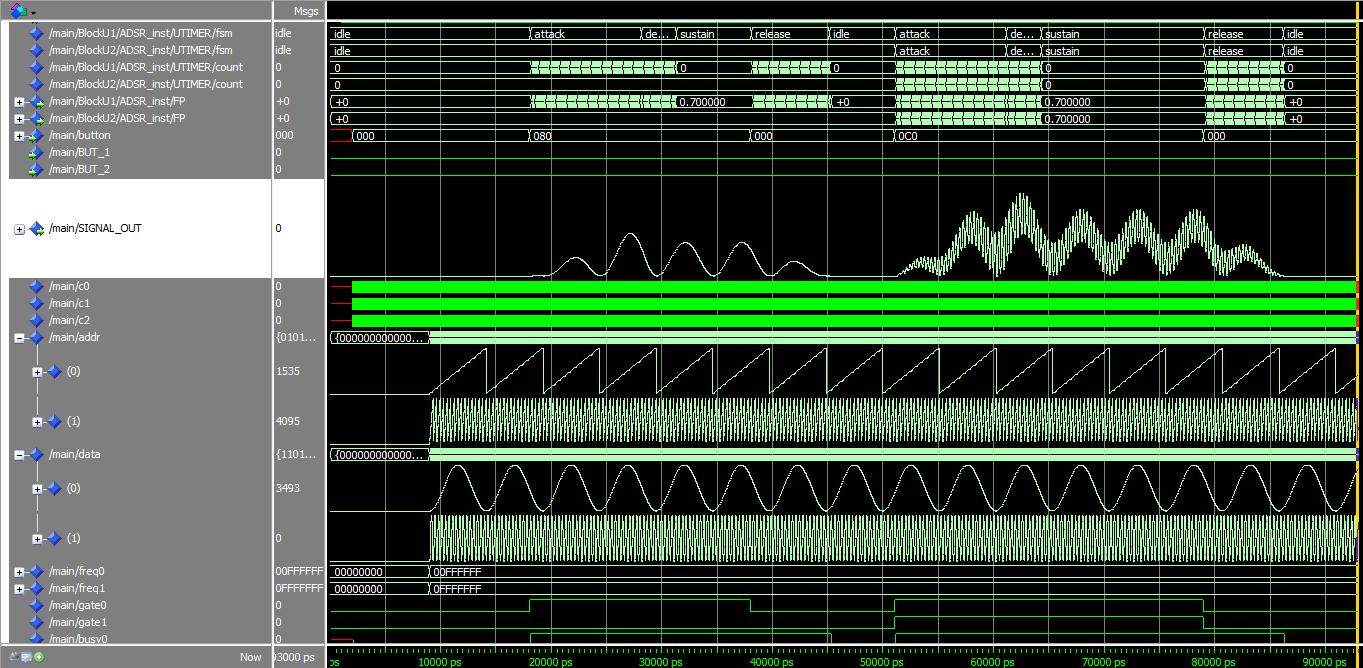
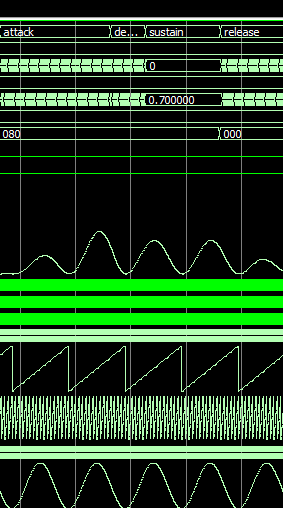
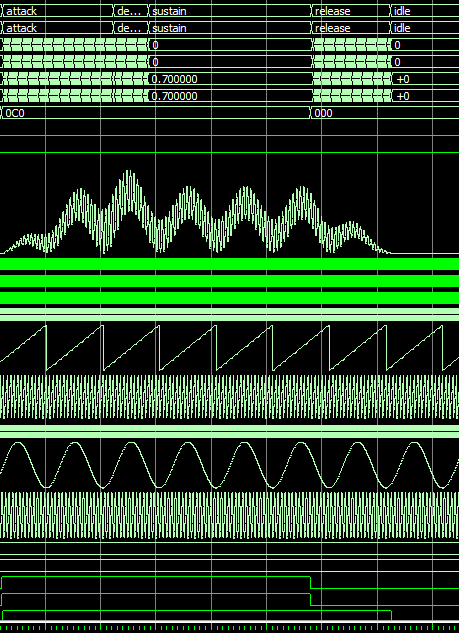
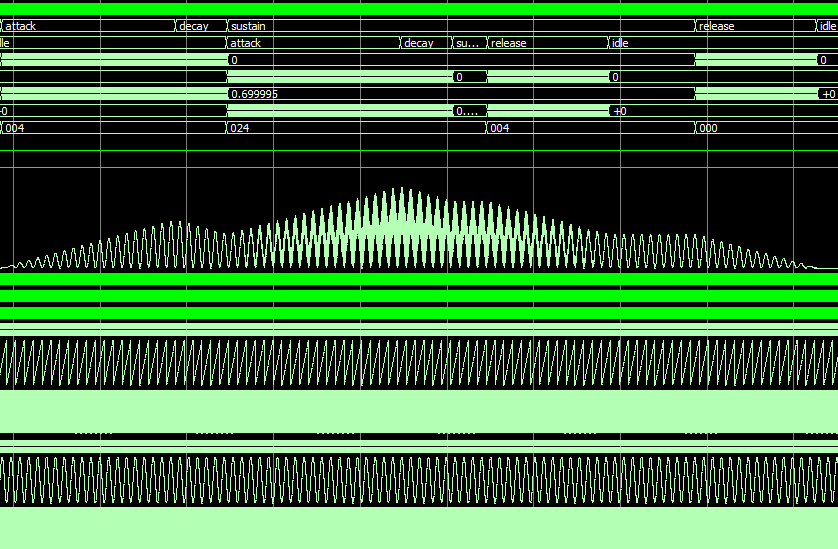


Рисунок 3.21 ЦАП

* 1. **РЕЗУЛЬТАТИ СИМУЛЯЦІЙ**

1. На рисунку 3.22 та 3.23 наведена діаграми симуляції усього проекту. Отриманні дані співпадають з очікуваними.

Рисунок 3.22

1. Рисунок 3.23
2. На рисунку 3.23 зображений збільшену ділянку діаграми. На рисунку зображені дві діаграми адрес та дві діаграми сигналу синусоїди. Спочатку була натиснута одна клавіша, про це свідчить рисунок 3.24.
3. 
4. Рисунок 3.24
5. Наступний рисунок 3.25 зображує дві натиснуті клавіші, частота другої у декілька разів більша за першу. Тому на перший сигнал наче наклали шум.
6. 
7. Рисунок 3.25
8. У наступній симуляції, що зображена на рисунку 3.22 були збільшені значення часу атаки та інших станів, про це свідчить більш довгий сигнал.
9. На рисунку 3.26 зображений збільшений фрагмент діаграми.
10. 
11. Рисунок 3.26

Спочатку у стан атаки було переведено перший блок, на діаграмі вихідного сигналу видно зростання простої синусоїди. Після того як перший блок пройшов стани атаки та спадання і перейшов у стан утримання, була натиснута друга клавіша і другий блок переведено у стан атаки. Ця діаграма доводить повноцінну та коректну роботу як окремих модулів, так і усього проекту в цілому.

Далі, на рисунках 3.27 та 3.28, наведені діаграми сигналу інших форм( пила та меандр).

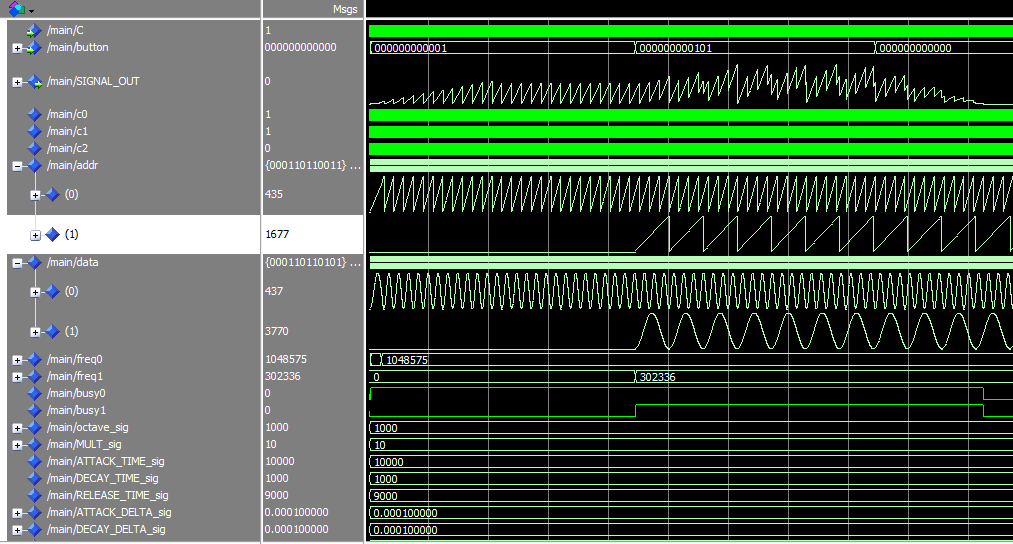


Рисунок 3.27



Рисунок 3.28

**ВИСНОВКИ ДО РОЗДІЛУ 3**

Увесь проект зайняв на кристалі ПЛІС:

4188 логічних елементів

2252 регістри

30 контактів IO

98524 біти пам’яті (12 бітів довжина слова\*212 кількість слів \* 2 кількість модулів)

14 вбудованих помножувачів з розрядністю 9 бітів

1 блок PLL

Звісно це є обмежена версія продукту. Маючи на кристалі більше логічних елементів у склад проекту можна включити до 12 блоків синтезу, що будуть обробляти кожну клавішу, цим саме можна звільнитися від арбітру клавіш. Також можна додати фільтри та різного роду ефекти. Маючи більше модулів пам’яті, можлива реалізація запису у цю пам’ять оригінальних сигналів замість звичайної синусоїди. Це дозволить синтезувати звук будь-якого реального інструменту.

**ВИСНОВКИ**

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. <http://opcb.kpi.ua/wp-content/uploads/2015/03/DR-%D0%A4%D0%98%D0%9E%D0%A2.pdf>
2. <https://www.intel.ru/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf>
3. <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-53001.pdf>
4. <http://www.digitalmusicacademy.ru/lesson-sound-as-a-physical-phenomenon>
5. <http://www.digitalmusicacademy.ru/lesson-sound-timbre>
6. <http://www.digitalmusicacademy.ru/lesson-synthesis-fundamentals>
7. <http://www.digitalmusicacademy.ru/lesson-additive-synthesis>
8. <http://www.digitalmusicacademy.ru/lesson-oscillator>
9. <http://www.digitalmusicacademy.ru/lesson-adsr-envelope>
10. <http://www.digitalmusicacademy.ru/lesson-adsr-envelope>
11. <http://www.russianelectronics.ru/leader-r/review/2192/doc/48223/>
12. <https://uk.wikipedia.org/wiki/%D0%9E%D0%BA%D1%82%D0%B0%D0%B2%D0%B0>
13. <https://uk.wikipedia.org/wiki/%D0%A1%D0%BC%D1%83%D0%B3%D0%BE%D0%B2%D0%B8%D0%B9_%D1%84%D1%96%D0%BB%D1%8C%D1%82%D1%80>
14. <https://uk.wikipedia.org/wiki/FM-%D1%81%D0%B8%D0%BD%D1%82%D0%B5%D0%B7>
15. <http://synthmusic.ru/articles/synthesistypes/subtractive/>
16. <http://synthmusic.ru/articles/synthesistypes/additive>
17. <http://virartech.ru/articles/wave_forms.php>
18. <https://uk.wikipedia.org/wiki/%D0%9D%D0%BE%D1%82%D0%B8>
19. <https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BC%D0%B1%D1%80>
20. <https://uk.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA>
21. <https://uk.wikipedia.org/wiki/%D0%90%D0%BC%D0%BF%D0%BB%D1%96%D1%82%D1%83%D0%B4%D0%B0>
22. <https://uk.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B0>

Додаток А

Додаток Б