ISSN 2522-9818 (print)
ISSN 2524-2296 (online) *Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)*

УДК 681.326 DOI: https://doi.org/10.30837/ITSSI.2024.27.192

О. Шкіль, Д. Рахліс, І. Філіпенко, В. Корнієнко, Т. Рожнова

АВТОМАТИЗОВАНЕ ПРОЄКТУВАННЯ ВБУДОВАНИХ СИСТЕМ ЦИФРОВОГО ОБРОБЛЕННЯ СИГНАЛІВ НА ПЛАТФОРМІ *SoC*

Об'єктом дослідження є процедури автоматизованого проєктування та аналізу алгоритмів цифрового оброблення сигналів на технологічній платформі *SoC*. **Предмет вивчення** – моделі, методи та процедури проєктування та оптимального вибору компонентів *SoC* для реалізації алгоритмів цифрового оброблення сигналів аудіоспектра. **Метою дослідження** є розроблення моделей та процедур для визначення можливостей компромісного розподілу обчислень алгоритмів оброблення сигналів у циклі автоматизованого проєктування на технологічній платформі *SoC* за критерієм продуктивності й доцільності використання апаратної та програмної реалізації алгоритмів. У статті розв'язуються такі **завдання**: розгляд процедур взаємодії процесорного ядра з програмованою логікою у складі систем на кристалі; розвиток процедур автоматизованого проєктування та аналізу систем оброблення сигналів із використанням мов програмування та мов опису апаратури для реалізації вбудованих систем. Упроваджуються такі **методи**: імплементація алгоритмів цифрового оброблення сигналів мовою програмування *C* та інструментів високорівневого синтезу для реалізації *IP*-блоків, діагностичний експеримент способом генерації тестових патернів сигналів та аналіз результатів оброблення на виході системи. **Досягнуті результати.** На основі аналізу процедур взаємодії процесорного ядра та програмованої логіки на обраній платформі *SoC* спроєктовано модель системи оброблення сигналів аудіоспектра. Практичну реалізацію виконано на базі стеку інструментальних засобів САПР *Vivado/Vitis/Vitis HLS*. Проведено верифікацію запропонованої моделі з використанням програмованого генератора тестових сигналів та аналізу отриманих характеристик цифрових фільтрів на виході системи. **Висновки.** У статті проаналізовано принципи проєктування вбудованих систем оброблення інформації, що реалізуються в системах на кристалі. Розглянуто принципи побудови та аналізу систем цифрового оброблення сигналів на базі систем на кристалі, що містять програмовану логіку та процесорну частину. Розроблені методи апробовано на алгоритмах *CIC*- та *FIR*-фільтрів на технологічній платформі *SoC FPGA* сімейства *ZYNQ*-7000 фірми *Xilinx*.

Ключові слова: вбудовані системи; системи на кристалі; *FPGA*; мова програмування *C*; алгоритми цифрового оброблення сигналів; аудіосигнали; цифрові фільтри.

Вступ

Сучасні вбудовані системи з мультимедійним складником дедалі частіше містять завдання цифрового оброблення сигналів. Завдання, що можуть бути виконані, полягають як у покращенні та стисканні відео й аудіо, так і в розпізнаванні та аналізі для подальшого оброблення. Зокрема типове завдання оброблення сигналу аудіоспектра поділяється на аналого-цифрове перетворення, оброблення інформації, видачу результатів на цифро-аналоговий перетворювач. Окремим сегментом виокремлюють апаратні та програмні реалізації аудіоефектів вбудованих платформ, зокрема фільтрації та симуляції луни. У сфері відеоконфереційного зв'язку гостро постає проблема формування променя для фільтрації сигналу (*beamforming*).

Технологічною платформою для реалізації вбудованих систем у сфері відеоконференцзв'язку є системи на кристалі (*System on Chip, SoC*), у яких інтегруються такі елементи, як процесор (процесори, зокрема спеціалізовані), пам'ять, кілька периферійних пристроїв, спеціалізовані обчислювальні блоки та їх з'єднання. Усе згадане вище становить оптимальний набір для деякого заздалегідь відомого функціонала, наприклад, оброблення та передачі відео- та аудіоінформації [1]. Розвиток інструментів проєктування для *SoC* сприяв новітній тенденції використання високорівневого синтезу для реалізації *IP*-блоків (*intellectual property*), що можуть бути синтезовані на *FPGA*-частині (*Field Programmable Gate Array*) системи на кристалі та інтегровані до платформи. Для реалізації алгоритмів цифрового оброблення сигналів традиційним для *SoC* вважається реалізація на *FPGA*-частині. Але через обмеження платформи та можливу відсутність часткової реконфігурації та затримки, що можуть бути нею викликані, постає актуальне питання оптимального розподілу обчислень між апаратною та програмною частиною

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) ISSN 2524-2296 (online)

SoC в умовах реалізації алгоритмів для оброблення в ділянці звукового спектра частот.

Аналіз останніх досліджень і публікацій

У статті [2] розглянуто використання програмованих логічних інтегральних схем *FPGA* у сфері оброблення цифрових сигналів звуку в режимі реального часу. Визначено переваги застосування *FPGA* для досягнення високої продуктивності та низького часу затримки з метою оброблення аудіосигналів. Розглянуто можливості масштабування рішення для розв'язання завдань у багатоканальних системах оброблення.

Приділено увагу класифікації в реальному часі сигналів *FM*-діапазону (*Frequency Modulation*) у роботі [3], також проаналізовано й запропоновано визначення *MFCC*-ознак (*Mel-Frequency Cepstral Coefficients*, коефіцієнти мел-частотного кепстрального перетворення) перед виконанням алгоритму класифікації. Запропонована реалізація дає змогу в реальному часі виконувати автоматичне індексування аудіоданих із *FM*-діапазону. Практичну реалізацію запропонованих рішень виконано на базі *FPGA* сімейства *Virtex*-6 від *Xilinx*. Окремо в статті проаналізовано використання ресурсів платформи та отримані параметри швидкодії системи.

У дослідженні [4] розглянуто питання побудови прототипу системи активного шумозниження *ANC* (*Automatic Noise Cancellation*) із застосуванням апаратних засобів на основі *FPGA*. Порушено питання вибору реалізації на базі рухомої та фіксованої точки та методи вирішення практичних проблем, які спостерігалися під час виконання роботи. Отримано та проаналізовано характеристики побудованої системи як у разі усунення шуму у вузькому спектрі, так і широкосмугового шуму.

Робота [5] демонструє реалізацію модульної системи звукових ефектів затримки на платформі *SoC*. Для виконання проєкту використовувалася плата розроблення *ZedBoard*. Програмована логіка на процесорі *Zynq* застосовувалася для оброблення сигналу, а *ARM*-частина (*Advanced RISC Machine*) використовувалася для зв'язку між програмованою логікою та програмним забезпеченням, запущеним на комп'ютері, для керування різними параметрами звукових ефектів у реальному часі.

Інтеграцію бібліотеки *FFMPEG* до платформи *ZYNQ* розглянуто й проаналізовано в доповіді [6]. На стороні системи оброблення (*processing system, PS*) *ZYNQ* використовується відкритий код бібліотеки *FFmpeg* для розбору аудіоданих із мережі у форматі *MP3* з подальшим розпаковуванням у формат *PCM* (*Pulse Coded Modulation*), які передаються на сторону програмованої логіки (*programmable logic, PL*) *ZYNQ* через *DMA* (*Direct Memory Access*). У роботі наведено використання вбудованого *IP*-ядра логічного аналізатора *Vivado ILA (Integrated Logic Analyzer)* для відтворення даних, що надходять у *PCM*-форматі. Проаналізовано отриману продуктивність і використані ресурси платформи.

У праці [7] розглядається реалізація оброблення аудіосигналів у режимі реального часу на *FPGA* з аналізом попередніх рішень у вигляді реалізації *Audio IPs*, що розроблялися вручну на *VHDL* або використовувалися заздалегідь визначені *IPs* із середовища розроблення. Наведено приклад використання високорівневого синтезу (*HLS*), який дає змогу здійснювати потік компіляції від високорівневих специфікацій оброблення аудіосигналів *DSP* (*digital signal processing*) до бітових потоків *FPGA*. У статті презентуються принципи та реалізація першого "компілятора аудіо *DSP*", призначеного для *FPGA*. Виконано практичний експеримент із побудовою комплексної системи оброблення аудіосигналів у реальному часі.

У роботі [8] запропоновано використання алгоритму двовимірного швидкого перетворення Фур'є (*2D-FFT*) для вивчення багатьох сучасних систем, застосовуваних у сфері безпеки та біометрії. Упровадження цього алгоритму, що є обчислювально-інтенсивною задачею, обмежене через складність його апаратного проєктування. Перша мета цієї роботи – відокремити вплив апаратно-програмного спільного проєктування (*Hw/Sw co-design*) на час оброблення та використання ресурсів. По-друге, пропонується інноваційна архітектура для алгоритму *2D-FFT*, протестована на *SoC Zynq*, що вимагає менше часу оброблення та пам'яті порівняно з традиційним алгоритмом. У цій статті запропоновано три реалізації алгоритму *2D-FFT* з використанням *SoC Zynq*. Перша основана на процесорі *ARM*. Друга – це рішення з апаратно-програмним проєктом традиційного алгоритму *2D-FFT* на гібридній платформі, яка поєднує процесор *ARM Cortex-A9* з *FPGA*. Третя також є рішенням з апаратно-програмним ко-дизайном, що використовує оптимізований алгоритм *2D-FFT* для аналізу в реальному часі високороздільних зображень.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

Визначено продуктивність запропонованих методів і наведено фрагменти практичних реалізацій.

Деталі реалізації та дослідження особливостей оброблення зображень у реальному часі на платформі *FPGA* наведено в статті [9], де виконано реалізацію типових алгоритмів оброблення зображень на основі *HLS*-інструменту. Проаналізовано продуктивність і окреслено особливості реалізації таких алгоритмів, як ерозія, лінійна фільтрація та розширення.

Розгляд питання реалізації класичних алгоритмів *DSP* в аудіоспектрі запропоновано в дослідженні [10], де подано реалізації класичних звукових ефектів, реалізованих на *FPGA*. Поєднуючи методи цифрового оброблення сигналів *DSP* з можливостями мови *VHDL*, пропонуються ефективні архітектури з погляду використання ресурсів. Застосовується чип *Artix* 7 від *Xilinx* разом із *Xilinx Vivado Design Studio* 2020.1. Проаналізовано результати використання ресурсів платформи.

Робота [11] демонструє переваги методології ко-дизайну *HW/SW* на платформі *SoC* для оброблення звуку. Час розроблення та продуктивність проєктування суттєво покращено завдяки створенню блокових конструкцій. Вивчено можливості використання середовищ *Matlab/Simulink* для проєктування обраних аудіоефектів та їх подальшого синтезу у вигляді *IP*-ядер на обраній платформі.

У праці [12] розглядається реалізація алгоритму *Big Bang-Big Crunch (BB-BC)* на платформі *FPGA Xilinx Virtex-5*. Пропонується реалізація паралельної архітектури обчислення для фаз *BB-BC* алгоритму та виконується порівняльний аналіз із програмною реалізацією на платформі *CUDA* й високорівневим синтезом із використанням *Vitis HLS*.

Проблема застосування довгої арифметики на платформі *FPGA* досліджується в роботі [13]. Запропоновано вирішення проблеми завдяки використанню фреймворку *IMpress*, що автоматично обирає оптимальні вирази на основі вимог до ресурсів певної програми. *IMpress* автоматично перетворює вирази множення цілих чисел у поведінкові описи мовою *C++* та ініціює компіляцію для платформи *FPGA* з високорівневого синтезу. *IMpress* пропонує високий контроль над використанням і балансом ресурсів платформи.

Процес реалізації та ко-дизайну трансиверу *WiFi* діапазону з використанням *HLS* аналізується в дослідженні [14]. Проаналізовано продуктивність отриманої реалізації з аналогом мовами *VHDL/Verilog* та діагностичний експеримент на практиці з використанням *SDR*-приймача (*Software-Defined Radio*). Наведено висновки щодо доцільності застосування *HLS*-синтезу на платформі *SoC* для високошвидкісного оброблення сигналів.

Дослідження процесу побудови алгоритмів оброблення відео та аудіо в реальному часі наведено в праці [15], де використано плату налагодження *ZedBoard* як платформу для реалізації запропонованих рішень. У роботі описано покрокову реалізацію системи та досягнуто результатів продуктивності, беручи до уваги реалізацію алгоритмів оброблення аудіо на *Verilog* і відео частини на *VHDL*.

З огляду на наявні роботи та завдання у сфері реалізації цифрових фільтрів разом із зростанням обчислювальних потужностей *SoC* доцільно визначити оптимальний розподіл обчислювальних витрат алгоритму фільтрації між *PL*- і *PS*-частинами *SoC*.

Як приклад реалізації цифрових фільтрів на платформі *SoC* розглядається гребінчастий фільтр та фільтр нижніх частот із кінцево-імпульсною характеристикою. Метою дослідження є розроблення моделей та процедур для визначення можливостей компромісного розподілу обчислень алгоритмів оброблення сигналів у циклі автоматизованого проєктування на технологічній платформі *SoC* за критерієм продуктивності та доцільності використання апаратної та програмної реалізації алгоритмів. Як *SoC* застосовується *Xilinx ZYNQ* 7020 з платою налагодження *ZedBoard*.

Проєктування та тестування цифрових фільтрів

Загалом вихід більшості алгоритмів *DSP* може бути описано як

$$y[n] = T\{x[n]\}, \quad (1)$$

де $y[n]$ – значення на виході системи;

$x[n]$ – вхідне значення;

T – функція оброблення.

Для лінійних дискретних систем є загальновизначеним вираз згортки ($\star$) з метою отримання вихідного значення системи [18–20]. Різнецеве рівняння для лінійної системи, що не залежить від часу (*Linear Time Invariant*), має вигляд

$$y[n] = T\{x[n]\} = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = h[n] \star x[n], \quad (2)$$

де $h[k]$ – імпульсна характеристика фільтра;

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

$h[n]$ – імпульсна характеристика фільтра, яка згортається із вхідними відліками $x[n]$.

Для задач реалізації фільтрів із кінцево-імпульсною характеристикою із сімейства алгоритмів *DSP* необхідно реалізувати одновимірну згортку (операцію *convolution*) з імпульсною характеристикою фільтра. Приклад операції згортки наведено на рис. 1 [21].

Для проєктування фільтрів та отримання необхідних значень коефіцієнтів використано програмний пакет *MATLAB Online* та інструмент *filterDesigner* (рис. 2).

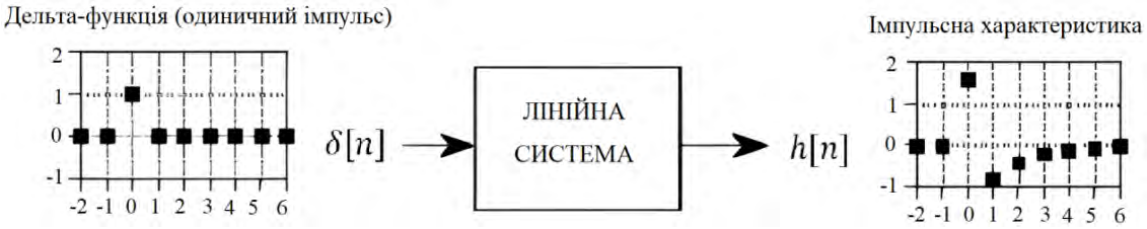

Рис. 1. Візуалізація згортки з дельта-функцією

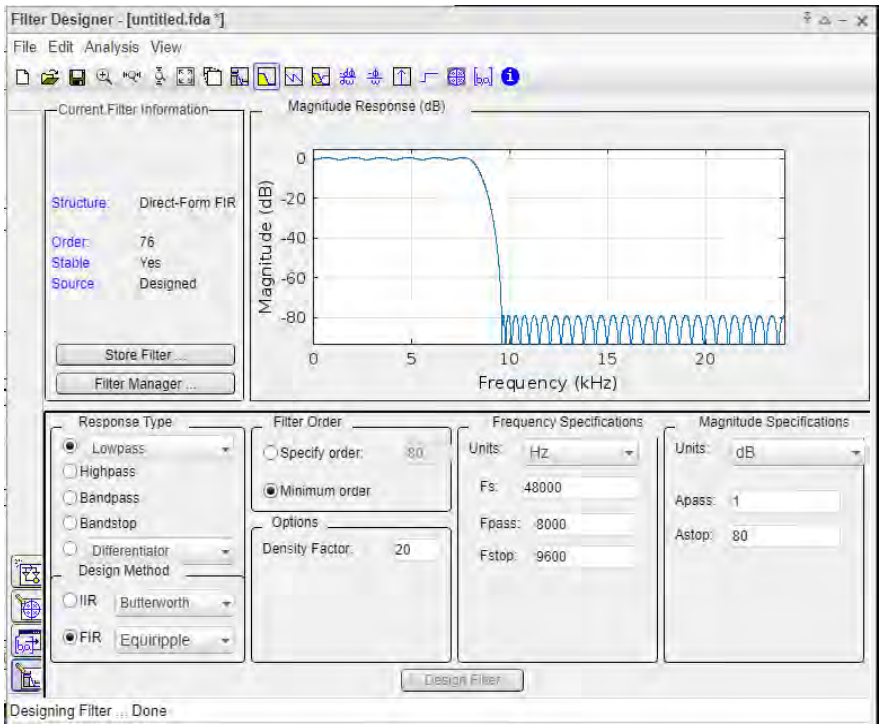

Рис. 2. Середовище *Filter Designer* у *MATLAB* для визначеного КІХ-фільтра

Filter Designer дає змогу обрати необхідний тип фільтра з кінцево-імпульсною характеристикою (КІХ, *FIR*) або безкінечно-імпульсною характеристикою (БІХ, *IIR*) та частотні параметри фільтра.

У разі реалізації КІХ-фільтра є можливість обрати початкову (*Fpass*) та кінцеву (*Fstop*) частоту пропускної смуги (*transition band*). Для отримання результатів продуктивності було реалізовано *FIR*-фільтр нижніх частот для частотного діапазону людського голосу з мінімальними викривленнями, що можуть бути помітні. Параметри фільтра були відповідно обрані *Fpass* = 8 *KHz* та *FStop* = 9600 *KHz*.

З метою тестування моделі фільтра використано інтерактивне середовище розробки *Jupyter Notebook* з візуалізацією на базі бібліотеки *Matplotlib*. Тестування виконано шляхом застосування вбудованих функцій *lfiter* та отримання спектрів вхідного та вихідного сигналів із застосуванням перетворення Фур'є на сигналі до і після фільтрації. До фільтрації вхідний сигнал був створений із синусоїдальних компонент визначеного набору частот, а саме 50, 300, 5000, 9000, 10000, 12000, 15000 Гц відповідно з обраною частотою дискретизації 48 КГц. На рис. 3 зображено отримані спектри вхідного та вихідного сигналів.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

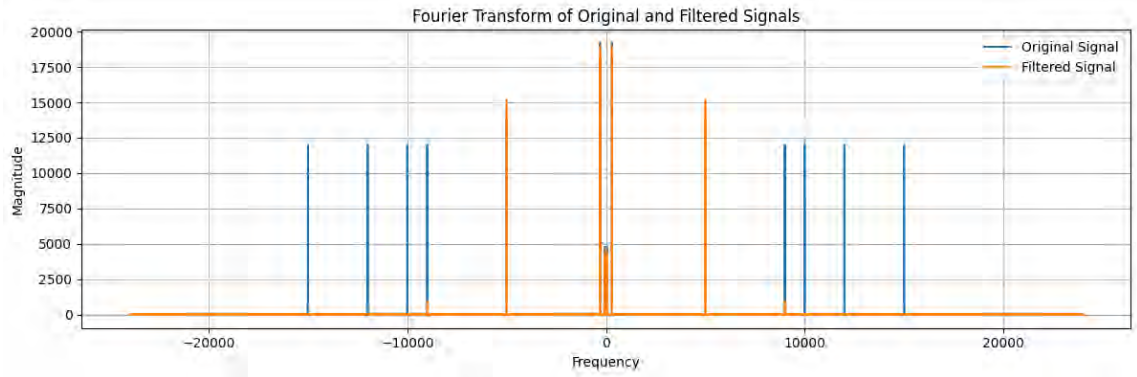

Рис. 3. Тестування КІХ-фільтра з використанням *Jyputer Notebook* та *Matplotlib*

Порівняльний аналіз апаратної та програмної реалізації фільтрів

Платформа *Xilinx ZYNQ* містить частини *PL* і *PS*. *PS*-частина реалізована у вигляді двох повноцінних *ARM Cortex A*9 ядер із технологією *ARM Neon* для реалізації інструкцій *Single Instruction Multiple Data (SIMD)*. Отже, є можливість використовувати для реалізації *DSP*-алгоритмів як *ARM*-частину без застосування *SIMD*-інструкцій, так і з ними. Окремо виокремлюється частина *PL*, на якій можуть бути реалізовані апаратно спеціалізовані *IP*-ядра для потреб користувача та реалізована взаємодія з *PS*-частиною з використанням шини *AXI (Advanced eXtensible Interface)*. Технологія *Arm Neon* – це вдосконалене розширення архітектури *SIMD* для процесорів *A-profile* і *R-profile*. Регістри *Neon* розглядаються як вектори елементів одного типу даних з інструкціями *Neon*, що діють на кілька елементів одночасно. Підтримується декілька типів даних, зокрема операції з рухомою точкою та цілочисельна арифметика. Технологія *Neon* призначена для покращення мультимедійної взаємодії з користувачем шляхом прискорення кодування та декодування аудіо та відео, реалізації інтерфейсів користувача, двовимірної та тривимірної графіки, ігор. *Neon* також може прискорити алгоритми оброблення сигналів і функції для таких завдань, як оброблення аудіо та відео, розпізнавання голосу й обличчя, комп'ютерний зір і "глибоке" навчання (*deep learning*) [17].

Робота з прискорювачем *NEON* у *ARM* є можливою або з використанням *Intrinsic* функцій компілятора, або із застосуванням офіційних бібліотек. Одним із варіантів є розроблення на базі бібліотеки *NE*10 від офіційного виробника. *NE*10 містить як програмну реалізацію типових алгоритмів *DSP*, так і прискорену з використанням *NEON*. Залежно від наявності *NEON* на цільовому ядрі обираються відповідні реалізації. Додатково, під час компіляції бібліотеки потрібно вказати необхідний доступний функціонал і рівень оптимізації. У нашій ситуації було вказано максимальний рівень оптимізації компілятора – O3 – та додатково було долучено *-ffast-math* як для бібліотеки, так і для програмної частини застосунку. Окремо було додано прапорець (*flag*) збірки специфічної для *ARM*-частини *ZYNQ*.

Для реалізації функціоналу *FIR*-фільтра на базі бібліотеки *NE*10 необхідно ініціалізувати *NE*10 та встановити параметри фільтра. Зокрема вказати *state_buffer*, у якому зберігаються проміжні результати оброблення на базі структури циклічного буферу. Після ініціалізації бібліотеки та структури фільтра необхідно викликати *ne_10_fir_float*, до якого потрібно передавати отриманий вхідний блок і конфігураційну структуру фільтра.

З метою визначення продуктивності апаратної реалізації було виконано *FIR*-фільтри з використанням константного набору коефіцієнтів для отримання оптимізації *constant multiplier propagation* та додаткової оптимізації із застосуванням властивості симетрії коефіцієнтів. Через те що коефіцієнти фільтра є симетричними щодо центрального відліку, було реалізовано підхід, описаний у дослідженні [22], а саме зменшення кількості операцій множення завдяки використанню симетрії коефіцієнтів. Цей підхід дає змогу отримати більш оптимальну реалізацію після високорівневого синтезу.

У разі непарної кількості коефіцієнтів центральний коефіцієнт потрібно розраховувати окремо, оскільки немає операції попереднього додавання. Для фільтра з п'яти коефіцієнтів $(0.078, 0.253, 0.335, 0.253, 0.078)$ припустимий масив

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

коефіцієнтів $h[5]$, які є симетричними щодо центрального коефіцієнта:

$$ac_{fixed}\langle 8.1\rangle h[5] = \{0.078, 0.253, 0.335, 0.253, 0.078\},$$

де $ac_{fixed}\langle 8.1\rangle$ – це вбудований тип даних для реалізації операцій арифметики з фіксованою точкою, де 8 – розмірність даних, 1 – кількість біт для зберігання цілої частини.

Розгортання циклу множення-складання (*multiply-accumulate, MAC*) має такий вигляд:

$$temp = h[0] \star regs[0] + h[1] \star regs[1] + h[2] \star regs[2] + h[3] \star regs[3] + h[4] \star regs[4],$$

де $regs[i]$, $i = \overline{0,4}$ – це буфери для зберігання вхідних значень сигналу x.

Тоді для прикладу фільтра з п'яти коефіцієнтів можна записати:

$$temp = 0.078 \star (regs[0] + regs[4]) + 0.335 \star regs[2] + 0.253 \star (regs[1] + regs[3]).$$

Діаграму КІХ-фільтра із вказаною оптимізацією наведено на рис. 4.

Особливостями реалізації можна вважати використання типів *ap_fixed* та *shift_class* для реалізації циклічного буферу. Для передачі вхідних семплів (дискретних значень сигналів) застосовано інтерфейси *AXI-Lite* та *AXI-HP*. Після синтезу *IP*-ядра для обраного *SoC* було отримано структуру викликів, зображену на рис. 5, після оптимізації *Vitis HLS*.

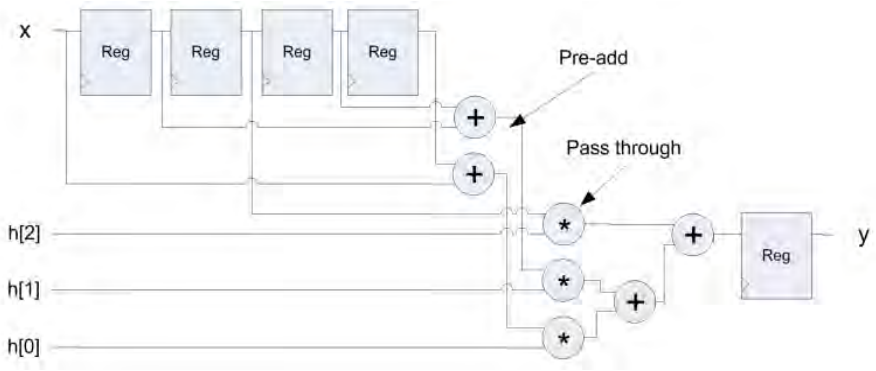

Рис. 4. Структура фільтра з непарною симетрією коефіцієнтів

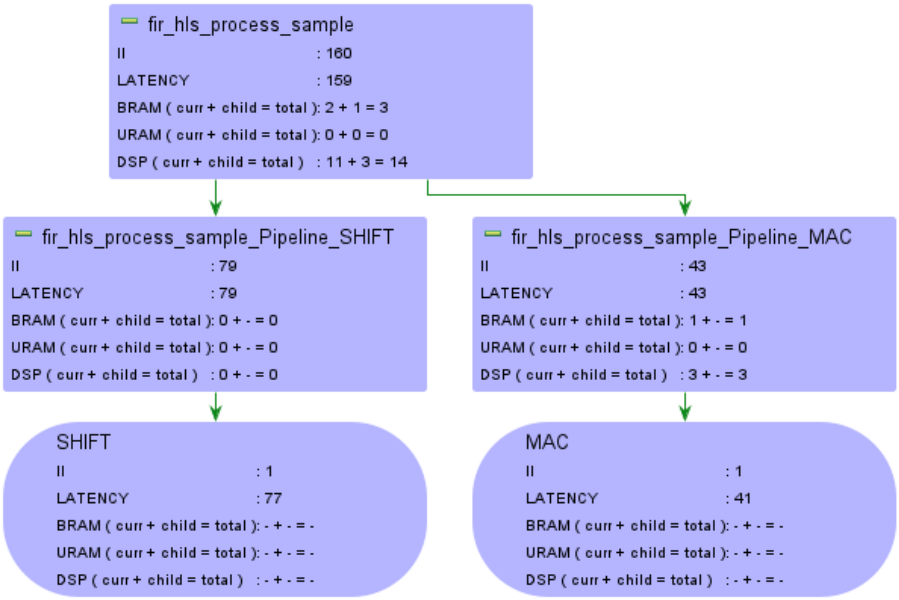

Рис. 5. Діаграма викликів *FIR*-фільтра в середовищі *Vitis HLS*

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

Гребінчастий фільтр реалізується на базі кільцевого буферу для побудови лінії затримки та опційних блоків коефіцієнтів для вхідних і вихідних семплів сигналу. Загальна структура гребінчастого фільтра зі зворотними зв'язками (*feedforward*) зображена на рис. 6.

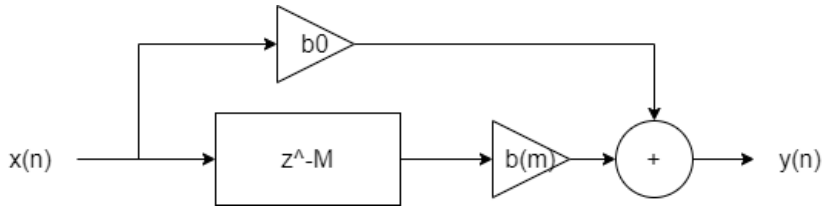

Рис. 6. Загальна структура *feedforward* гребінчастого фільтра

На рис. 6 $b0$ – це коефіцієнт множення вхідних семплів; *z^-M* – затримка вхідного сигналу на M семплів; $b(m)$ – послаблення затриманих семплів. Отже, прямий сигнал $x(n)$ подається в обхід та на лінію затримки. Вихід $y(n)$ є лінійною комбінацією прямого та затриманого сигналу [23]. Для реалізації гребінчастого фільтра було реалізовано кільцевий буфер з метою збереження вхідних семплів та формування лінії затримки. Окремо впроваджено методи для встановлення довжини лінії затримки та передачі вхідних семплів сигналу. Для реалізації профілювання часу виконання функцій було реалізовано макрос, що використовує вбудований таймер *ARM A9* через виклик *XTime_GetTime()* з *Vitis SDK*.

Методика ко-дизайну для розроблення на технологічній платформі *SoC*

Метод ко-дизайну в умовах використання *SoC* є схожим із типовим процесом проєктування для системи на кристалі. Зокрема етапи проєктування передбачають розроблення та написання *testbench* для окремих *IP*-блоків оброблення даних. Подальші етапи містять інтеграцію до *System Block Design* розроблених блоків і налаштування процесорної частини *ZYNQ*. Для взаємодії з процесорним ядром типовим є використання шини *AXI-Lite* для конфігураційних параметрів блоків або передачі незначних обсягів інформації. Для обміну даними, що є критичними за часом їх оброблення, доцільно застосовувати шини *AXI Stream* для пересилання між *IP*-ядрами та *DMA+DDR* з метою взаємодії між процесорною та *FPGA*-частинами системи. Програмна частина пристрою розробляється в середовищі *Vitis IDE*, створення та тестування *IP*-блоків виконується у *Vitis HLS*. Фінальна інтеграція апаратної частини відбувається в середовищі *Vivado IDE*. Зазначимо, що платформа підтримує як можливе використання *PetaLinux* для створення образу системи, так і можливу роботу в режимі *baremetal* або із застосуванням *FreeRTOS* як операційної системи реального часу.

Основні деталі типового маршруту проєктування для платформи *SoC* наведено в роботі [16], де викладено основні етапи створення проєкту вбудованої системи з особливостями налагодження програмного забезпечення та фінальної інтеграції системи.

Архітектура системи

З метою реалізації оброблення алгоритмів сигналів у аудіоспектрі на апаратній платформі *SoC* необхідно формувати відповідну архітектуру системи. Зокрема потрібно визначити можливість використання алгоритмів, розроблених на *PS*-частині до переносу їх у середовище *Vitis HLS*, і тестування у вигляді *IP*-блоків. Для перевірки та порівняльного аналізу використано середовище аналізу *ARTA*. З інтерфейсних блоків, що були реалізовані, є трансивер *I2S* аудіоінтерфейсу, блок *AXI GPIO* для задання адреси аудіокодеку та переключення фільтрів під час роботи системи, блок *AXI-I2C* для первинної конфігурації аудіокодеку та призначення вхідних і вихідних інтерфейсів аналого-цифрових та цифро-аналогових перетворювачів (АЦП та ЦАП), що вбудовані в кодек. Також окремо застосовано блок *UART* для видачі діагностичних повідомлень. Реалізацію фільтрів виконано на базі бібліотеки *NE*10, для синтезу *IP*-блоків використано *Vitis HLS*, зважаючи на *high-level-synthesis*-специфічні особливості. Тестування фільтрів виконувалося в середовищі *Scipy* разом з *Jupyter Notebook* і розробленими скриптами. На рис. 7 зображено архітектуру розробленої системи на базі *ZYNQ* та плати налагодження *ZedBoard* для експериментального дослідження.

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

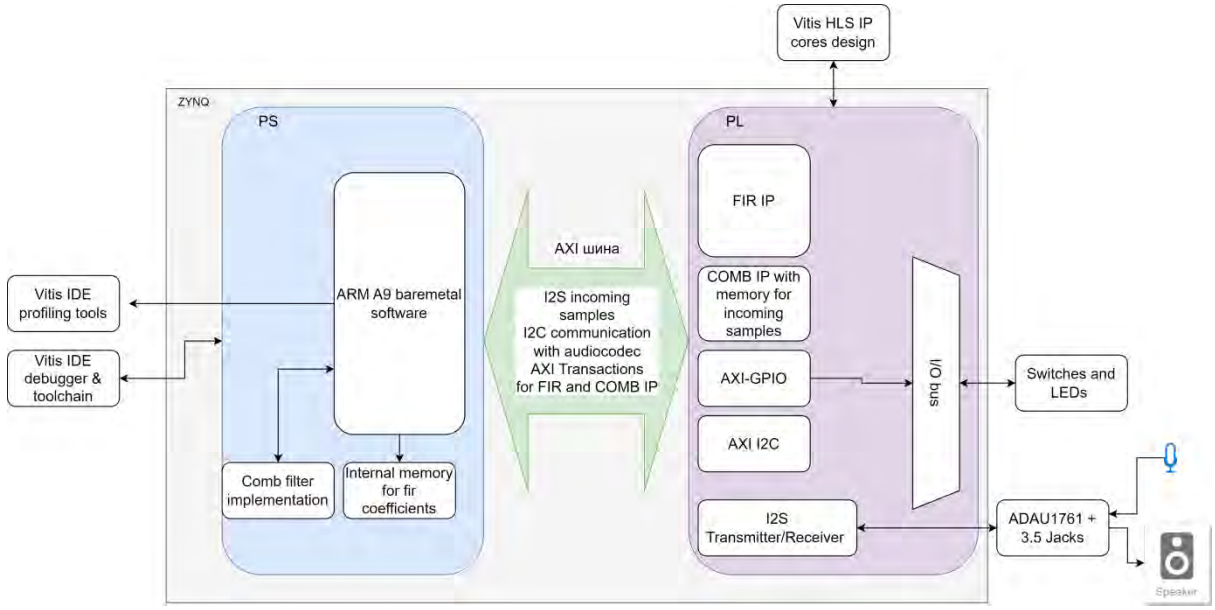

Рис. 7. Архітектура розробленої системи на базі *SoC* та плати налагодження *ZedBoard*

Розроблену архітектуру реалізовано в середовищі *Vivado IDE* з інтеграцією *IP*-ядер гребінчастого та *FIR*-фільтрів. Було додано блоки *I2S/I2C* периферії. Діаграму реалізованої системи, отриманої в середовищі *Vivado*, наведено на рис. 8.

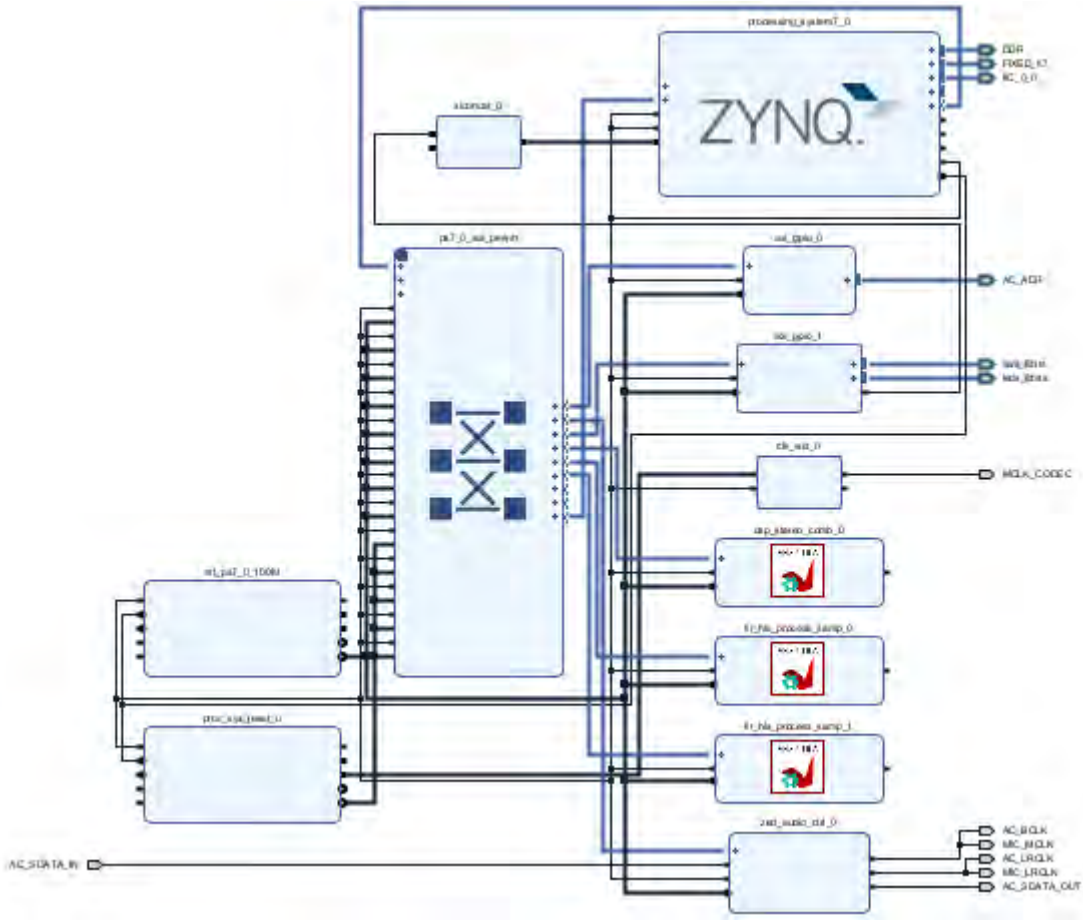

Рис. 8. *Block Diagram* розробленої архітектури на базі плати налагодження *ZedBoard*

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

Результати досліджень та їх обговорення

У табл. 1 подано результати, отримані для реалізованих моделей фільтра нижніх частот і гребінчастого фільтра з використанням бібліотеки *NE*10 з підтримкою *ARM NEON, AXI-Lite* та шин *AXI-HP*.

Отже, незважаючи на традиційне припущення щодо досягнення кращих результатів швидкодії в разі перенесення обчислень на апаратну частину *SoC (PL)*, у конкретному випадку оброблення *sample-by-sample* без буферизації та блокового оброблення для конкретних фільтрів аудіоспектра досягнуто результат. А саме програмна реалізація з використанням вбудованої бібліотеки *NE*10 для роботи з векторними інструкціями *SIMD* бо нативна реалізація може давати кращі часові результати без необхідності перенесення обчислення на апаратну частину. У такому разі *SoC ZYNQ* може застосовуватися як вбудована апаратна платформа з набором периферії, що може бути додана та синтезована за необхідності в частині *PL*.

Таблиця 1. *Результати тестування фільтрів низьких частот і гребінчастого фільтра*

Тип фільтра	**КІХ *PS*-частина**		**КІХ *PL*-частина**			**Гребінчастий фільтр**	
	з *NE*10	без *NE*10	без оптимізації парності коефіцієнтів, шина *AXI4-HP*	без оптимізації симетрії коефіцієнтів, шина *AXI4-Lite*	з оптимізацією симетрії коефіцієнтів, шина *AXI4-Lite*	(програмна реалізація)	(апаратна реалізація)
Використання апаратних ресурсів	–, *PS*-частина	–, *PS*-частина	*DSP:14 FF:2277 BRAM:3*	*DSP:14 FF:1441 BRAM:3*	*DSP:14 FF:1475 BRAM:3*	–, *PS*-частина	*LUT:694 FF:613 BRAM:8*
Час виконання на платформі *SoC ZYNQ*	1.85 us	3.51 us	3.16 us	2.95 us	2.65 us	0.15 us	2.02 us

Висновки та перспективи подальшого дослідження

Унаслідок проведених досліджень розроблено алгоритми фільтрації для вбудованих систем оброблення інформації на платформі *SoC ZYNQ* як у блоці *PS* мовою програмування *C*, так і в блоці *PL* з використанням *Vitis HLS* з метою визначення можливостей компромісного розподілу ресурсів. Під час верифікації здійснено діагностичний експеримент способом генерації тестових шаблонів (патернів) сигналів та аналізу отриманих характеристик фільтрів на виході системи. Тестові послідовності подавалися з допомогою програмованого генератора сигналів. Фіксування результатів діагностичного експерименту здійснювалось завдяки програмному забезпеченню *Arta* та монітору послідовного терміналу *MobaXterm* з метою отримання спектрограм вихідного сигналу та результатів профілювання алгоритмів. Досягнуті результати аналізувалися як за часом виконання алгоритмів, так і за якісними показниками застосування апаратних і програмних ресурсів. Аналіз показав, що використання *PS*-частини для оброблення *sample-by-sample* було найбільш ефективним. Подальші напрями досліджень передбачають визначення продуктивності реалізації алгоритмів, таких як придушення луни, виділення спектра сигналу людського голосу, реалізацію аудіоефектів, визначення доцільного розподілу обчислень у багатоканальних системах оброблення звуку, реалізацію *DOA (Direction Of Arrival)* сімейства алгоритмів для мікрофонних матриць та аналіз реалізації алгоритмів компресії на базі кодеків *OPUS* і *Vorbis*.

Список літератури

1. Lee E. A., Seshia S. A. Introduction to embedded systems: A cyber-physical system approach. MIT Press, 2017. 564 p. URL: https://ptolemy.berkeley.edu/books/leeseshia/releases/LeeSeshia_DigitalV2_2.pdf (дата звернення: 04.02.2024).
2. Popoff M., Michon R., Risset T., Orlarey Y., Letz S. Towards an FPGA-based compilation flow for ultra-low latency audio signal processing. *Proceedings of the 19th Sound and Music Computing (SMC-22),* June 5–12, Saint-Étienne, France. 2022. P. 555–562. DOI:10.1109/ASAP57973.2023.00018
3. Wassi G., Iloga S., Romain O., Granado B., Tchuenté M. FPGA-based simultaneous multichannel audio processor for musical genre indexing applications in broadcast band. *Journal of parallel and distributed computing*. 2018. Vol. 119. P. 146–161. DOI: 10.1016/j.jpdc.2018.02.011
4. Wang T., Bohanan S. Active noise cancellation with FPGA – practical considerations. *INTER-NOISE and NOISE-CON congress and conference proceedings (NOISE-CON23)*. May 15–18, Grand Rapids, USA. 2023. Vol. 266, no. 1. P. 1036–1043. DOI: 10.3397/NC_2023_0124

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) ISSN 2524-2296 (online)

5. Cannon D., Fang T., Saniie J. Modular Delay Audio Effect System on FPGA. *IEEE International Conference on Electro Information Technology (eIT)*. May 19–21, Mankato, USA. 2022. P. 248–251. DOI: 10.1109/eIT53891.2022.9813875
6. Xie W., Yang F. Design and implementation of audio stream processing based on ZYNQ. *IEEE 6th information technology and mechatronics engineering conference (ITOEC)*, March 4-6, Chongqing, China. 2022. P. 589–592. DOI: 10.1109/ITOEC53115.2022.9734347
7. Popoff M., Michon R., Risset T., Cochard P., Letz S., Orlarey Y., Dinechim de F. Audio DSP to FPGA Compilation: The Syfala Toolchain Approach. Research report №9507. Grame, Emeraude: Inria. 2023. 18 p. URL: https://hal-lara.archives-ouvertes.fr/hal-04099135/ (дата звернення: 04.02.2024).
8. Kortli Y., Gabsi S., Jridi M., Alfalou A., Atri M. Hw/Sw Co-Design technique for 2D fast fourier transform algorithm on Zynq SoC. *Integration.* 2021. Vol. 82. P. 78–88. DOI: 10.1016/j.vlsi.2021.09.005
9. Azzaz M., Maali A., Kaibou R., Kakouche I., Mohamed S., Hamil H. FPGA HW/SW codesign approach for real-time image processing using HLS. *1st International Conference on Communications, Control Systems and Signal Processing (CCSSP'20),* May 16–17, EL OUED, Algeria. 2020. P. 169–174. DOI: 10.1109/CCSSP49278.2020.9151686
10. Dragoi C., Anghel C, Stanciu C., Paleologu C. Efficient FPGA Implementation of Classic Audio Effects. *13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI'21),* July 1-3, Pitesti, Romania. 2021. P. 1–6. DOI: 10.1109/ECAI52376.2021.9515041
11. Esen Y., San İ. Low-latency SoC design with high-level accelerators specific to sound effects. *International Journal of Advances in Engineering and Pure Sciences*. 2021. Vol. 33. P. 78–87. DOI: 10.7240/jeps.897556
12. Zhang Y., Wang C., Lei G., Lu Y., Sun F., Xu C., Li X., Zhou X. An FPGA-based accelerated optimization algorithm for real-time applications. *Journal of Signal Processing Systems*. 2020. Vol. 92, No. 10. P. 1155–1176. DOI: 10.1109/ISPA/IUCC.2017.00098
13. Ustun E., San I., Yin J., Yu C., Zhang Z. IMpress: large integer multiplication expression rewriting for FPGA HLS. *30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'22)*, May 15–18, New York City, USA. 2022. P.1–10. DOI: 10.1109/FCCM53951.2022.9786123
14. Havinga T., Jiao X., Liu W., Moerman I. Accelerating FPGA-based WI-FI transceiver design and prototyping by high-level synthesis. *31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'23)*, May 8–11, Marina Del Rey, USA. 2023. P. 1–7. DOI: 10.1109/FCCM57271.2023.00047
15. Deulkar A. S., Kolhare N. R. FPGA implementation of audio and video processing based on Zedboard. *International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC'20)*, October 30–31, Aurangabad, India. 2020. Vol. 6. P. 143580–143591. DOI: 10.1109/ACCESS.2021.3120470
16. Shkil A., Rakhlis D., Filippenko I., Korniienko V. Design and self-diagnostics of cyberphysical control devices on SOC platform. *Innovative technologies and scientific solutions for industries*.Vol. 4(26). P. 122–134. DOI: 10.30837/ITSSI.2023.26.122
17. Neon. Arm Developer. URL: https://developer.arm.com/Architectures/Neon (дата звернення: 29.01.2024).
18. Signal processing stack exchange. URL: https://dsp.stackexchange.com/questions/66451/fir-filtering-operation-also-convolution (дата звернення: 04.02.2024).
19. Convolution theory. URL: https://personal.utdallas.edu/~raja1/EE%203302%20Fall%2016/GaTech/cconvdemo/help/theory.html#:~:text=Convolution%20is%20an%20operation%20by,and%20output%20y(t) (дата звернення: 04.02.2024).
20. Lineartime-invariant systems and convolution. URL: https://redwood.berkeley.edu/wpcontent/uploads/2018/08/lti_convolution.pdf (дата звернення: 04.02.2024).
21. Smith S. W. The scientist and engineer's guide to digital signal processing. 2nd ed. San Diego, Calif: California Technical Pub., 1999. 650 p. URL: https://ia801301.us.archive.org/23/items/GuideToDigitalSignalProcessing/Guide%20To%20Digital%20Signal%20Processing.pdf
22. Fingeroff M. High-Level Synthesis Blue Book. Bloomington: Xlibris Corporation, 2010. 286 p. URL: https://www.cse.usf.edu/~haozheng/teach/cda4253/doc/hls/hls_bluebook_uv.pdf (дата звернення: 04.02.2024).
23. Feedforward Comb Filters. URL: https://www.dsprelated.com/freebooks/pasp/Feedforward_Comb_Filters.html (дата звернення: 04.02.2024).

References

1. Lee, E. A., Seshia, S. A. (2017), "Introduction to embedded systems: A cyber-physical system approach, MIT Press", 564 p. available at: https://ptolemy.berkeley.edu/books/leeseshia/releases/LeeSeshia_DigitalV2_2.pdf (last accessed: 04.02.2024).
2. Popoff, M., Michon, R., Risset, T., Orlarey, Y., Letz, S. (2022), "Towards an FPGA-based compilation flow for ultra-low latency audio signal processing". *Proceedings of the 19th Sound and Music Computing (SMC-22),* June 5–12, Saint-Étienne, France, P. 555–562. DOI: 10.1109/ASAP57973.2023.00018
3. Wassi, G., Iloga, S., Romain, O., Granado, B., Tchuenté, M. (2018), "FPGA-based simultaneous multichannel audio processor for musical genre indexing applications in broadcast band", *Journal of parallel and distributed computing*, Vol. 119, P. 146–161. DOI: 10.1016/j.jpdc.2018.02.011
4. Wang, T., Bohanan, S. (2023), "Active noise cancellation with FPGA – practical considerations", *INTER-NOISE and NOISE-CON congress and conference proceedings (NOISE-CON23)*, May 15–18, Grand Rapids, USA, Vol. 266, no. 1, P. 1036–1043. URL: DOI: 10.3397/NC_2023_0124

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) *Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)*

5. Cannon, D., Fang, T., Saniie, J. (2022), "Modular Delay Audio Effect System on FPGA", *IEEE International Conference on Electro Information Technology (eIT)*, May 19–21, Mankato, USA, P. 248–251. DOI: 10.1109/eIT53891.2022.9813875
6. Xie, W., Yang, F. (2022), "Design and implementation of audio stream processing based on ZYNQ", *IEEE 6th information technology and mechatronics engineering conference (ITOEC)*, March 4–6, Chongqing, China, P. 589–592. DOI: 10.1109/ITOEC53115.2022.9734347
7. Popoff, M., Michon, R., Risset, T., Cochard, P., Letz, S., Orlarey, Y., Dinechim, de F. (2023), "Audio DSP to FPGA Compilation: The Syfala Toolchain Approach", Research report №9507, Grame, Emeraude: Inria,18 p. available at: https://hal-lara.archives-ouvertes.fr/hal-04099135/ (last accessed: 04.02.2024).
8. Kortli Y., Gabsi S., Jridi M., Alfalou A., Atri M. (2021), "Hw/Sw Co-Design technique for 2D fast fourier transform algorithm on Zynq SoC", *Integration,* Vol. 82, P. 78–88. DOI: 10.1016/j.vlsi.2021.09.005
9. Azzaz, M., Maali, A., Kaibou, R., Kakouche, I., Mohamed S., Hamil H. (2020), "FPGA HW/SW codesign approach for real-time image processing using HLS", *1st International Conference on Communications, Control Systems and Signal Processing (CCSSP'20),* May 16–17, EL OUED, Algeria, P. 169–174. DOI: 10.1109/CCSSP49278.2020.9151686
10. Dragoi, C., Anghel, C, Stanciu, C., Paleologu, C. (2021), "Efficient FPGA Implementation of Classic Audio Effects", *13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI'21),* July 1–3, Pitesti, Romania, P. 1–6. DOI: 10.1109/ECAI52376.2021.9515041
11. Esen, Y., San, İ. (2021), "Low-latency SoC design with high-level accelerators specific to sound effects", *International Journal of Advances in Engineering and Pure Sciences*, Vol. 33, P. 78–87. DOI: 10.7240/jeps.897556
12. Zhang, Y., Wang, C., Lei, G., Lu, Y., Sun, F., Xu, C., Li, X., Zhou, X. (2020). "An FPGA-based accelerated optimization algorithm for real-time applications", *Journal of Signal Processing Systems*, Vol. 92, no. 10, P. 1155–1176. DOI: 10.1109/ISPA/IUCC.2017.00098
13. Ustun, E., San, I., Yin, J., Yu, C., Zhang, Z. (2022), "IMpress: large integer multiplication expression rewriting for FPGA HLS", *30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'22)*, May 15–18, New York City, USA, P. 1–10. DOI: 10.1109/FCCM53951.2022.9786123
14. Havinga, T., Jiao, X., Liu, W., Moerman, I. (2023), "Accelerating FPGA-based WI-FI transceiver design and prototyping by high-level synthesis", *31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'23)*, May 8–11, Marina Del Rey, USA, P. 1–7. DOI: 10.1109/FCCM57271.2023.00047
15. Deulkar, A. S., Kolhare, N. R. (2020), "FPGA implementation of audio and video processing based on Zedboard", *International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC'20)*, October 30–31, Aurangabad, India, Vol. 6, P. 143580–143591. DOI: 10.1109/ACCESS.2021.3120470
16. Shkil, A., Rakhlis, D., Filippenko, I., Korniienko, V. (2023), "Design and self-diagnostics of cyberphysical control devices on SOC platform", *Innovative technologies and scientific solutions for industries*, Vol. 4(26), P. 122–134. DOI: 10.30837/ITSSI.2023.26.122
17. "Neon", Arm Developer, available at: https://developer.arm.com/Architectures/Neon (last accessed: 29.01.2024).
18. "Signal Processing Stack Exchange", available at: https://dsp.stackexchange.com/questions/66451/fir-filtering-operation-also-convolution (last accessed: 04.02.2024).
19. "Convolution Theory", available at: https://personal.utdallas.edu/~raja1/EE%203302%20Fall%2016/GaTech/cconvdemo/help/theory.html#:~:text=Convolution%20is%20an%20operation%20by,and%20output%20y(t) (last accessed: 04.02.2024).
20. "Linear time-invariant systems and convolution", available at: https://redwood.berkeley.edu/wp-content/uploads/2018/08/lti_convolution.pdf (last accessed: 04.02.2024).
21. Smith, S. W. (1999), "The scientist and engineer's guide to digital signal processing", 2nd ed., San Diego, Calif: California Tech. Pub., 650 p. available at: https://ia801301.us.archive.org/23/items/GuideToDigitalSignalProcessing/Guide%20To%20Digital%20Signal%20Processing.pdf (last accessed: 04.02.2024).
22. Fingeroff, M. (2010), "High-Level Synthesis Blue Book", Bloomington: Xlibris Corporation, 286 p. available at: https://www.cse.usf.edu/~haozheng/teach/cda4253/doc/hls/hls_bluebook_uv.pdf (last accessed: 04.02.2024).
23. "Feedforward Comb Filters", available at: https://www.dsprelated.com/freebooks/pasp/Feedforward_Comb_Filters.html (last accessed: 04.02.2024).

Надійшла 01.03.2024

Відомості про авторів / About the Authors

Шкіль Олександр Сергійович – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри автоматизації проєктування обчислювальної техніки, Харків, Україна; e-mail: oleksandr.shkil@nure.ua; ORCID ID: https://orcid.org/0000-0003-1071-3445

Рахліс Дарія Юхимівна – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри автоматизації проєктування обчислювальної техніки, Харків, Україна; e-mail: dariia.rakhlis@nure.ua; ORCID ID: https://orcid.org/0000-0002-6652-1840

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) *ISSN 2524-2296 (online)*

Філіпенко Інна Вікторівна – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри автоматизації проєктування обчислювальної техніки, Харків, Україна; e-mail: inna.filippenko@nure.ua; ORCID ID: https://orcid.org/0000-0002-3584-2107

Корнієнко Валентин Русланович – Харківський національний університет радіоелектроніки, аспірант кафедри автоматизації проєктування обчислювальної техніки, Харків, Україна; e-mail: valentyn.korniienko1@nure.ua; ORCID ID: https://orcid.org/0000-0001-7070-5127

Рожнова Тетяна Григорівна – кандидат технічних наук, Харківський національний університет радіоелектроніки, доцент кафедри автоматизації проєктування обчислювальної техніки, Харків, Україна; e-mail: tetiana.rozhnova@nure.ua; ORCID ID: https://orcid.org/0000-0002-4484-8674

Shkil Alexander − PhD (Engineering Sciences), Associate Professor, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Design Automation, Kharkiv, Ukraine.

Rakhlis Dariia − PhD (Engineering Sciences), Associate Professor, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Design Automation, Kharkiv, Ukraine.

Filippenko Inna − PhD (Engineering Sciences), Associate Professor, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Design Automation, Kharkiv, Ukraine.

Korniienko Valentyn – Kharkiv National University of Radio Electronics, PhD student at the Department of Design Automation, Kharkiv, Ukraine.

Rozhnova Tetiana − PhD (Engineering Sciences), Kharkiv National University of Radio Electronics, Associate Professor at the Department of Design Automation, Kharkiv, Ukraine.

AUTOMATED DESIGN OF EMBEDDED DIGITAL SIGNAL PROCESSING SYSTEMS ON *SoC* PLATFORM

The object of the study is the procedures for automated design and analysis of digital signal processing algorithms on the *SoC* technology platform. **The subject of the study** is models, methods and procedures for designing and optimal selection of *SoC* components for the implementation of digital signal processing algorithms for audio spectrum. **The aim of the study** is to develop models and procedures for determining the possibilities of a compromise distribution of signal processing algorithm computations in the cycle of computer-aided design on the *SoC* technology platform in terms of performance and the feasibility of using hardware and software algorithms realization. The article solves the following **tasks**: consideration of the procedures for interacting the processor core with programmable logic as part of system-on-chip systems; development of procedures for computer-aided design and analysis of signal processing systems using programming languages and hardware description languages for the implementation of embedded systems. The following **methods** are being used: implementation of digital signal processing algorithms in the C programming language and high-level synthesis tools for realizing IP blocks, diagnostic experiment by generating test signal patterns, and analysis of the processing results at the system output. **The results achieved.** Based on the analysis of the procedures for the interaction of the processor core and programmable logic on the selected *SoC* platform, a model of the audio spectrum signal processing system is designed. The practical implementation was performed based on the *Vivado/Vitis/Vitis HLS* CAD tool stack. The proposed model was verified using a programmable test signal generator and analyzing the obtained characteristics of digital filters at the system output. **Conclusions.** The article analyzes the principles of designing embedded information processing systems implemented in system-on-chip. The principles of building and analyzing digital signal processing systems based on system-on-chip containing programmable logic and processor parts are considered. The developed methods have been tested on the algorithms of *CIC* and *FIR* filters on the technological platform of *SoC FPGA* of the *ZYNQ-7000* family of *Xilinx* company.

Keywords: embedded systems; systems-on-chip; *FPGA*; *C* programming language; digital signal processing algorithms; audio signals; digital filters.

Бібліографічні описи / Bibliographic descriptions

Шкіль О. С., Рахліс Д. Ю., Філіпенко І. В., Корнієнко В. Р., Рожнова Т. Г. Автоматизоване проєктування вбудованих систем цифрового оброблення сигналів на платформі *SoC*. *Сучасний стан наукових досліджень та технологій в промисловості*. 2024. № 1 (27). С. 192–203. DOI: https://doi.org/10.30837/ITSSI.2024.27.192

Shkil, A., Rakhlis, D., Filippenko, I., Korniienko, V., Rozhnova, T. (2024), "Automated design of embedded digital signal processing systems on SOC platform", *Innovative Technologies and Scientific Solutions for Industries*, No. 1 (27), P. 192–203. DOI: https://doi.org/10.30837/ITSSI.2024.27.192