https://doi.org/10.31891/2219-9365-2025-82-56
УДК 621.3.049.77

МАЙКІВ Ігор
Західноукраїнський національний університет
https://orcid.org/0009-0000-8064-4858
e-mail: imaykiv@yahoo.com
КВАСОВСЬКИЙ Олександр
Західноукраїнський національний університет
https://orcid.org/0000-0001-9856-3917
e-mail: o.kvasovskyi@gmail.com

ПОДІЛЬНИК ЧАСТОТИ ІЗ ЗМІННИМ КОЕФІЦІЄНТОМ ПОДІЛУ, ДЛЯ ЦИФРОВИХ СИСТЕМ НА БАЗІ ПЛІС 7-СЕРІЇ ФІРМИ XILINX

В роботі представлено метод реалізації подільників частоти зі змінним коефіцієнтом поділу для систем на базі програмованих логічних інтегральних схем (ПЛІС) 7-ї серії фірми XILINX. В запропонованому методі подільники реалізують на базі апаратних таблиць перетворення (Look-Up Table), які функціонують в режимі регістра зсуву, без застосування традиційних двійкових лічильників реалізованих на D-тригерах.

Ключові слова: подільник частоти, регістр зсуву, програмована логічна інтегральна схема.

MAIKIV Ihor, KVASOVSKYI Oleksandr
West Ukrainian National University

FREQUENCY DIVIDER WITH VARIABLE DIVISION RATIO FOR DIGITAL SYSTEMS BASED ON FPGA 7-TH SERIES OF XILINX

Complex computing devices with reconfigurable architectures are widely used in real-time systems where it is necessary to change the operation algorithm. Field Programmable Gate Arrays (FPGAs) are base components for such devices. FPGAs enable to design of computing devices wherein the hardware structure is configured to implement the operation algorithm in the most efficient manner. This approach allows for more effective technical solutions compared to those based on microprocessors and microcontrollers.

When developing such devices on FPGAs, it is necessary to implement a set of synchronization and control signals with frequencies in hundreds or even thousands of times lower than the main clock frequency. Modern FPGAs include a set of Mixed-Mode Clock Manager (MMCM) hardware modules to generate the required set of frequencies. However, the number of MMCMs is limited, and their signals are typically global within the project. Therefore, using MMCMs is not justified when it is necessary to implement a local set of synchronization and control signals within a separate module of a complex project.

The article presents a method for implementing digital counters/dividers with a variable division ratio for systems based on the 7th series of XILINX FPGAs. The method assumes that dividers are implemented using multifunctional Look-Up Tables (LUTs) operating in shift register mode, without employing traditional binary counters/dividers implemented with D flip-flops.

Using a shift register with connected input and output is one of the most effective ways to implement dividers. The length of the register determines the division ratio (N) of such a divider. This solution allows the output signal frequency to be reduced by a factor of N compared to the main clock frequency, while the pulse duration remains the same as the clock signal period.

The divider operates in two modes: 1) division mode; 2) initialization mode. In division mode, the output provides a signal divided by N. The divider switches to initialization mode after a hardware reset or when the user decides to change the division ratio. Unlike well-known solutions, the proposed technical design uses a small Finite State Machine (FSM) to control divider operation in initialization mode. The application of an FSM allows the implementation of a divider with a variable division ratio without the need to update the design.

The obtained results show that the divider requires fewer FPGA hardware resources for implementation compared to traditional binary counters built with D flip-flops. The proposed divider can be used as a base component in the design of programmable dividers/timers. Connecting several dividers in series allows the total division ratio factor to be increased by multiplying the division factors of each divider, achieving a total division factor of 10^3–10^6.

The design of a custom IP-core of a programmable divider/timer supporting of the AXI system bus, will be the next step of researching.

Keywords: frequency divider, shift register, field programmable gate array.

Стаття надійшла до редакції / Received 07.04.2025
Прийнята до друку / Accepted 02.05.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Обчислювальні пристрої із реконфігурованою апаратною архітектурою [1-4] широко використовуються в системах реального часу, де необхідно часто змінювати алгоритм функціонування системи. Прикладами таких систем є радіолокаційні станції, системи адаптивної фільтрації та цифрового опрацювання сигналів, системи зв'язку. Елементною базою при створенні таких пристроїв є програмовані логічні інтегральні мікросхеми (ПЛІС) [5, 6], що дозволяє створювати обчислювальні та керуючі пристрої,

апаратна структура яких як найкраще налаштована на реалізацію необхідного алгоритму. В цілому, це дозволяє отримати більш ефективні технічні рішення (ТР), порівняно із рішеннями на базі мікропроцесорів та мікроконтролерів.

В процесі створення таких пристроїв, необхідно реалізувати набір сигналів синхронізації та керування, частота яких може на декілька порядків бути нижчою від основної частоти синхронізації, на якій працює пристрій. Зазвичай, для формування необхідного набору сигналів синхронізації в ПЛІС використовують апаратні компоненти Mixed-Mode Clock Manager (MMCM) [7, 8], число яких обмежене, а самі сигнали зазвичай є глобальними в межах проекту. Тому, застосування PLL та MMCM не завжди оправдане, коли необхідно реалізувати локальний набір синхронних керуючих сигналів в середині окремого модуля, або для забезпечення взаємодії між окремими модулями, що вимагає створення окремих подільників для формування таких сигналів.

АНАЛІЗ ТРАДИЦІЙНИХ МЕТОДІВ РЕАЛІЗАЦІЇ ДІЛЬНИКІВ НА ПЛІС

Проектуючи компоненти цифрових пристроїв на одній із мов високого рівня, що використовуються для опису апаратних засобів (VHDL, Verilog, SystemVerilog), розробник створює поведінкову RTL-модель [5, 6], яка описує алгоритм роботи модуля на абстрактному рівні, та не пов'язана з апаратними ресурсами ПЛІС конкретного виробника. На наступних етапах виконується аналіз RTL-моделі та синтез, під час якого відбувається трансляція HDL-програми в набір базових апаратних компонентів ПЛІС, необхідних для його реалізації, виходячи із вибраного типу ПЛІС. Такий підхід дозволяє створювати бібліотеку параметризованих модулів які не прив'язані до конкретної архітектури ПЛІС, та можуть багаторазово використовуватись у різних проектах. Однак, отримані результати синтезу не завжди виявляються оптимальними. Як приклад, на рис. 1 наведено еквівалентну схему 10-розрядного лічильника, отриману в програмному пакеті Vivado 2020.2 фірми Xilinx, за результатами аналізу RTL-моделі, написаної на мові SystemVerilog.

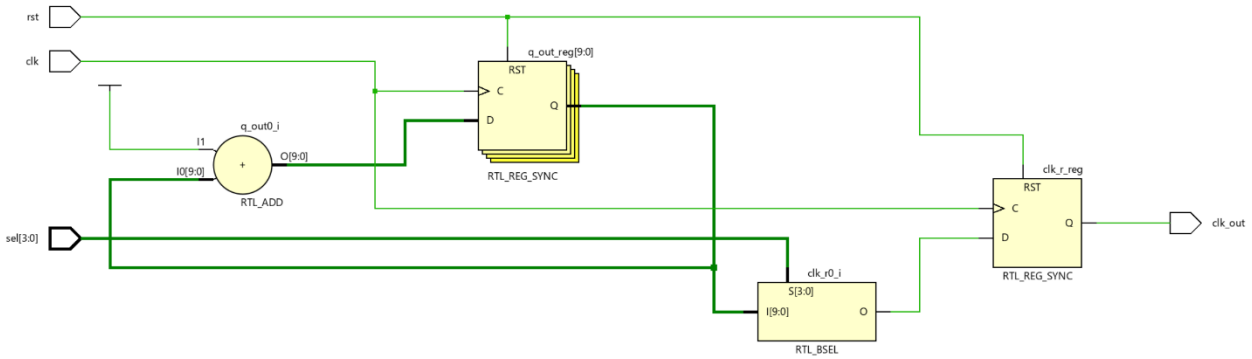

Рис. 1. Еквівалентна схема 10-розрядного лічильника в програмному пакеті Vivado 2020.2.

Можна побачити, що лічильник містить 10-розрядний паралельний регістр (RTL_REG_SYNC), суматор (RTL_ADD) та мультиплексор (RTL_BSEL), а також тригер D-тригер, який забезпечує регістровий вихід модуля. Для його реалізації необхідно 10 LUT які реалізують комбінаційні схеми (КС) та 11 D-тригерів.

Наведений приклад наочно показує що традиційні підходи до реалізації подільників мало ефективні, коли необхідно забезпечити коефіцієнт ділення більше ніж 512-1024 (9-10 розрядів), оскільки це вимагає значних апаратних затрат на їх реалізацію. Також, це може призвести до зменшення максимальної робочої частоти лічильника, через збільшення часу проходження сигналу між окремими розрядами лічильника, оскільки D-тригери можуть бути розташовані у різних макрокомірках (cells) ПЛІС.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є: пошук нових підходів та рекомендацій до реалізації подільників, виходячи із особливостей архітектури ПЛІС, який дозволить: 1) зменшити апаратні затрати на їх реалізацію; 2) збільшити максимальну робочу частоту; 3) змінювати коефіцієнт ділення без необхідності вносити зміни у проект; 4) формувати широку сітку частот як із парним, так і не парним коефіцієнтом поділу.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Базовим компонентом ПЛІС 7-ї серії фірми Xilinx є макрокомірка (macrocell), яка містить вісім 5-входових апаратних таблиць перетворення (Look-Up Table - LUT), вісім тригерів, а також додаткові КС [9, 10]. Всі макрокомірки поділені на два типи: 1) логічні макрокомірки (logic-cell); 2) макрокомірки із пам'яттю (memory-cell). Їх принциповою відмінністю є тип LUTів. У логічних макрокомірок, LUTи дозволяють реалізувати лише КС до 6-змінних. Разом з тим, макрокомірки із пам'яттю LUTи є багатофункціональними компонентами, на базі яких можна реалізувати:

1) комбінаційну схему 1-6 – змінних (LUT1-LUT6);

2) регістр зсуву із послідовним входом та послідовним виходом, змінної довжини та максимальною довжиною у 32 розряди (SRLC32E);

3) 1-розрядний постійний або оперативний запам'ятовуючи пристрій в 32 адреси (RAM32X1S, ROM32X1).

Кожен із вище вказаних цифрових елементів можна реалізувати на базі LUTa, оголосивши в проекті один із доступних макросів [11], які безпосередньо вказують програмі-синтезатору як саме повинен бути налаштований LUT, та яку функцію він буде виконувати. Це дозволяє розробнику безпосередньо у проекті оголошувати необхідну кількість регістрів зсуву для формування ліній затримки, блоків пам'яті малого об'єму.

Один із найбільш ефективних способів реалізації подільників є використання регістра зсуву у якого з'єднані вхід та вихід [12]. Коефіцієнт поділу (N) такого подільника визначає довжина регістра. На відміну від традиційних лічильників/подільників в яких на кожному із виходів формується меандр, таке рішення дозволяє отримати на виході сигнал, частота якого в N менша від частоти синхронізації, але тривалість імпульсу залишається такою ж як в сигналу синхронізації.

Оголосивши в проекті регістр зсуву на базі LUT, розробник отримує 32-розрядний регістр зсуву, схема якого наведена на рис. 2 [11].

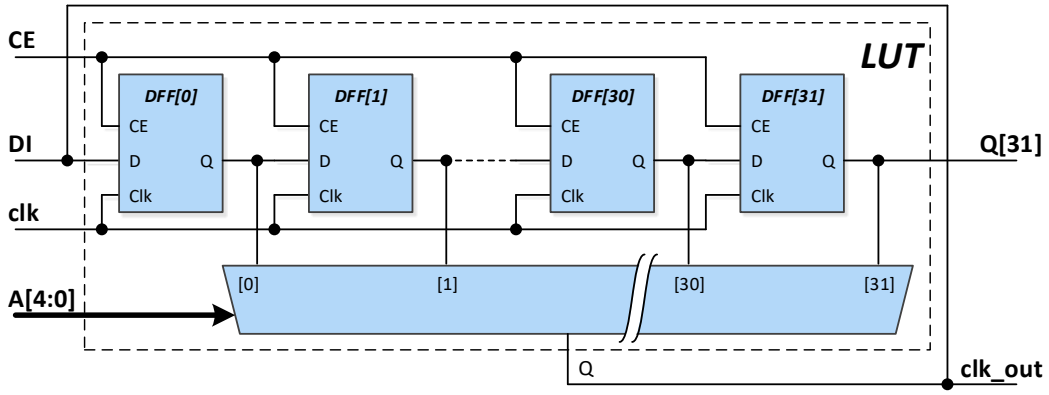

Рис. 2. LUT в режимі регістра зсуву

Регістр має два виходи:

1) Q[31] – безпосередньо під'єднаний до найстаршого розряду;

2) Q – під'єднаний до одного із розрядів, номер якого заданий двійковим числом на шині адрес A[4:0].

Таким чином, змінюючи значення на шині адрес A[4:0] можна змінити довжину регістра зсуву, а отже і коефіцієнт поділу. Вхід CE (Clock Enable) дозволяє/зупиняє роботу регістра.

Однак, представлене рішення має ряд принципових недоліків, безпосередньо пов'язаних із архітектурою LUT, а тому не може безпосередньо використовуватись для реалізації подільника із змінним коефіцієнтом ділення. Головними із них є:

1) в регістра відсутній вхід скидання (reset), тому він не очищується при скиданні.

2) змінюючи в процесі роботи «на льоту» довжину регістра (коефіцієнт поділу) може виявитись що регістр не містить лог.1 в жодному із розрядів, коли зменшили його довжину, або регістр може містити декілька лог.1, коли збільшили його довжину.

Тому, після апаратного скидання або зміни довжини регістра (коефіцієнта поділу) необхідно виконати процедуру ініціалізації, яка приведе його у вихідний стан, та включає дві операції:

1) очищення вмісту всіх старших розрядів.

2) завантаження лог.1 у молодший розряд регістра (Least Significant Bit – LSB).

Для того, щоб реалізувати процедуру ініціалізації необхідно декілька додаткових вузлів які виконають вище вказані операції, а також керують процесом ініціалізації.

Схема подільника зі змінним коефіцієнтом ділення на базі LUT представлена на рис. 3.

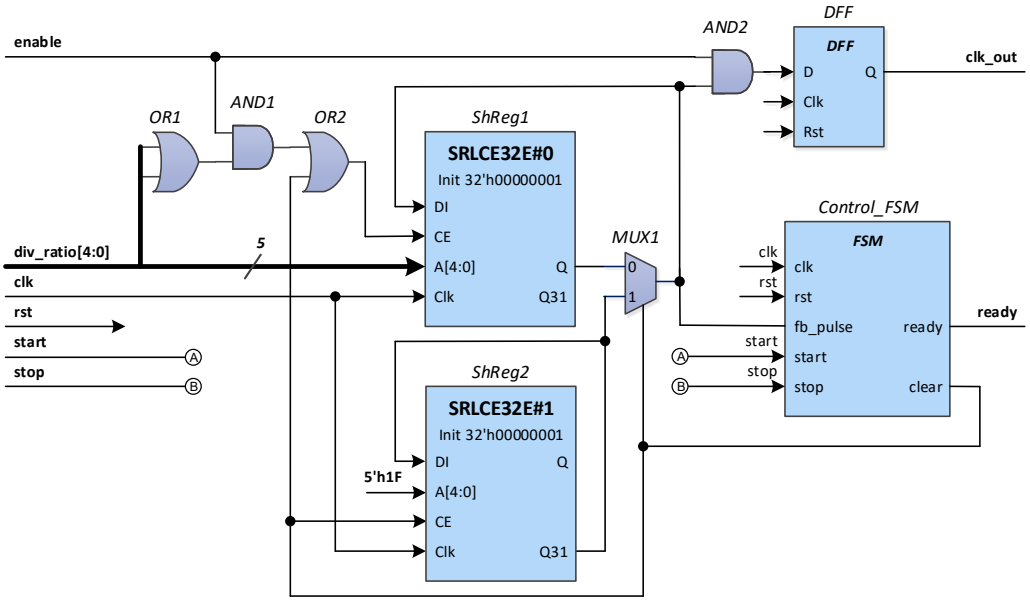

Рис. 3. Подільник із змінним коефіцієнтом ділення на базі LUT

Подільник включає наступні компоненти:

1) регістр зсуву ShReg1 на базі LUT - виконує функцію дільника із змінним коефіцієнтом поділу;
2) регістр зсуву ShReg2 на базі LUT - використовується на етапі ініціалізації, та містить ініціалізуючу константу int32 h00000001;
3) керуючий цифровий автомат (ЦА) Control_FSM - керує роботою подільника в режимі ініціалізації;
4) мультиплексор MUX1 – включений в коло зворотного зв'язку регістр зсуву ShReg1;
5) D-тригер DFF – забезпечує регістровий вихід подільника;
6) Логічні елементи (ЛЕ) OR1, OR2, AND1, AND2 – функціональне призначення яких буде розглянуте в тексті.

Також у дільника присутній набір входів та виходів, функціональне призначення яких наведено нижче:

1) clk – вхід синхронізації;
2) rst – вхід системного скидання;
3) div_ratio[4:0] – 5-розрядна вхідна шина, яка задає коефіцієнт поділу;
4) enable – вхід керування який дозволяє/забороняє роботу дільника, без зміни коефіцієнта поділу;
5) start - вхід на який надходить керуючий сигнал що інформує Control_FSM про початок роботи системи;
6) stop – вхід на який надходить керуючий сигнал що інформує Control_FSM про завершення роботи системи та початок процедури ініціалізації подільника;
7) ready – вихід, сигнал на якому інформує інші вузли системи про те що дільник завершив процедуру ініціалізації та готовий до роботи;
8) clk_out – вихід, на який поступає поділений в N-раз сигнал.

Коефіцієнт поділу дільника (N) обчислюється за наступним виразом:

$$N = 1 + div\_ratio[4:0], \tag{1}$$

де, $div\_ratio[4:0]$ - число у двійковому форматі в діапазоні 0-31.

Із виразу (1) слідує, що подільник забезпечує коефіцієнт ділення на будь-яке число в діапазоні 2-32. Окремий випадок, коли $div\_ratio[4:0] = 0$, то на виході подільника присутня лог. 1.

Подільник працює в двох режимах: 1) ділення (division mode); 2) ініціалізації (initialization mode). Розглянемо роботу дільника в кожному із вище вказаних режимів.

Після того як у ПЛІС завантажено файл конфігурації, подільник переходить у режим ділення (division mode). Регістр зсуву ShReg1 містить константу int32 h00000001. Двійкове число на шині div_ratio[4:0] поступає на порт A[4:0] регістра ShReg1, та задає коефіцієнт поділу. Одночасно, це число поступає на 5-входовий ЛЕ OR1, який визначає чи двійкове число div_ratio[5:0] більше нуля.

Вихід ЛЕ OR1 з'єднаний з одним із входів ЛЕ AND1. На другий вхід ЛЕ AND1 надходить сигнал enable. Сигнал із виходу ЛЕ AND1, через ЛЕ OR2 поступає на вхід CE регістр ShReg1. Якщо на вході enable присутній лог.0 або значення div_ratio[4:0] рівне нулю, то на вході CE регістра ShReg1 присутній лог.0 і дільник зупиняється. Якщо на вході enable присутня лог.1 а div_ratio[4:0] більше нуля, то на вхід CE регістра ShReg1 поступає лог.1, і дільник розпочинає (продовжує) роботу, та через N тактів синхросигналу, лог. 1 з'явиться на Q виході регістра ShReg1. Сигнал із виходу Q через мультиплексор MUX1 надходить на вхід DI та знову поступає в регістр ShReg1. Одночасно цей сигнал надходить на один із входів ЛЕ AND2 та через D-тригер (DFF) надходить до інших вузлів системи.

В режим ініціалізації подільник переходить після апаратного скидання (reset), або коли користувач вирішив змінити коефіцієнт поділу. На цьому етапі, робота дільника зупиняється, та виконується процедура ініціалізації, яка триває 32 такти, під час якої в регістр ShReg1 завантажують константу int32 h00000001. Для формування такої послідовності використовують регістр зсуву ShReg2, який налаштований як 32-розрядний регістр зсуву, що містить константу int32 h00000001. Його вихід Q31 безпосередньо з'єднаний із входом DI, а також із другим входом мультиплексора MUX1.

Оскільки, робота подільника зупинена то процесом ініціалізації керує ЦА Control_FSM. ЦА працює в одному клоковому домені із іншими компонентами та синхронізується сигналом clk. Окрім раніше описаних сигналів, у ЦА також наявні інші входи та виходи: 1)fb_pulse – вхід на який поступає сигнал що сигналізує про завершення процедури очищення ShReg1; 2) clear – вихід, сигнал з якого включає процедуру ініціалізації та керує мультиплексом MUX1. Алгоритм, роботи керуючого ЦА представлено у вигляді графа переходів (ГП) на рис. 4. ЦА допускає чотири можливі стани: 1) зсуву (SHIFT); 2) завершення очищення (CLEAR_DONE); 3) очікування (IDLE); 4) ділення (DIVISION_MODE). Кожному зі станів відповідає своя вершина в якій зазначено назва стану, а також набір вихідних сигналів та значення, які вони приймають у кожному із цих станів. Перехід з одного стану у наступний відбувається якщо виконується умова переходу.

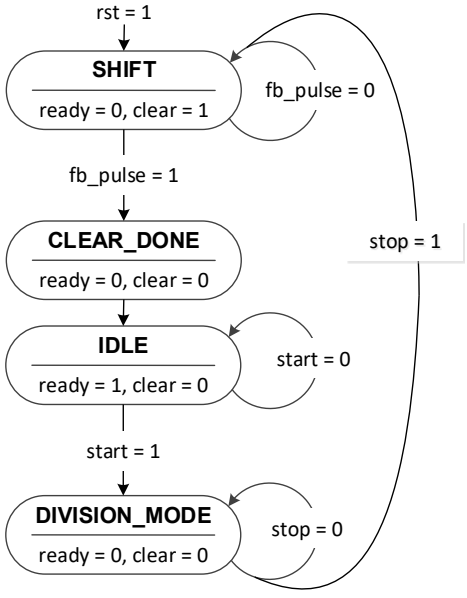

Рис. 4. Граф-переходів керуючого ЦА

Як можна побачити, процедура ініціалізації подільника охоплює три стани: зсуву (SHIFT), завершення очищення (CLEAR_DONE) та очікування (IDLE), в які послідовно переходить ЦА.

Після апаратного скидання, ЦА безумовно переходить у стан SHIFT. На виході clear з'являється лог.1. що поступає на керуючий вхід мультиплексора MUX1, який з'єднує вихід Q31 регістр зсуву ShReg2 із входом DI регістр зсуву ShReg1. Одночасно сигнал clear надходить на входи CE регістрів зсуву ShReg1, ShReg2 та дозволяє їх роботу. У цьому стані ЦА перебуває 32 такти, поки на виході Q31 регістра ShReg2 не з'явиться лог.1, яка поступає на вхід DI регістра зсуву ShReg1 а також на вхід fb_pulse ЦА. В результаті, ЦА переходить у стан CLEAR_DONE, в якому перебуває лише один такт сигналу синхронізації, під час якого лог.1 записується у молодші розряди регістрів ShReg1 та ShReg2. Далі ЦА переходить у стан IDLE, який завершує процес ініціалізації. У цьому стані, вихід ready переходить у лог.1, інформуючи інші вузли пристрою про те що дільник завершив процедуру ініціалізації та готовий до роботи. В стані IDLE подільник перебуває поки на вході start не з'явиться лог.1, після чого ЦА переходить у стан DIVISION_MODE, і подільник розпочинає роботу в режимі ділення. При цьому, вихід ready повертається у лог.0. В стані DIVISION_MODE подільник перебуває поки на вході stopt не з'явиться лог.1, після чого ЦА переходить у стан SHIFT і розпочнеться процедура ініціалізації.

РЕЗУЛЬТАТИ СИНТЕЗУ ТА МОДЕЛЮВАННЯ

Щоб оцінити ефективність запропонованого рішення, на мові SystemVerilog реалізовано HDL-проект подільника, із використанням програмного пакета фірми XILINX Vivado 2020.2. Апаратне тестування виконано на платі Nexys A7, фірми Digilent [13], яка реалізована на ПЛІС фірми XILINX XC7A100T-1CSG324C [14].

Узагальнена структурна схема подільника, отримана після завершення етапу синтезу, наведена на рис. 5, та включає два окремі модулі: 1) control_fsm – керуючий ЦА Control_FSM; 2) shift_reg – виконує ділення сигналу та ініціалізацію регістра після скидання, або зміни коефіцієнта поділу.

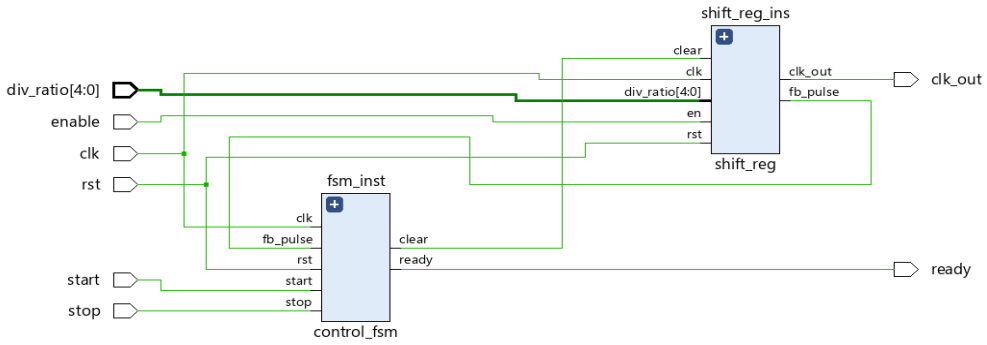

Рис. 5. Узагальнена стриктурна схема подільника після завершення етапі синтезу

На рис. 6 наведена еквівалентна схема модуля shift_reg, що отримана після завершення етапу синтезу, та повністю відповідає схемі наведеній на рис. 3.

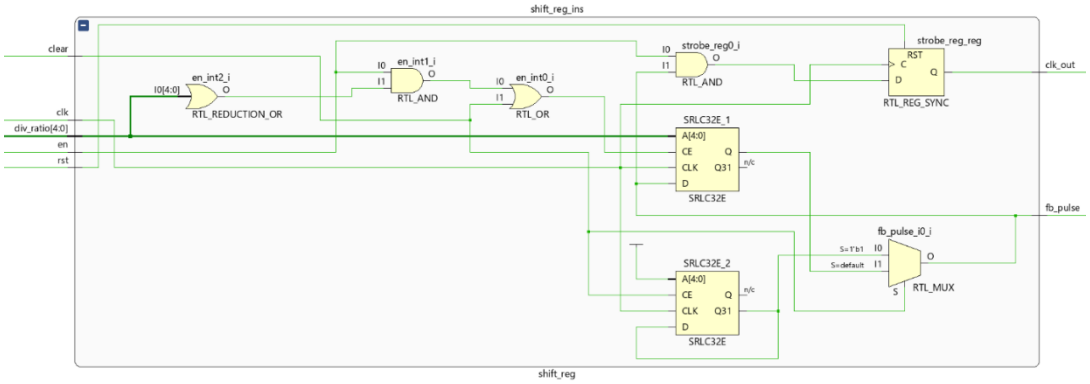

Рис. 6. Еквівалентна схема модуля shift_reg

На рис. 7 наведена еквівалентна схема модуля control_fsm, синтезована програмним пакетом Vivado 2020.2. Керуючий ЦА реалізовано як автомат Мура із двома регістровими виходами clear та ready. Два тригери state_reg[1:0] формують регістр стану ЦА, який містить інформацію про його поточний стан. Використання регістрових виходів дозволяє уникнути появи «глітчів» які можуть виникати на виході КС, при переході ЦА із одного стану в інший.

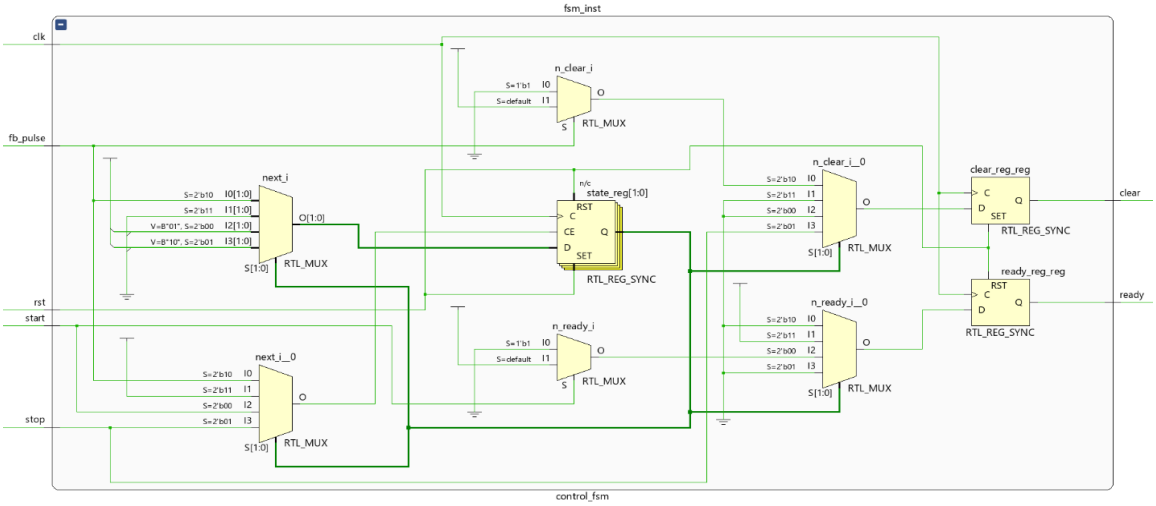

Рис. 7. Еквівалентна схема модуля control_fsm

Нижче, на рис. 8 наведено часові діаграми отримані при моделюванні роботи подільника в режимі ділення (DIVISION_MODE), із коефіцієнтом поділу 1:10 при частоті сигналу синхронізації clk 100 МГц.

Як можна побачити на наведених часових діаграмах, після завершення процедури ініціалізації ЦА перебуває в стані IDLE. Сигналу start, переводить ЦА в стан DIVISION_MODE. Одночасно вихід ready переходить у лог.0, інформуючи інші вузли системи що подільник розпочав роботу. Коли на вхід enable поступає лог.1 то через 100 нс. на виході clk_out з'являється сигнал із частотою 10 МГц. При цьому, тривалість імпульсу сигналу clk_out становить 10 нс, що відповідає тривалості періоду сигналу синхронізації clk.

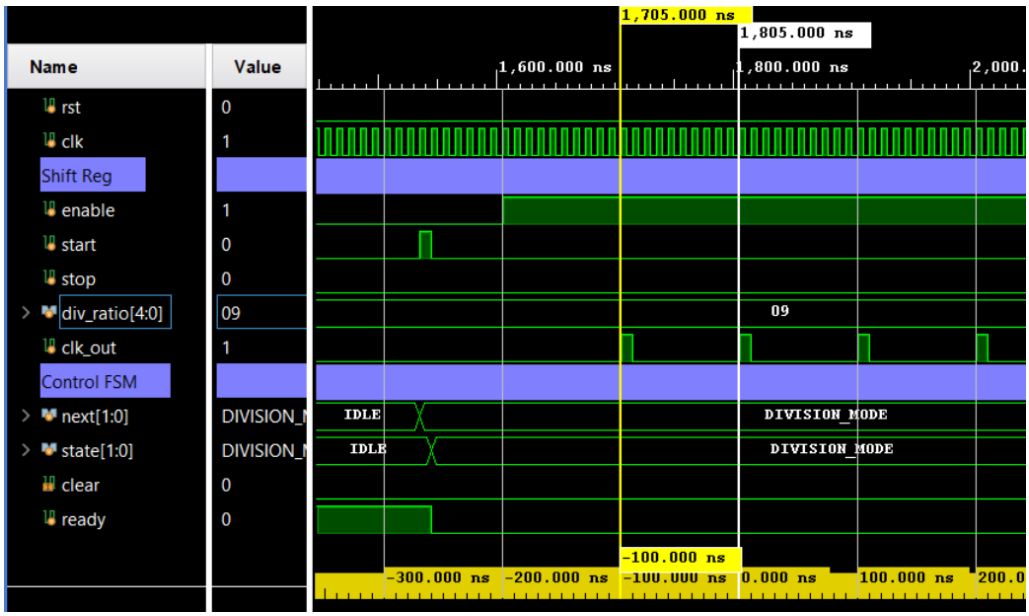

Рис. 8. Роботи подільника в режимі DIVISION_MODE (коефіцієнт поділу 1:10)

На рис. 9 наведено часові діаграми отримані при моделюванні процедури ініціалізації подільника. Як можна побачити із нижче наведених часових діаграм, ЦА Control_FSM перебуває в стані DIVISION_MODE. Перед тим як розпочати процедуру ініціалізації, користувач зупиняє подільник, встановивши на вході enable лог.0. Вся наступна процедура ініціалізації відбувається по алгоритму наведеному на рис. 4, та триває 320 нс (32 такти сигналу синхронізації).

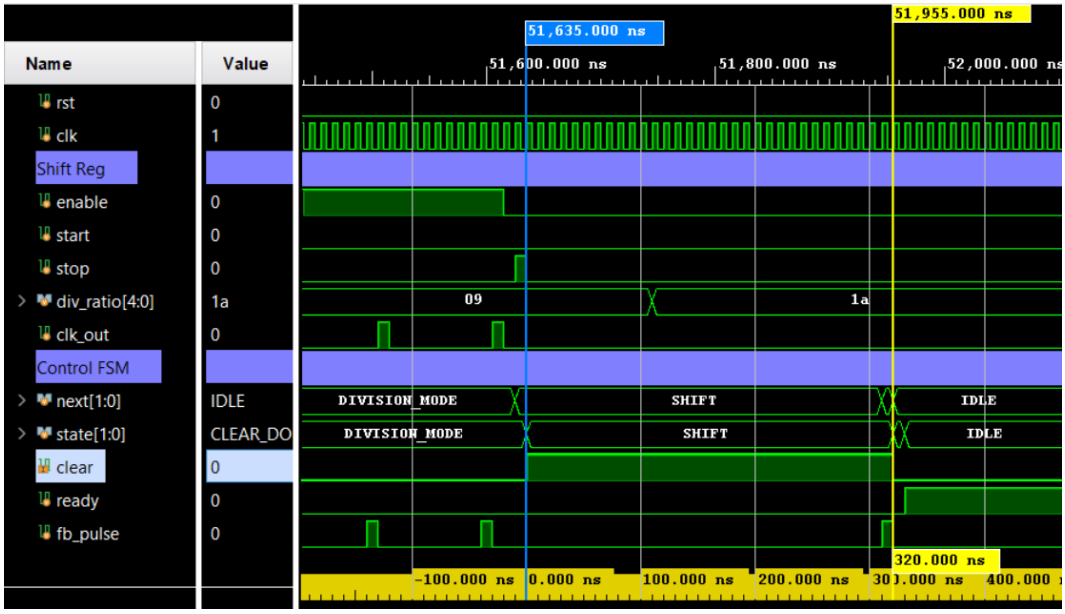

Рис. 9. Моделювання процедури ініціалізації подільника

На рис. 10 представлені дані про апаратні ресурси необхідні для реалізації подільника із змінним коефіцієнтом поділу N=2-32, звідки слідує що для його реалізації необхідно 8 LUTів та 5 тригерів.

Name	Slice LUTs (63400)	Slice Registers (126800)
divider	8	5
fsm_inst (control_fsm)	4	4
shift_reg_ins (shift_reg)	4	1

Рис. 10. Апаратні ресурси необхідні для реалізації подільника

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

За результатами проведеного дослідження, знайшов подальший розвиток метод реалізації подільників на базі регістрів зсуву в цифрових системах на базі ПЛІС. На відміну від відомих рішень, застосування керуючого ЦА дозволило реалізувати подільник із змінним коефіцієнтом поділу без необхідності оновлення проекту, який підтримує як «парні» (2^n) та «не парні» (2^{n+1}) коефіцієнти поділу в діапазоні N=2-32. Перевагою запропонованого підходу є те, що подільники реалізуються на базі багатофункціональних апаратних LUTів, які працюють в режимі регістра зсуву.

Для оцінки ефективності методу, на мові SystemVerilog створено HDL-проект подільника, проведено моделювання та експериментальні дослідження. Отримані результати показали що подільник потребує менше апаратних затрат на реалізацію, порівняно із традиційними методами побудови двійкових лічильників на D-тригерах. Запропонований подільник можна використати як базовий компонент при створенні програмованих подільників/таймерів. Їх послідовне включення дозволяє збільшити загальний коефіцієнт поділу, шляхом перемноження коефіцієнтів поділу кожного подільника, та забезпечити загальний коефіцієнт ділення $2\text{-}10^6$.

ПЛІС широко використовують як апаратну платформу для реалізації систем на кристалі (СнК) (System on Chip –SoC) різноманітного призначення. Для їх створення використовують ядро процесора та набір готових параметризованих IP-модулів, кожен із яких реалізує певну функцію пристрою. При необхідності забезпечити специфічні функціональні можливості, розробник створює нові IP-модулі. Взаємодія між процесором та периферійними компонентами відбувається по AMBA/AXI системній шині, яка де-факто стала промисловим стандартом.

Створення IP-ядра програмованого подільника/таймера з підтримкою AXI системної шини, для подальшого використання в СнК, буде наступним кроком дослідження.

References

1. Roger Woods FPGA-based Implementation of Signal Processing Systems. 2nd ed. / Roger Woods, John McAllister, Gaye Lightbody, Ying Yi // Wiley, 2017. - 360p.

2. Mounir Maaref Architecting and Building High-Speed SoCs: Design, develop, and debug complex FPGA-based systems-on-chip // Packt Publishing, 2022. - 426 p.

3. Ross K. Snider Advanced Digital System Design using SoC FPGAs: An Integrated Hardware/Software Approach // Springer, 2023. – 455 p.

4. Jim Ledin Architecting High-Performance Embedded Systems: Design and build high-performance real-time digital systems based on FPGAs and custom circuits // Packt Publishing, 2021. - 376 p.

5. Frank Bruno The FPGA Programming Handbook: An essential guide to FPGA design for transforming ideas into hardware using SystemVerilog and VHDL. 2nd ed. / Frank Bruno, Guy Eschemann // Packt Publishing, 2024. - 550 p.

6. Pong P. Chu FPGA Prototyping by SystemVerilog Examples: Xilinx MicroBlaze MCS SoC. 2nd ed. // Wiley, 2018. - 656 p.

7. 7-th Series FPGAs Clocking Resources. User Guide. UG472 (v1.14) // AMD (Xilinx), 2018. - 114 p. URL: https://docs.amd.com/v/u/en-US/ug472_7Series_Clocking

8. Jim Tatsukawa MMCM and PLL Dynamic Reconfiguration. Application Note. XAPP888 (v1.8) // AMD (Xilinx), 2019. - 20 p. URL: https://docs.amd.com/v/u/en-US/xapp888_7Series_DynamicRecon

9. 7-th Series FPGAs Configurable Logic Block User Guide. UG474 (v1.9) // AMD(Xilinx), 2025. - 81 p. URL: https://docs.amd.com/r/en-US/ug474_7Series_CLB/

10. Nick Mehta Xilinx 7 Series FPGAs: The Logical Advantage. White Paper. WP405 (v1.0) // AMD (Xilinx), 2012. – 9 p. URL: https://docs.amd.com/v/u/en-US/wp405-7Series-Logical-Advantage

11. Vivado Design Suite 7-th Series FPGA and Zynq 7000 SoC Libraries Guide. UG953 (v2024.2) // AMD (Xilinx), 2024. - 622 p. URL: https://docs.amd.com/r/en-US/ug953-vivado-7series-libraries/

12. Sarah Harris, David Harris Digital Design and Computer Architecture, RISC-V Edition .// Morgan Kaufmann, 2021. - 592 p.

13. Digilent Nexys A7. Reference Manual. URL: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual

14. 7-th Series FPGAs Data Sheet: Overview. DS180 (v2.6.1) // AMD(Xilinx), 2020. – 19 p. URL: https://docs.amd.com/v/u/en-US/ds180_7Series_Overview