УДК 004.7

ДОЛГОЛЕНКО А.О.,
ЯЦУН В.О.

РЕАЛІЗАЦІЯ ОПЕРАЦІЙНОГО ПРИСТРОЮ СУМАТОРА/ВІДНІМАЧА З ПЛАВАЮЧОЮ КРАПКОЮ ДЛЯ ЯДРА СУПЕРСКАЛЯРНОГО ПРОЦЕСОРА

В цій статті описаний операційний пристрій суматора/віднімача з плаваючою комою для ядра суперскалярного мікропроцесора та спеціалізованих апаратних засобів на ПЛІС. Суматор розроблено як набір комбінаційних схем, без використання елементів пам'яті й мікропрограмного керування. Для реконфігурації суматора на обробку операндів потрібного формату не потребується ніяких додаткових дій, крім подачі на його керуючі входи сигналу оброблюваного формату.

This article describes addition/subtraction device operating on floating point numbers designed for cores of superscalar microprocessors and specialized hardware built using FPGA. The adder is designed as a set of combinational circuits without memory elements and without microprogram management. No additional actions are required for the adder to be reconfigured for processing operands of a required format but setting the required format into its control inputs.

Ключові слова: суматор, суперскалярний мікропроцесор, CISC, RISC.

Вступ

Починаючи з мікропроцесорів *Intel P6* (1995 р.) та *AmD K5* (1996 р.), для побудови мікропроцесорів з архітектурою *x86-64* використовується технологія *OoOE (Out-of-Order Execution)*, що заснована на реалізації обмеженої архітектури потоку даних [1]. Такі мікропроцесори отримали назву суперскалярних мікропроцесорів [2], або мікропроцесорів з архітектурою *CISC-RISC («CISC-outside RISC-inside»),* де: *CISC - Complex Instruction Set Computing* (набір команд в архітектурі *x86-64*), *RISC - Reduced instruction set computing* (скорочений набір команд, що реалізується множиною операційних пристроїв мікропроцесора.) В процесі роботи таких мікропроцесорів *CISC* – команди, що виконуються відповідно до лічильника команд, декодуються на множину *RISC*– команд [3-5] (їх також називають: *uop, micro-ops*, або подібними термінами).

Планування виконання *RISC* – операцій здійснюється, відповідно до архітектури потоку даних [6,1], на підставі готовності до виконання операндів виконуваних *RISC* – операцій. До набування *RISC* – операціями стану готовності до виконання, вони розміщуються в комірках найбільш швидкодіючої пам'яті мікропроцесора, що носять назву в різних джерелах: *Out-of-Order Window,* комірки універсального планувальника, або комірки станції резервування. Кількість цих комірок постійно збільшується, наприклад, у *Intel:* 168 в *Sandy Bridge* (2011 р.), 192 в *Haswell* (2013 р.), 224 в *SkyLake* (2015 р.). *RISC* – операція, котра набула стану готовності, може бути переданою з комірки станції резервування в вільний операційний пристрій, що може її виконати. Таким чином в сучасних мікропроцесорах організовується динамічний паралелізм на рівні *RISC* – операцій.

Метою роботи є розробка швидкодіючого операційного пристрою суматора/віднімача з плаваючою крапкою, який не потребує в своїй роботі ніяких додаткових мікропрограмних засобів і може використовуватися для динамічного розгалуження на рівні *RISC* – операцій як в роботі ядра сучасного суперскалярного мікропроцесора так і, наприклад, при побудові спеціалізованих апаратних засобів на ПЛІС.

Стандарт *IEEE Std 754*™ для арифметики з плаваючою крапкою

Представлення чисел з плаваючою крапкою схоже на широко використовувану форму представлення чисел в наукових обчисленнях і складається з двох частин: значущої частини числа (або мантиси) f і показника ступеня (експоненти, або порядку) e. Число з плаваючою крапкою x представляється у вигляді $\pm, e_x, f_x$ і має значення:

$x = \pm f_x r^{ex}$, де r – основа системи числення (база експоненти, або основа ступені). Стандарт для арифметики з плаваючою крапкою [7] припускає використання $r = 2$ і $r = 10$.

Деякі скорочення, що прийняті в цьому стандарті:

LSB – найменш значущий біт;

MSB – найбільш значущий біт;

NaN – не число.

Формат представлення чисел з плаваючою крапкою показаний на рис. 1.

Рис. 1 – Формат представлення чисел з плаваючою крапкою

Абсолютне значення мантиси нормалізованого числа знаходиться в діапазоні [1, 2). При нормалізації вона зсувається, поки в $| f |$ *MSB* не буде дорівнювати 1. При кожному лівому зсуві виконується операція $e = e - 1$, при кожному правом зсуві $e = e + 1$. Таким чином, двозначна крапка в уявленні f завжди розташована після біта *MSB*.

Біт *MSB* мантиси нормалізованого числа $| f |$, який завжди дорівнює 1, з уявлення числа видаляється і в пам'яті зберігається тільки дрібна частина *f*. В арифметичному пристрої цей прихований біт відновлюється і тим самим сприяє підвищенню точності представлення операндів, не займаючи місця в пам'яті.

Число з плаваючою крапкою має два знаки: знак числа (*sign*), відображається окремим бітом; знак порядку, відображається зміщенням порядку (*bias*).

Для обчислень з плаваючою крапкою стандарт [7] передбачає використання типів даних (форматів), зведених у табл.1.

Так як число 0 не може бути представлено у формі з нормалізованою мантисою, для його представлення існує спеціальний код: $sign = 0 \; V \; 1, \; e + bias = e_{min}, \; f = 0$.

Спеціальний код також існує для представлення *NaN*: $sign = 0 \; V \; 1, \; e + bias = e_{max}, \; f \neq 0$.

Для представлення невизначених результатів, таких як 0/0, використовується значення ±∞: $sign = 0 \; V \; 1, \; e + bias = e_{max}, \; f = 0$.

Коли одне з цих спеціальних значень бере участь як операнд в арифметичній операції, результат операції формується відповідно до наступних правил:

$$
\begin{aligned}
&число/(\pm\infty) = \pm 0;\\
&число \pm 0 = число;\\
&\pm 0\text{-}число = \text{-}число;\\
&\pm 0 (\text{x} V /) число = \pm 0;\\
&(число V \pm\infty V \pm 0)/\pm 0 = \pm\infty;\\
&\pm\infty (+V\text{-}V\text{x}V/) число = \pm\infty;\\
&NaN(+V\text{-}V\text{x}V/)(число V \pm\infty V \pm 0) = NaN,
\end{aligned}
\qquad (1)
$$

не чекаючи завершення операції.

Денормалізоване число це число з плаваючою крапкою у якого експонента дорівнює нулю (e_{min}) і ненульова мантиса: $sign = 0 \; V \; 1, \; e+bias = e_{min}, \; f \neq 0$. При виконанні арифметичної операції з денормалізованим операндом відновлення прихованого біта в ньому не проводиться. Використання денормалізованних операндів дозволяє зробити ефект втрати значимості менш різким. Без використання таких операндів деякі малі значення, які не представимі як нормалізовані числа, округлюються до 0. Таке рішення погіршує, в деяких випадках, точність обчислень з плаваючою крапкою. Наприклад, число $(0.1)_2 2^{126}$ не має нормалізованого подання в одинарному форматі. Якщо ми замість цього числа в подальших розрахунках будемо використовувати 0, то матиме місце «*поступова втрата точності результату*». Однак, реалізація в обчислювальному пристрої можливості обробки денормалізованих чисел може викликати небажані апаратурні, часові і вартісні витрати. У зв'язку з цим, стандарт [7] не вимагає обов'язкової реалізації обробки денормалізованих чисел. Рис. 2 ілюструє область денормалізованих чисел в одиночному форматі стандарту. Вони всі знаходяться у відкритому інтервалі $(0, | min |)$.

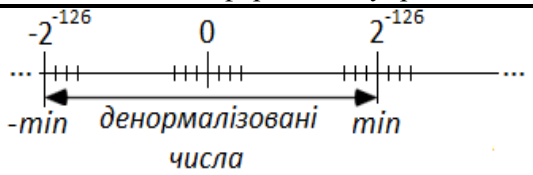

Рис. 2 – Область денормалізованих чисел в одиночному форматі стандарту IEEE

Табл. 1 – Типи даних (точність) для представлення чисел з плаваючою крапкою

Параметр	Половин-на (*SF*)	Одинарна (*F*)	Подвійна (*DF*)	Подвійна розширена (*DEF*)	Учетверена (*QF*)
Всього біт (*n* біт)	16	32	64	80	128
Знак числа (*sign*)	1	1	1	1	1
Порядок (кільк. бітів)	5	8	11	15	15
Зсув порядку (*bias*)	15	127	1023	16383	16383
Значення e_{max}	31	255	2047	32767	32767
Значення e_{min}	0	0	0	0	0
Мантиса (*k* біт)	10+1	23+1	52+1	64+1	112+1
Діапазон уявлення мантиси	$[1,2 - 2^{-10}]$	$[1,2 - 2^{-23}]$	$[1,2 - 2^{-52}]$	$[1,2 - 2^{-64}]$	$[1,2 - 2^{-112}]$
Нуль (± 0)	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$
Денормалізоване число	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$
Нескінченність ($\pm\infty$)	$\pm, 31, f=0$	$\pm, 255, f=0$	$\pm, 2047, f=0$	$\pm, 32767, f=0$	$\pm, 32767, f=0$
Не число (*NaN*)	$\pm, 31, f\neq 0$	$\pm, 255, f\neq 0$	$\pm, 2047, f\neq 0$	$\pm, 32767, f\neq 0$	$\pm, 32767, f\neq 0$
Число	$e \in [-15, 16]$ $(\pm 1.f \; x2^e)$	$e \in [-127, 128]$ $(\pm 1.f \; x2^e)$	$e \in [-1023,1024]$ $(\pm 1.f \; x2^e)$	$e \in [-16383,16384]$ $(\pm 1.f \; x2^e)$	$e \in [-16383,16384]$ $(\pm 1.f \; x2^e)$
Еквівалентне число десяткових цифр	≈ 3.3	≈ 7.2	≈ 15.9	≈ 19.2	≈ 34.0

Стандарт [7] також визначає виконання чотирьох базових арифметичних операцій (додавання, віднімання, множення, ділення), а також кореня квадратного, що стосується очікуваної точності в їх результатах. В принципі результати обчислень повинні відповідати результатам, які були б отримані, при виконанні всіх проміжних розрахунків з нескінченною точністю.

Алгоритм виконання операцій додавання/віднімання з плаваючою крапкою

Виконання арифметичних операцій додавання/віднімання над операндами з плаваючою крапкою концептуально просте. Однак необхідно проявляти обережність у реалізації обладнання для забезпечення коректності та уникнення невиправданої втрати точності. Крім того має бути реалізована можливість обробки будь-яких винятків.

Розглянемо виконання додавання $c = a+b$, тому що віднімання зводиться до додавання додаткового коду зображення мантиси f_b від'ємника: $(\pm f_a \, 2^{ea}) + (\pm f_b \, 2^{eb}) = \pm f_c \; 2^{ec}$.

Ця операція вимагає, щоб порядки обох операндів були вирівняні до початку складання їх мантис. Тому спочатку:

begin {add}

if $e_a = e_b$ *then begin* $e_c = e_b$*; go to M1 end;*

if $e_a > e_b$ *then begin* $e_c = e_a$*;* $f_b = f_b 2^{-(ea-eb)}$ {арифметичне зрушення f_b на e_a-e_b розрядів вправо} *end else begin* $e_c = e_b$*;* $f_a = f_a 2^{-(eb-ea)}$ {арифметичне зрушення f_a на e_b-e_a розрядів вправо} *end;*

M1: $f_c = f_a + f_b$ {Обчислення f_c буде виконуватися в додатковому модифікованому коді (з двома знаковими бітами). Суматор матиме $2(k+1)$ розрядів, де k – розрядність мантис операндів}.

Тепер потрібно виконати нормалізацію f_c, забезпечивши її знаходження в діапазоні: $1 \leq |f_c| < 2$

і виконати корекцію e_c, якщо вона необхідна. Після нормалізації f_c, потрібно виконати округлення f_c і ще раз перевірити її нормалізацію. Для цього:

m2 = 0; M2: if $|f_c| \geq 2$ *then begin* $f_c = f_c 2^{-1}$; $e_c = e_c + 1$; *if m2 = 0 then {*округлення f_c*} begin* $f_c = f_c + [(not\ sign(c)) 2^{-(k+1)}]_{k+2}$ *{*після округлення f_c скорочується до $k+2$ розрядів*}; m2=1; go to M2 end; go to M3 end;*

m4 = 2k; M4: if $|f_c| < 1$ *then begin* $f_c = f_c 2$; $e_c = e_c - 1$; $m4 = m4-1$; *if m4>0 then go to M4 else go to M5 end;*

*{*округлення f_c*} if m2 = 0 then begin* $f_c = f_c + [(not\ sign(c)) 2^{-(k+1)}]_{k+2}$ *{*після округлення f_c скорочується до $k+2$ розрядів*}; go to M5 end;*

M3: if $e_c \geq e_{max}$ *{*переповнення формату результату*} then begin* $e_c = e_{max}$; $f_c = 0$ *go to M6 end;*

M5: if $e_c \leq e_{min}$ *{*втрата значущості, або нуль*} then begin* $e_c = e_{min}$; $f_c = 0$; *go to M6 end else if* $|f_c| \geq 2$ *then begin* $f_c = f_c 2^{-1}$; $e_c = e_c + 1$; *go to M3 end;*

M6: end {add}

Структурна схема суматора/віднімача з плаваючою крапкою, що працює за цим алгоритмом, приведена на рис. 3. Операнди спочатку розпаковуються. Розпакування включає в себе виділення: знака, порядка і мантиси, а також відновлення MSB для кожного з операндів. Під час розпакування також здійснюється конвертація форматів операндів до внутрішнього формату арифметичного пристрою (до формату *QF*). Розпаковані операнди тестуються на наявність серед них винятків: 0, *NaN*, ±∞. При наявності винятків, результат операції формується відповідно до (1) і відправляється на упаковку, минаючи додавання/віднімання.

Блок вирівнювання мантис операндів здійснює правий арифметичний зсув мантиси операнду з меншим порядком. Величина зсуву визначається різницею порядків операндів. Немає необхідності зрушувати мантису з меншим порядком більше ніж на k розрядів.

Таким чином, як тільки різниця порядків перевищує k, можна вважати, що результат виконуваної операції не зміниться, якщо число з меншим порядком замінити на ± 0, відповідно. У зв'язку з цим блок управління може відразу ж ініціювати формування результату, відповідно до (1), в блоці упаковки результату.

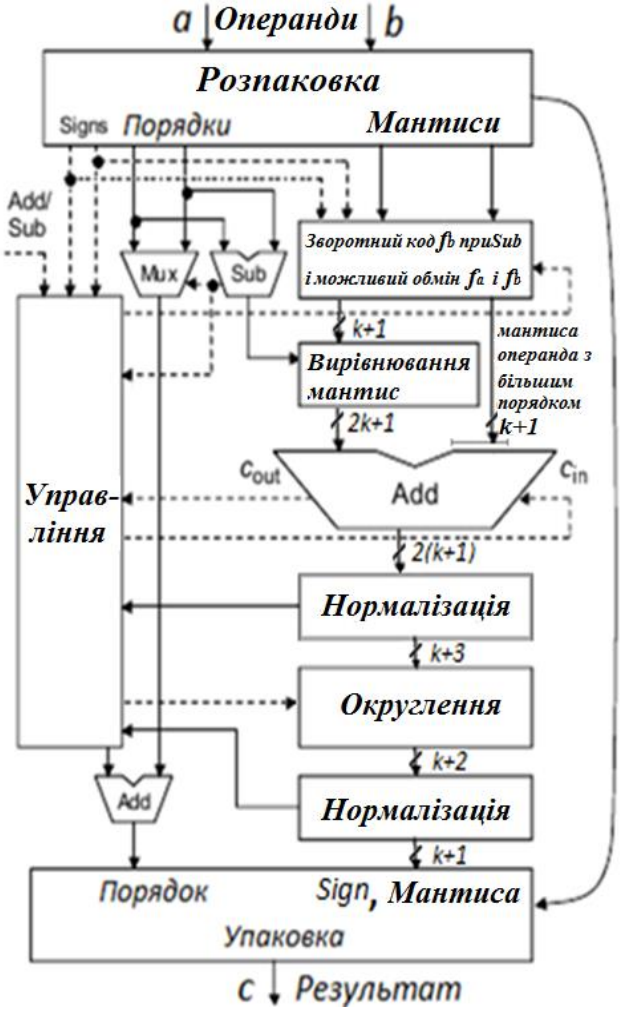

Рис. 3 – Суматор/віднімач з плаваючою крапкою

Різниця двох порядків використовується для визначення зрушуваної мантиси і кількості розрядів арифметичного зсуву вправо для здійснення вирівнювання порядків операндів.

В блоці вирівнювання мантис реалізується можливість зсуву тільки однієї мантиси. Якщо зрушенню підлягає мантиса іншого операнду, то блок можливого обміну забезпечує заміну місць мантис.

Обчислена в суматорі $2(k+1)$ – розрядна мантиса результату f_c нормалізується в першому блоці нормалізації. При цьому f_c може зсуватися як ліворуч, так і праворуч відповідно до вищеприведеного алгоритму. Нормалізована $(k+3)$ – розрядна мантиса f_c округлюється до $k+2$ розрядів у блоці округлення. Після округлення вона знову може втратити нормалізацію. При цьому в другому блоці нормалізації буде виконана її повторна нормалізація до $k+1$ розрядів.

Приклад: Обчислимо різницю чисел $\boldsymbol{a} = [37.625]_{10}$ і $\boldsymbol{b} = [3.852]_{10}$, представлених операндами з плаваючою крапкою половинної точності.

Рішення:

$\boldsymbol{a} = 2^5*1.0010110101$

$\boldsymbol{b} = 2^1*1.1110110100$

$\boldsymbol{a} = 2^5*01.0010110101$ {Знак і k - розрядна f_a }.

$\boldsymbol{b} = 2^5*11.11100001001011100111$ {Знак і $2k$ – розрядна інвертована мантиса f_b на виході блоку вирівнювання мантис (в цьому блоці для вирівнювання порядків операндів був виконаний арифметичний зсув мантиси операнда з меншим порядком на 4 розряду вправо)}.

111.11100001001011100111{f_b на першому вході суматора, її знак продубльований}.

001.0010110100 {f_a на другому вході суматора, її знак продубльований}.

1{Вхід c_{in} суматора, для формування додаткового коду f_b}.

001.00001110001011010000 {2 знакові розряди та $2k$ – розрядна мантиса fc на виході суматора}.

001.00001110001 {2 знакові розряди і $(k+1)$ розрядна мантиса fc на виході першого блоку нормалізації (на вході блоку нормалізації fc вже була нормалізованою, тому в блоці тільки проведено її усічення без зміни порядку результату)}.

001.00001110001

+1 {Для округлення fc до k розрядів у блоці округлення}.

001.0000111001 {2 знакових розряду і k – розрядна мантиса fc на виході блоку округлення}.

01.0000111001 {Знаковий розряд і k – розрядна мантиса fc на виході другого блоку нормалізації (при виконанні округлення fc її нормалізація не порушилась, тому в цьому блоці тільки проведено видалення одного знакового розряду без зміни порядку результату)}.

$\boldsymbol{c}=2^5*1.0000111001$ {Різниця $\boldsymbol{c} = [33.781]_{10}$}.

Блок додавання/віднімання мантис з реконфігурованою структурою

Розроблюваний суматор орієнтований на обробку всіх форматів чисел з плаваючою крапкою, що передбачені стандартом [7] (див. табл. 1). Таких форматів п'ять: половинна точність (*SF*) – 16 розрядів, одинарна (*F*) – 32, подвійна (*DF*) – 64, подвійна розширена (*DEF*) – 80 та учетверина (*QF*) – 128. Для досягнення цієї мети блок додавання/віднімання мантис повинен мати можливість оброблювати, відповідно формату: 12, 25, 54, 66 та 114 бітні мантиси (разом із знаковим та прихованим бітом). Структурна схема такого блоку показана на рис. 4. В залежності від оброблюваного формату, мантиси x та y, що представлені в додатковому коді, надходять на інформаційні входи блоку, як це показано на рис 4, а на керуючий вхід однорозрядних мультиплексорів подається 0 тільки в тому мультиплексорі, що відповідає оброблюваному формату операндів. При виконанні віднімання на вхід вхідного перенесення суматора $\sum_i$ і керуючий вхід мультиплексорів додатково подається ***1***, що забезпечує передачу на вхід блоку додаткового коду від зображення мантиси y**.**

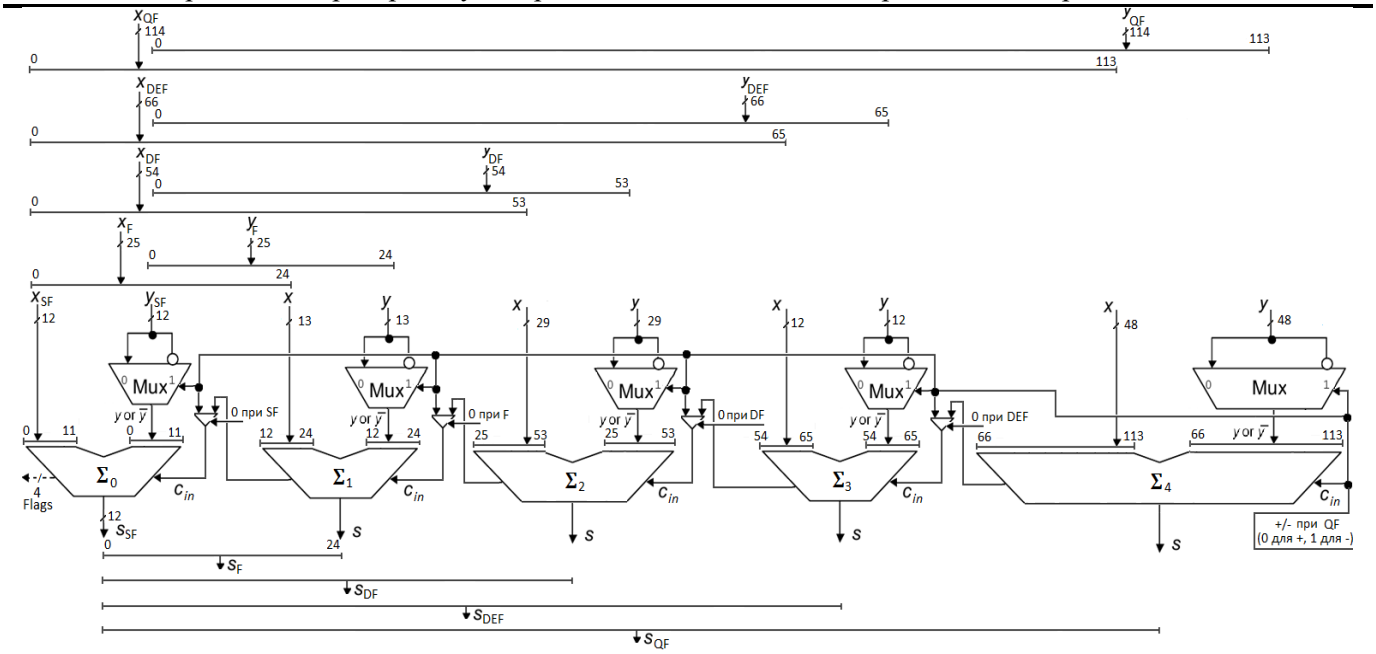

Рис. 4 – Структурна схема блоку додавання/віднімання мантис з реконфігурованою структурою

Мантиси *x* і *y*, при обробці операндів формату *QF*, подаються на входи всіх п'яти суматорів. При цьому на керуючі входи всіх однорозрядних мультиплексорів з блоку керування (див. рис. 3) на вищеописані входи подається код 1, що забезпечує передачу вихідного переносу суматора $\sum_i$, через перший вхід ***i***-го однорозрядного мультиплексора на вхід переноса суматора $\sum_{i-1}$.

Мантиси *x* і *y*, при обробці операндів формату *DEF*, через відповідні мультиплексори подаються на входи чотирьох старших суматорів ($\sum_0$ - $\sum_3$). При цьому на керуючий вхід однорозрядного мультиплексору, що зв'язаний своїм виходом із входом переноса $\sum_3$, з блоку керування подається код 0, що забезпечує блокування вихідного переносу суматора $\sum_4$ та передачу на 4 старших суматори схеми, в залежності від виконуваної операції, прямого чи додатного коду від зображення мантиси ***y***.

Мантиси *x* і *y*, при обробці операндів форматів: *DF*, *F* та *SF* подаються, відповідно, на входи суматорів: $\sum_0$ - $\sum_2$, $\sum_0$ - $\sum_1$ та $\sum_0$. При цьому з блоку керування подається код 0 на керуючий вхід тільки того однорозрядного мультиплексора, що зв'язаний своїм виходом із входом переноса суматора молодших розрядів операндів.

Для мінімізації затримки розповсюдження сигналу в блоці додавання/віднімання мантис був проведений аналіз [9] вибору конкретної схеми реалізації суматорів $\sum_0$ - $\sum_4$, на підставі відомих схем побудови суматорів [8].

Біти станів: *Zero*, *Sign flag*, *Overflow* та *Carry flag* в блоці додавання/віднімання мантис формуються одночасно з формуванням результату виконання операції, представленого додатковим кодом. Виходи суматора мають 4 додаткові виходи, призначені для передачі архітектурних станів процесора. Сформовані архітектурні стани, зберігаються в спеціально виділених бітах регістра станів. Так, наприклад, в архітектурі *x86-64* [2] як регістр станів використовується 64 – розрядний регістр *RFLAGS*.

Блок розпаковки операндів

Схема блоку розпаковки операндів приведена на рис. 5. Її основу складають дві вхідні 128-розрядні шини упакованих операндів: *pack bus x* та *pack bus y* та чотири вихідні шини розпакованих операндів: 114-розрядні шини мантис операндів *unpack bus* M_x та *unpack bus* M_y, а також 15-розрядні шини порядків операндів *unpack bus* E_x та *unpack bus* E_y. Незалежно від формату оброблюваних

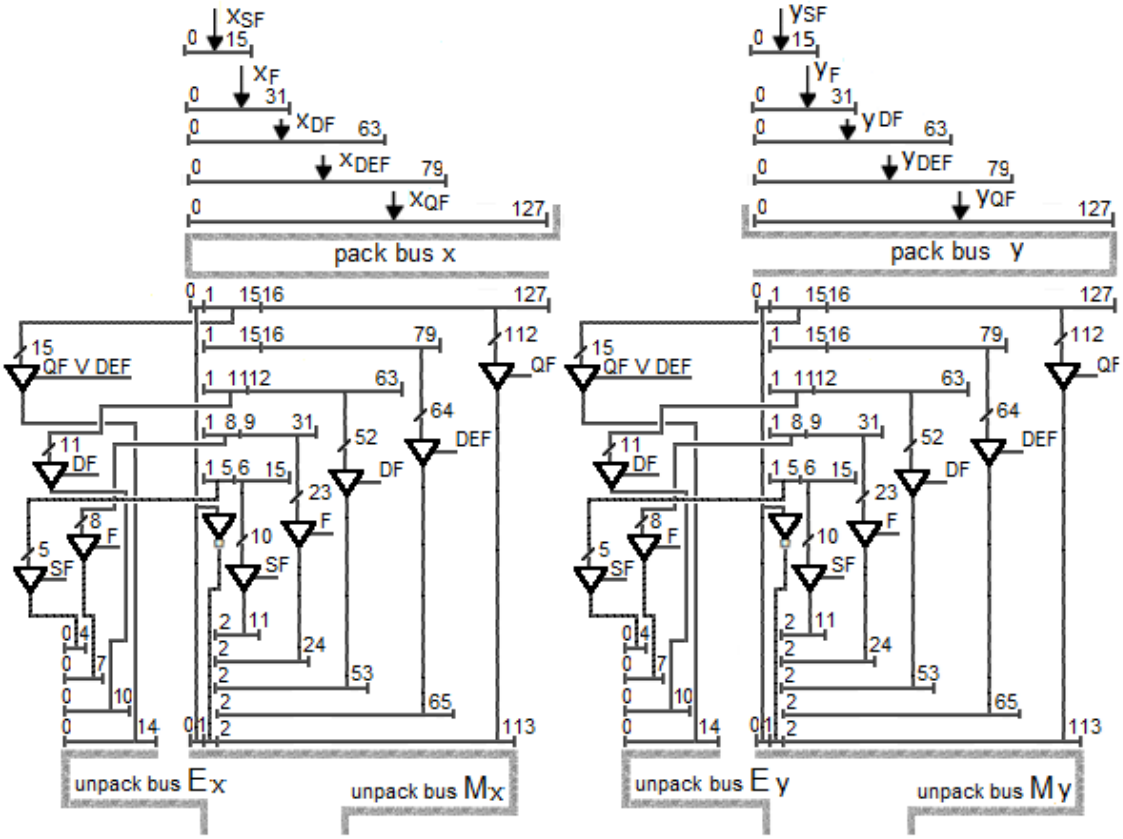

Рис. 5 – Функціональна схема блоку розпаковки операндів

операндів *x* та *y* вони завжди, на початку виконання операції додавання/віднімання, поступають із блоку регістрів з плаваючою крапкою загального призначення в старшу частину своєї вхідної шини. Розпаковка операндів здійснюється шляхом виконання потрібних зв'язків між вхідними шинами упакованих операндів та вихідними шинами порядків і мантис блоку розпаковки. Для створення потрібних зв'язків, в залежності від оброблюваного формату використовуються буфери з трьома станами, що створюють конструктивне «або» на вихідних шинах блоку. Для запобігання відмови блоку регістрів з плаваючою крапкою загального призначення потрібно забезпечити умову, що в кожний момент часу, в залежності від оброблюваного формату операндів, повинен бути відкритим тільки один з буферів, що поєднуються конструктивним «або» на вихідній шині блоку.

При розпаковці операндів в 0-ий розряд кожної з вихідних шин мантис з відповідної вхідної шини напряму заноситься знак операнда, а в 1-ий розряд цих же шин, за допомогою інверторів, із знаків операндів відновлюються приховані біти.

Блок нормалізації

Якщо в результаті виконання операцій в блоках обробки мантис та порядків блок керування суматором з плаваючою крапкою виявив виникнення наступних ситуацій: признак *Zero* (признак рівності нулю мантиси результату), *NaN* («не число»), ±∞ («нескінченність»), то блок керування зразу ж ініціює виконання операції упаковки числа відповідного результату, пропускаючи роботу блоку нормалізації. В іншому випадку відбувається нормалізація результату. Функціональна схема блоку нормалізації приведена на рис. 6.

Основу блоку нормалізації складають вхідна 116-розрядна шина *bus* M_s, що є вихідною шиною блоку обробки мантис і містить: в 0-му розряді сигнал *Overflow*, в 1-му розряді сигнал *Carry flag*, в 2-му розряді сигнал *Sign flag*, в 3 – 115 розрядах значення мантиси результату та 113-розрядна вихідна шина *bus normalize* M_s, що містить: в 0-му

розряді сигнал *Sign* ***s***, а в розрядах 1 – 112 нормалізовану мантису без прихованого біта.

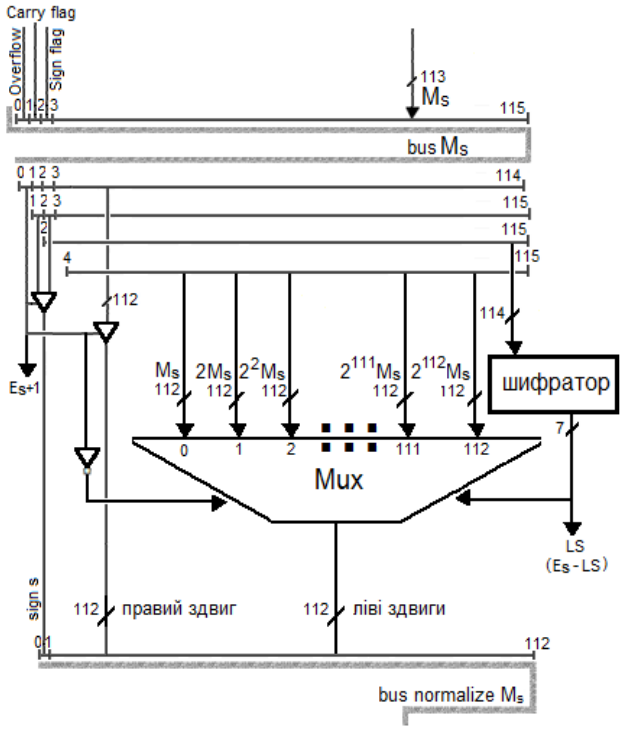

Рис. 6 – Функціональна схема блоку нормалізації

Всі операції нормалізації можливо розділити на 3 групи.

1. Мантиса результату є нормалізованою (мантиса результату передається із вхідної шини на вихідну без здвигу з видаленням прихованого біту).

2. Переповнення розрядної сітки (виконується правий здвиг мантиси на 1 розряд з видаленням прихованого біту та видається сигнал E_s+1 в блок керування суматором з плаваючою крапкою для збільшення порядку результату на 1).

3. Зменшення точності результату (виконуються ліві здвиги, що дорівнюють кількості «0» до першої правої значущої «1» в мантисі для додатного результату, або кількості «1» до першого правого значущого «0» в мантисі для від'ємного результату з видаленням прихованого біту та видається сигнал E_s - *LS* в блок керування суматором з плаваючою крапкою для зменшення порядку результату на число лівих здвигів).

Робота блока нормалізації ґрунтується на організації безперебійної роботи конструктивного «або» на вихідній шині. Для цього вивід з третього стану 112 – розрядного буферу, що виконує правий здвиг мантиси результату, та 112 – розрядного мультиплексора, що виконує ліві здвиги та безздвигову передачу, здійснюється відповідно, прямим та інверсним значенням одного й того ж сигналу *Overflow*. Така організація роботи схеми виключає одночасний вихід з третього стану буфера і мультиплексора.

Для формування знака результату його передачі використовується двухвходовий однорозрядний мультиплексор без третього стану, на керуючий вхід якого подається сигнал *Overflow*. При переповненні результату в якості його знаку передається *Carry flag*, в інших випадках – *Sign flag*.

Визначення кількості лівих здвигів, або безздвигової передачі здійснюється за допомогою шифратора. Виходи шифратора подаються на керуючі входи мультиплексора та в блок керування суматором з плаваючою крапкою, де вони використовуються для збільшення порядку результату на число лівих здвигів.

Видалення прихованого біта та формування здвинутої мантиси на потрібну кількість бітів вліво забезпечується зв'язками між вхідною шиною та відповідними розрядами на входах мультиплексора.

Блок обробки порядків

Схема блоку обробки порядків приведена на рис. 7. Основу блоку обробки порядків складають дві вхідні 15-розрядні шини порядків операндів: *unpack bus* E_x і *unpack bus* E_y та 15-розрядна вихідна шина порядку результату *bus* E_s. Незалежно від формату оброблюваних операндів *x* та *y* їх прядки, на початку виконання операції додавання/віднімання, поступають на вхідні шини блоку із блоку розпаковки операндів. Кожна з вхідних шин паралельно навантажена на чотири комбінаційні схеми: схему виявлення мінімального значення порядку, схему виявлення максимального значення порядку, віднімач порядків та мультиплексор порядків.

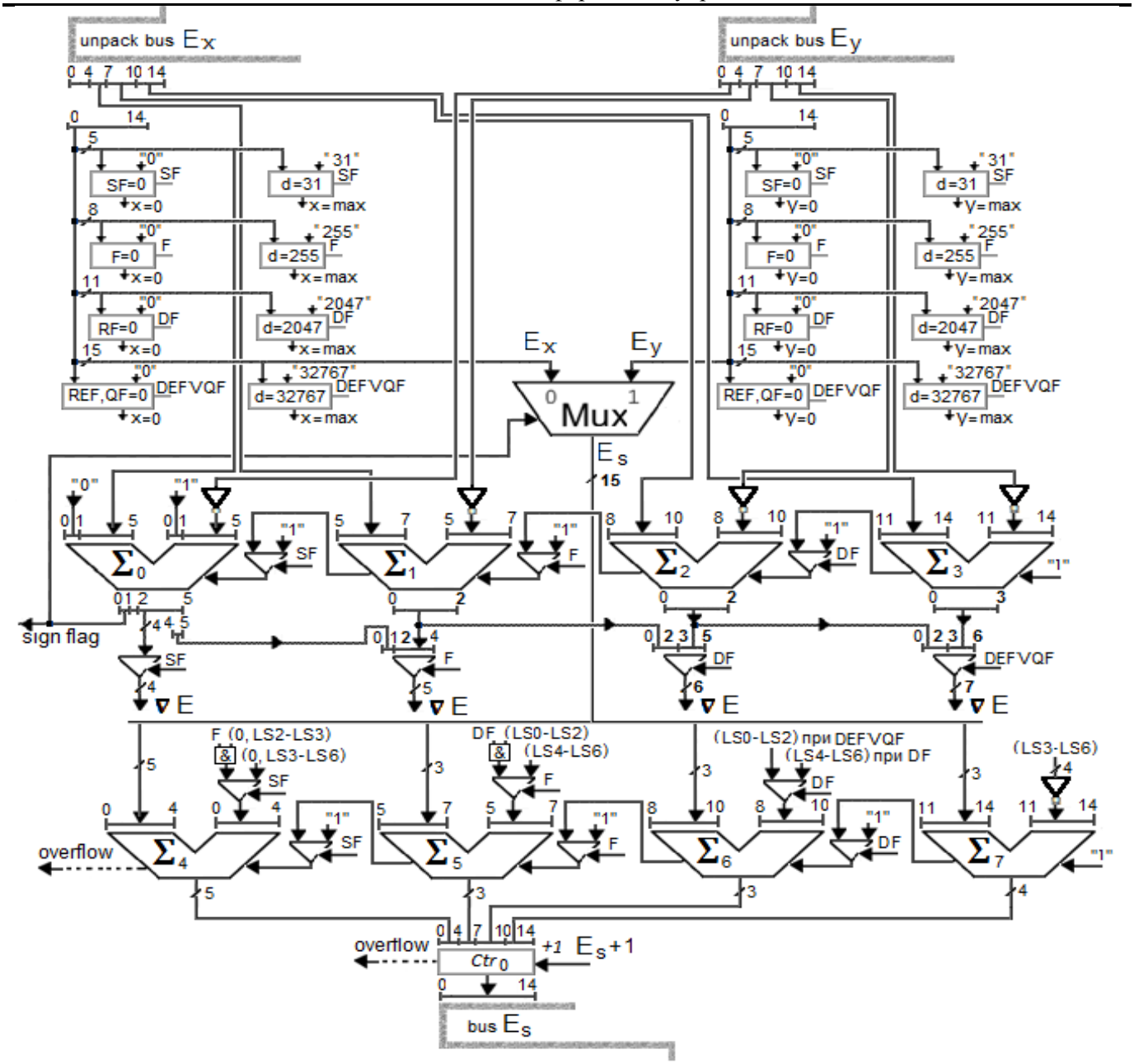

Рис. 7– Структурна схема блоку обробки порядків

Схема виявлення мінімального значення порядку, в залежності від формату оброблюваних операндів, порівнює відповідну кількість старших розрядів вхідної шини з нулем (*MSB* порядку, незалежно від формату представлення операндів, завжди знаходиться в нульовому розряді вхідної шини). П'ять оброблюваних форматів операндів мають чотири різних формати для представлення порядку (див. табл. 1.1). Схема виявлення мінімального значення порядку складається з чотирьох схем порівняння з трьома станами, виходи котрих поєднуються за допомогою конструктивного «або» разом в сигнал $x=0$ (або $y=0$) для блока керування виконанням операцією. Вивід кожної з цих схем із третього стану, здійснюється сигналом обробки відповідного формату операндів, що надходять з блока керування виконанням операцією (наприклад, при обробці подвійного формату $DF=1$).

Схеми виявлення максимального значення порядків порівнюють порядки операндів з константою максимального значення порядку, що для кожного з форматів порядків має своє значення (див. табл. 1.1). При виникненні сигналу $x=max$ (або $y=max$) число x (чи y) вважається за нескінченість без перевірки, відповідно, $M_x=0$ (чи $M_y=0$). Таке припущення призводить до

того, що якщо один з оброблюваних операндів був *NaN* (не числом), то воно буде автоматично прирівняне до ±∞. Наскільки таке рішення відповідає вимогам [7], потребує подальшого дослідження. При формуванні сигналу $x = max$ (або $y = max$) блок керування виконанням операцією, не очікуючи завершення виконання операції, ініціює перехід до упаковки результату відповідно до (1). Наприклад, якщо при виконанні операції x-y виникає сигнал $y = max$, то в якості результату буде упаковано: $s = -y$ (нескінченість із зміненим знаком).

Схема віднімача порядків складається з суматорів: $\sum_0 - \sum_3$, лінійок інверторів, що зв'язані з другим входом відповідного суматора та буферів з трьома станами, котрі своїми входами поєднані з виходами суматорів, відповідно до рис. 7, а їх виходи утворюють конструктивне «або», формуючи в підсумку семи розрядну шину $\blacktriangledown E$, та сигнал *sign flag*. За допомогою віднімача реалізується операція $E_x - E_y$. Віднімач реалізовано, як суматор, що оброблює кількість розрядів шин порядків відповідно до оброблюваного формату операндів. Для цього на других входах суматорів формується додатний код E_y. Схема віднімача реалізована таким чином, що без знакові порядки доповнюються знаковим розрядом. Знак $\boldsymbol{E_x}$ завжди є додатнім, а знак $\boldsymbol{E_y}$ – від'ємним. Завдяки цьому на віднімачі обраховується число $\blacktriangledown E$ в додатному коді, що представляє собою кількість розрядів на які виконується, в блоці вирівнювання мантис, правий здвиг мантиси числа з меншим порядком. Якщо $\blacktriangledown E \geq 0$ (при цьому *sign flag = 0*), то більшим за абсолютним значенням є число x. При цьому через нульовий вхід мультиплексора передається для подальшої обробки E_x в якості попереднього порядку результату E_s. Разом з цим в блок вирівнювання мантис передається число $\blacktriangledown E$, на яке в ньому буде виконано правий здвиг мантиси числа y. При $\blacktriangledown E < 0$ (*sign flag = 1*) більшим за абсолютним значенням є число y. При цьому, в якості попереднього порядку результату E_s, через перший вхід мультиплексора, передається для подальшої обробки E_y. Разом з цим в блок вирівнювання мантис передається число $\blacktriangledown E$, на яке в ньому буде виконано правий здвиг мантиси числа x.

З виходу мультиплексора E_s подається для подальшої обробки на перші входи другого віднімача, що зібраний з суматорів: $\sum_4 - \sum_7$. На другі входи цих суматорів, в залежності від оброблюваного формату операндів та за допомогою мультиплексорів й зв'язків з рис. 7, видається число *LS*. Це число формується в блоці нормалізації й несе інформацію про кількість лівосторонніх здвигів мантиси результату, які були виконані над нею в процесі нормалізації результату. В процесі цієї операції значення E_s зменшується на *LS*. З виходів другого віднімача E_s подається для подальшої обробки на входи комбінаційної схеми інкрементного лічильника.

Завдяки використанню лічильника значення E_s може збільшитися на одиницю у тому випадку, коли в результаті округлення мантиси результату до кількості розрядів, що відповідають оброблюваному форматові операндів, виникло переповнення її розрядної сітки. Після цієї корекції E_s, за допомогою схеми формування сигналу *overflow* здійснюється перевірка на переповнення, в процесі округлення результату, розрядної сітки порядку результату. Сигнал *overflow* видається в блок упаковки результату, де він при *overflow = 1* використовується для упаковки $y = \pm\infty$ (в залежності від $sign\ s$).

Висновки

В цій роботі описана структурна схема суматора з плаваючою крапкою. Перевагами розробленої схеми є те, що вона виконана як набір комбінаційних схем, без використання елементів пам'яті й мікропрограмного керування.

Операція сумування операндів починається й завершується за один цикл роботи суматора. На входи суматора з плаваючою крапкою операнди подаються з блоку регістрів загального призначення, а сигнал оброблюваного формату надходить з відповідної станції резервації, що зберігає *RISC* операцію виконуваної *CISC* команди типу сумування.

Результат виконаної операції заноситься в один з регістрів загального призначення та в буфер переупорядкування результатів. Таким чином, розроблений суматор з плаваючою крапкою орієнтований на використання в якості

одного із операційних пристроїв суперскалярного мікропроцесорного ядра з архітектурою *CISC- RISC.*

Для реконфігурування суматора на обробку операндів потрібного формату не потребується ніяких додаткових дій, крім подачі на його керуючі входи сигналу оброблюваного формату.

Виконана розробка буде корисною при проектуванні операційного пристрою сумування операндів з фіксованою чи з плаваючою крапкою, як при побудові ядра мікропроцесора з суперскалярною архітектурою, сумісною з сімейством x86-64, так і при побудові спеціалізованих апаратних засобів на ПЛІС.

Список посилань

1. Y. Patt, W. Hwu, et al, Experiments with HPS, a Restricted Data Flow Micro architecture for High Performance Computers, Digest of Papers, COMPCON 86, (March 1986), pp. 254-258.
2. John L. Hennessy, David A. Patterson. Computer Architecture. A Cuantitative Approach, USA, Morgan Kaufmann, 2012 – 497 p.+ add-ins.
3. Kanter, David (September 25, 2010). "Intel's Sandy Bridge Microarchitecture" (http://www.realworldtech.com/sandy-bridge/).
4. Kanter, David (November 13, 2012). "Intel's Haswell CPU Microarchitecture" (http://www.realworldtech.com/haswell-cpu/).
5. Луцький Г.М., Долголенко О.М., Аксьоненко С.В., Сторожук В.О. Моделювання обмеженої реалізації архітектури потоку даних в структурі суперскалярного процесора. Київ,- Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2014. – № 60. – с. 83-94.
6. J. Dennis: Data Flow Supercomputers; IEEE Computer, pp. 48-56, Nov. 1980.
7. IEEE 754: Standard for Binary Floating-Point Arithmetic [Електронний ресурс] / 3 апрель 2014. – URL: http://grouper.ieee.org/groups/754/.
8. Behrooz Parhami. Computer Arithmetic. Algorithms and Hardware Designs, New York, Oxford University Press, 2000– 491 p.
9. Яцун В.О., Долголенко О.М. Блок додавання та віднімання мантис з реконфігурованою структурою. Матеріали наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка – ІОТ-2016». Київ 25 – 27 квітня 2016, – с. 158-162 (http://fiot.kpi.ua/wp-content/uploads/2016/06/IOT-2016-OT.pdf).