АРИФМЕТИЧНІ ЦИФРОВІ ПРИСТРОЇ

Вступ

Арифметичні цифрові пристрої виконують операції над двійковими числами. Їхня основа — комбінаційні логічні схеми, виходи яких після завершення перехідного процесу визначаються поточними значеннями входів. Найважливішим арифметичним вузлом є суматор. На його основі реалізують не лише додавання, а й віднімання, збільшення та зменшення числа, порівняння, обчислення адрес і частину операцій арифметико логічного пристрою.

Матеріал починається з однорозрядного додавання, переходить до напівсуматора і повного суматора, а потім — до багаторозрядного суматора з послідовним поширенням перенесення. Окремо розглядаються віднімання через додатковий код, ознаки перенесення і переповнення, принцип прискореного формування перенесення та місце суматора в арифметичній частині процесора.

1 Двійкове додавання і фіксована розрядність

1.1 Розряд двійкового числа

Двійкове число складається з бітів. Біт із номером i має вагу 2^i. Правий біт має номер 0 і називається молодшим значущим бітом (Least Significant Bit, LSB). Лівий біт має найбільшу вагу і називається старшим значущим бітом (Most Significant Bit, MSB). Наприклад, число 0101_2 дорівнює $0\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0=5_{10}$.

У цифровому пристрої наперед задається кількість бітів. Чотирибітний вузол зберігає лише чотири біти результату. Біт, що виникає лівіше за MSB, не входить до слова результату, але може зберігатися як ознака перенесення.

1.2 Базові правила додавання

A	B	A+B	Біт суми S	Перенесення C
0	0	0_2	0	0
0	1	1_2	1	0
1	0	1_2	1	0
1	1	10_2	0	1

Запис $1+1=10_2$ означає, що в поточному розряді залишається 0, а одиниця передається до наступного старшого розряду. Якщо до двох операндів додається вхідне перенесення, можливий результат $1+1+1=11_2$: біт суми дорівнює 1, а вихідне перенесення також дорівнює 1.

Для трьох однорозрядних входів арифметична сума t може дорівнювати 0, 1, 2 або 3. Біт результату визначається як S=t mod 2, а перенос — як ціла частина від ділення t на 2.

2 Напівсуматор

2.1 Призначення і сигнали

Напівсуматор (Half Adder, HA) — комбінаційний пристрій для додавання двох однорозрядних чисел A і B. Він формує два виходи: S (Sum — сума) та C (Carry — перенесення). Разом виходи CS утворюють повний двобітний результат.

Наприклад, при A=1 і B=1 вихід S дорівнює 0. Це не означає нульовий арифметичний результат, оскільки C=1. Комбінація $CS=10_2$ дорівнює двом.

2.2 Таблиця істинності і рівняння

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Сума дорівнює одиниці, коли входи різні. Таку функцію виконує виключне АБО (exclusive OR, XOR). Перенесення дорівнює одиниці лише за двох одиничних входів, що відповідає логічному І (AND).

$$S=A \oplus B \qquad C=A \wedge B$$

Через базові операції AND, OR і NOT рівняння суми можна записати як $S=\overline{A} \wedge B \vee A \wedge \overline{B}$. Риска над змінною означає логічне заперечення (NOT), символ ∧ — логічне І (AND), символ ∨ — логічне АБО (OR), символ ⊕ — виключне АБО (XOR).

Напівсуматор не має входу перенесення. Тому він придатний для ізольованого додавання двох бітів, але його недостатньо для довільного розряду багаторозрядного суматора.

3 Повний суматор

3.1 Вхідне і вихідне перенесення

Повний суматор (Full Adder, FA) додає три однорозрядні величини: A, B та Cin (Carry In — вхідне перенесення). Він формує S і Cout (Carry Out — вихідне перенесення). Арифметично вузол обчислює A+B+Cin.

A	B	Cin	A+B+Cin	S	Cout
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	2	0	1
1	0	0	1	1	0

A	B	Cin	A+B+Cin	S	Cout
1	0	1	2	0	1
1	1	0	2	0	1
1	1	1	3	1	1

S дорівнює одиниці за непарної кількості одиниць на входах. Cout активний, якщо одиниці присутні щонайменше на двох із трьох входів. Тому Cout також називають функцією більшості для трьох змінних.

3.2 Булеві рівняння

$$S=A\oplus B\oplus Cin$$

$$Cout=A\wedge B \vee Cin\wedge(A\oplus B)$$

Добуток $A\wedge B$ означає, що розряд сам генерує перенос. Вираз $A\oplus B$ означає, що входи різні, а тому наявний Cin буде переданий на Cout. Альтернативна форма $Cout=A\wedge B \vee A\wedge Cin \vee B\wedge Cin$ безпосередньо відображає правило «принаймні дві одиниці».

3.3 Реалізація з двох напівсуматорів

Перший НА додає A і B та формує $S_1=A\oplus B$ і $C_1=A\wedge B$. Другий НА додає S_1 і Cin та формує остаточну суму $S=S_1\oplus Cin$ і перенос $C_2=S_1\wedge Cin$. Переноси об’єднуються операцією OR: $Cout=C_1\vee C_2$.

Етап	Входи	Сума	Перенесення
Перший НА	A, B	$S_1=A\oplus B$	$C_1=A\wedge B$
Другий НА	S_1, Cin	$S=S_1\oplus Cin$	$C_2=S_1\wedge Cin$
Об’єднання	C_1, C_2	—	$Cout=C_1\vee C_2$

Приклад. При A=1, B=1 і Cin=1 маємо $1+1+1=3=11_2$. Отже, S=1 і Cout=1. Через два НА: $S_1=0$, $C_1=1$; далі S=1, $C_2=0$; тому $Cout=1\vee 0=1$.

4 Багаторозрядний суматор

4.1 Ripple Carry Adder

Суматор із послідовним поширенням перенесення (Ripple Carry Adder, RCA) утворюється каскадним з’єднанням повних суматорів. Вихід Cout розряду i підключається до входу Cin розряду i+1. Для чотирьох бітів утворюються проміжні перенесення C1, C2, C3 та фінальне C4.

Розряди нумеруються справа наліво. Розряд 0 обробляє біти з вагою 2^0, розряд 1 — біти з вагою 2^1, а старший розряд 3 — біти з вагою 2^3. Позначення C_i завжди належить перенесенню, що входить у розряд i, а C_{i+1} — перенесенню, сформованому цим розрядом.

$$A_i+B_i+C_i=2C_{i+1}+S_i$$

Рівняння показує арифметичний зміст двох виходів повного суматора: S_i має вагу 2^i, а C_{i+1} переходить у наступну позицію і тому фактично має подвійну вагу відносно поточного розряду. Однакова структура повторюється для будь якої розрядності; змінюється лише кількість каскадів.

Розряд	Входи повного суматора	Виходи
0 молодший	A0, B0, C0	S0, C1
1	A1, B1, C1	S1, C2
2	A2, B2, C2	S2, C3
3 старший	A3, B3, C3	S3, C4

Під час звичайного додавання C0=0. Результат слова записується як S3S2S1S0. Для беззнакової арифметики C4 можна приєднати зліва і отримати п’ятибітний повний результат C4S3S2S1S0.

4.2 Алгоритм порозрядного розрахунку

Для кожного розряду i, починаючи з i=0, виконується:

1. обчислення $t_i=A_i+B_i+C_i$;
2. визначення $S_i=t_i \bmod 2$;
3. формування $C_{i+1}=1$, якщо $t_i\geq 2$, і $C_{i+1}=0$ в іншому випадку;
4. перехід до наступного старшого розряду.

Розрахунок не можна починати зі старшого біта, оскільки значення C_i надходить із попереднього молодшого розряду.

4.3 Детальний приклад 0101_2 плюс 0011_2

i	A_i	B_i	C_i	t_i	S_i	C_{i+1}
0	1	1	0	2	0	1
1	0	1	1	2	0	1
2	1	0	1	2	0	1
3	0	0	1	1	1	0

Отримані біти S3S2S1S0=1000_2. Десяткова перевірка: 0101_2=5, 0011_2=3, тому 5+3=8. Ланцюг C1=C2=C3=1 показує, як одиниця, сформована у молодшому розряді, впливає на старший біт результату.

4.4 Затримка перенесення

Усі FA фізично присутні одночасно, але старший розряд не завжди може завершити обчислення до появи C3. У найгіршому випадку перенос послідовно проходить крізь усі каскади. Найдовший шлях поширення сигналу називається критичним шляхом (critical path).

Приклад 0111_2+0001_2 утворює довгий ланцюг: молодший розряд генерує C1, а наступні розряди з парами 10 пропускають його до C3. Затримка n-розрядного RCA приблизно зростає пропорційно n, тому для великих слів застосовують прискорені способи формування перенесення.

5 Подання знакових чисел і додатковий код

5.1 Беззнакова і знакова інтерпретація

Бітовий код не містить типу числа. Код 1000_2 може означати 8 у беззнаковому форматі або −8 у чотирибітному додатковому коді. Інтерпретацію визначають формат даних і команда, що обробляє значення.

Розрядність n	Беззнаковий діапазон	Додатковий код
Загальний випадок	$0 \dots 2^n-1$	$-2^{n-1} \dots 2^{n-1}-1$
4 біти	0 … 15	−8 … +7
8 бітів	0 … 255	−128 … +127

У додатковому коді MSB виконує роль знакового біта: 0 відповідає невід'ємному числу, 1 — від'ємному. Для визначення модуля від'ємного числа потрібно інвертувати всі біти і додати одиницю.

5.2 Формування протилежного числа

$$-B=\overline{B}+1$$

Наприклад, для $B=0011_2$ інверсія всіх бітів дає $\overline{B}=1100_2$. Додавання одиниці формує 1101_2 — чотирибітний додатковий код числа −3.

У фіксованій n-бітній розрядності обчислення виконуються за модулем 2^n. Саме тому додавання коду −B до A дає ту саму n-бітну різницю, що й звичайне віднімання.

6 Віднімання через суматор

6.1 Основне перетворення

$$A-B=A+(-B)=A+\overline{B}+1$$

Окремий багаторозрядний віднімач не є обов'язковим. Достатньо інвертувати кожний біт B і встановити початкове перенесення C0=1. Додавання одиниці через C0 завершує формування додаткового коду числа B.

6.2 Керований суматор віднімач

Сигнал SUB (Subtract — віднімання) подається на XOR перед кожним входом B_i і одночасно на C0. Властивості XOR забезпечують $B_i \oplus 0=B_i$ і $B_i \oplus 1=\overline{B}_i$.

SUB	Вхід B_i до FA	C0	Операція
0	$B_i \oplus 0=B_i$	0	A+B
1	$B_i \oplus 1=\overline{B}_i$	1	$A+\overline{B}+1=A-B$

Один керувальний сигнал змінює обидві складові операції: інверсію B та додавання одиниці. Тому той самий каскад повних суматорів працює як суматор або віднімач.

6.3 Приклад 7 мінус 3

$A=0111_2$, $B=0011_2$. Інверсія B дає 1100_2, а C0=1. Отже, обчислюється 0111+1100+1.

i	A_i	$\bar{B}_i$	C_i	t_i	S_i	C_{i+1}
0	1	0	1	2	0	1
1	1	0	1	2	0	1
2	1	1	1	3	1	1
3	0	1	1	2	0	1

Результат $S=0100_2=4$, Cout=1. Для беззнакового віднімання Cout=1 означає відсутність позики, оскільки 7≥3.

6.4 Приклад 3 мінус 7

$A=0011_2$, $B=0111_2$. Після інверсії B отримуємо 1000_2, а C0=1 додає необхідну одиницю. Результат $0011+1000+1=1100_2$.

Код 1100_2 має знаковий біт 1. Для визначення модуля інвертуємо його: 0011_2, додаємо одиницю: $0100_2=4$. Отже, 1100_2 представляє −4. Фінальний Cout=0, тому в беззнаковій інтерпретації наявна позика.

7 Carry Borrow і Overflow

7.1 Перенесення Carry

Carry — ознака виходу беззнакового результату за межі слова. Для чотирибітного додавання 1111_2+0001_2 отримуємо $S=0000_2$ і C4=1. Повний результат дорівнює $1\ 0000_2=16$, але у чотирьох бітах зберігається лише 0000_2.

7.2 Позика Borrow

Borrow — позика під час беззнакового віднімання. У реалізації $A+\bar{B}+1$ фінальний Cout має протилежний зміст до Borrow.

$$\mathbf{Borrow=Cout}$$

Cout=1 означає, що A≥B і позика не потрібна. Cout=0 означає, що A<B і беззнаковий результат отримано за модулем 2^n. У конкретній архітектурі процесора домовленість щодо прапорця після SUB необхідно перевіряти за документацією.

7.3 Знакове переповнення Overflow

Overflow — ознака того, що математичний знаковий результат не входить у діапазон заданої розрядності. Суматор при цьому працює правильно і формує молодші n бітів, але їхня знакова інтерпретація не дорівнює математичному результату.

Для додавання Overflow можливий лише тоді, коли операнди мають однакові знаки, а результат — протилежний. Якщо A3, B3 і S3 є знаковими бітами чотирибітних чисел, то:

$$\mathbf{V=A3\wedge B3\wedge \bar{S}3 \vee \bar{A}3\wedge \bar{B}3\wedge S3}$$

Еквівалентне правило використовує переноси біля знакового розряду. C3 входить у MSB, C4 виходить із нього.

$$V=C3 \oplus C4$$

7.4 Приклад переповнення

0111_2+0011_2 відповідає +7+(+3). Чотирибітний суматор формує 1010_2. У додатковому коді 1010_2 означає −6, хоча математичний результат дорівнює +10. Два додатні операнди дали від'ємний код, тому V=1. Фінальний Cout у цьому випадку дорівнює 0, що підтверджує відмінність Carry від Overflow.

Для −3+(−2): $1101_2+1110_2=1\ 1011_2$. Чотирибітний результат 1011_2 дорівнює −5. Він входить у діапазон −8…+7, тому V=0, хоча Cout=1.

7.5 Порівняння ознак

Ознака	Що описує	Основна інтерпретація
Carry C	Перенесення за межі MSB	Беззнакове додавання
Borrow	Позика під час віднімання	Беззнакове віднімання
Overflow V	Вихід за знаковий діапазон	Знакова арифметика

Одна апаратна сума може мати C=1,V=0 або C=0,V=1. Потрібна ознака визначається типом операндів. Carry не замінює Overflow, а Overflow не використовується для перевірки беззнакового діапазону.

8 Суматор із випереджувальним перенесенням

8.1 Причина прискорення

Суматор із випереджувальним формуванням перенесення (Carry Look Ahead Adder, CLA) не очікує послідовного проходження C через кожний FA. Значення переносів виводяться безпосередньо з початкового C0 та властивостей усіх молодших розрядів.

8.2 Generate і Propagate

$$G_i=A_i \wedge B_i \qquad P_i=A_i \oplus B_i$$

Generate G_i дорівнює одиниці при $A_i=B_i=1$. Розряд генерує перенос незалежно від C_i. Propagate P_i дорівнює одиниці, коли A_i і B_i різні. Такий розряд передає вхідний перенос на вихід. Якщо $G_i=P_i=0$, перенос блокується.

$$C_{i+1}=G_i \vee P_i \wedge C_i \qquad S_i=P_i \oplus C_i$$

A_iB_i	G_i	P_i	Поведінка перенесення
00	0	0	Блокування
01	0	1	Поширення
10	0	1	Поширення
11	1	0	Генерування

8.3 Рівняння для чотирьох розрядів

$$C1=G0 \vee P0C0$$

$$C2=G1 \vee P1G0 \vee P1P0C0$$

$$C3=G2 \vee P2G1 \vee P2P1G0 \vee P2P1P0C0$$

$$C4=G3 \vee P3G2 \vee P3P2G1 \vee P3P2P1G0 \vee P3P2P1P0C0$$

Кожний добуток описує можливий шлях перенесення. Наприклад, P3P2G1 означає, що розряд 1 генерує перенос, а розряди 2 і 3 його поширюють. Добуток P3P2P1P0C0 описує проходження початкового перенесення через усі чотири розряди.

CLA використовує більше вентилів і складніше з'єднання, але скорочує кількість послідовних логічних рівнів. Для великої розрядності застосовують групову або ієрархічну організацію переносів.

9 Арифметична частина АЛП

Арифметико логічний пристрій (Arithmetic Logic Unit, ALU; українською АЛП) отримує операнди A і B та код операції. Суматор віднімач формує ADD або SUB, а логічна гілка паралельно обчислює AND, OR або XOR. Мультиплексор (Multiplexer, MUX) вибирає потрібний результат.

Складова	Призначення
Суматор віднімач	Додавання та віднімання
Логічна гілка	AND, OR, XOR та інші логічні операції
MUX	Вибір результату за кодом операції
Логіка прапорців	Формування Z, N, C і V

Прапорець Zero (Z) дорівнює одиниці, якщо всі біти результату нульові. Negative (N) зазвичай повторює знаковий біт результату. Carry (C) і Overflow (V) формуються на основі переносів і використовуються для беззнакової та знакової інтерпретації відповідно.

Комплексний приклад. A=0101_2, B=0111_2, SUB=1. Керовані XOR формують B*=1000_2, C0=1. Суматор обчислює 0101+1000+1=1110_2. У додатковому коді це −2, що відповідає +5−(+7). Cout=0 означає Borrow=1 для unsigned, а V=0 підтверджує коректність знакового результату.

10 Перевірка арифметичних схем

Функціональна перевірка підтверджує правильність результату та ознак для заданих входів. Часова перевірка визначає, чи встигає схема стабілізуватися до моменту фіксації результату. Правильна таблиця істинності не гарантує достатньої швидкодії.

Для суматора віднімача перевіряються такі класи випадків:

- додавання без перенесення і з одиничним перенесенням;
- довгий ланцюг поширення перенесення, наприклад 0111+0001;
- фінальний Carry, наприклад 1111+0001;
- віднімання для A>B, A=B і A<B;
- позитивне та негативне знакове переповнення;
- граничні коди +7, −8, 0 і −1 для чотирьох бітів.

Повний перебір чотирибітного суматора віднімача містить 16·16·2=512 комбінацій A, B та SUB. Такий тест легко автоматизувати. Для кожного набору результат і прапорці порівнюються з еталонною математичною моделлю тієї самої розрядності.

11 Типові помилки

Помилка	Чому виникає неправильний результат
Початок зі старшого розряду	Cin старшого розряду ще не визначений молодшим розрядом
Втрата C0 під час SUB	Обчислюється A+$\overline{B}$ замість A+$\overline{B}$+1
Інверсія B без урахування всіх бітів	Порушується формування додаткового коду
Ототожнення Carry та Overflow	Ознаки належать до різних інтерпретацій чисел
Читання коду без signed або unsigned	Один код може позначати різні значення
Перевірка лише десяткового результату	Не виявляються помилки у проміжних переносах і прапорцях

12 Основні терміни і позначення

Позначення або термін	Значення
HA Half Adder	Напівсуматор для A і B
FA Full Adder	Повний суматор для A, B і Cin
S Sum	Біт суми поточного розряду
Cin Carry In	Вхідне перенесення
Cout Carry Out	Вихідне перенесення
RCA Ripple Carry Adder	Суматор із послідовним поширенням перенесення
CLA Carry Look Ahead Adder	Суматор із випереджувальним переносом
G Generate	Ознака генерування перенесення
P Propagate	Ознака поширення перенесення
SUB Subtract	Керувальний сигнал віднімання
Borrow	Позика у беззнаковому відніманні
Overflow V	Вихід знакового результату за допустимий діапазон
ALU або АЛП	Арифметико логічний пристрій
MUX Multiplexer	Мультиплексор для вибору одного з результатів

Підсумок

Напівсуматор додає два біти, а повний суматор враховує вхідне перенесення. Каскад повних суматорів утворює RCA для багаторозрядних слів. Віднімання реалізується тим самим трактом за формулою $A-B=A+\overline{B}+1$. Carry і Borrow описують беззнакову арифметику, а Overflow — допустимість знакового результату. CLA скорочує затримку завдяки паралельним рівнянням Generate і Propagate. Суматор віднімач разом із MUX та логікою прапорців утворює основу арифметичної частини АЛП.