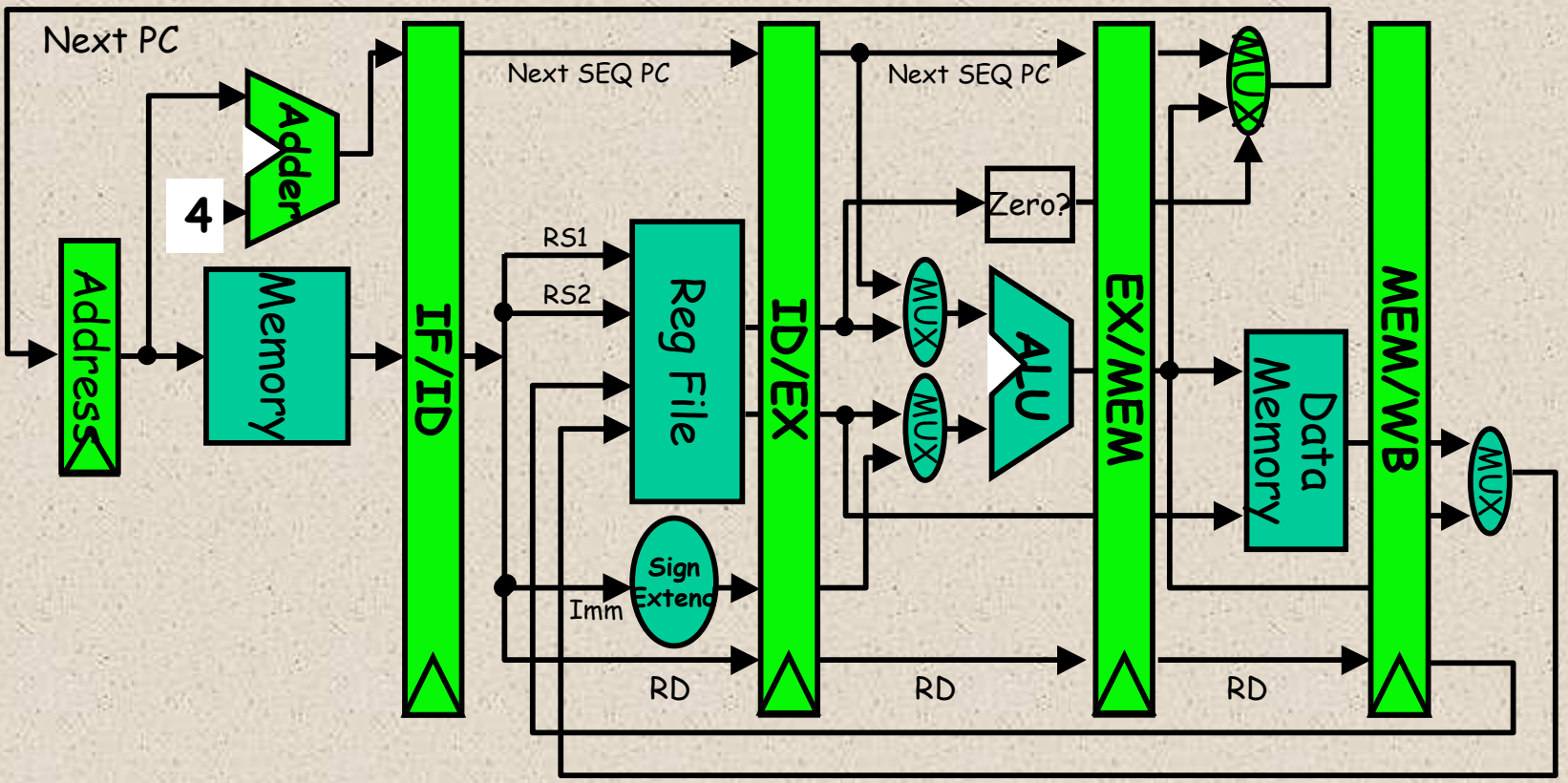
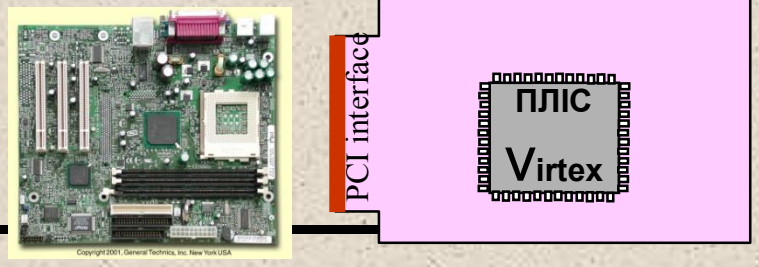
Проектування спеціалізованих обчислювачів



Синтез обчислювача

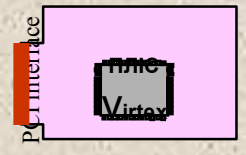
1) Вибирається **множина ресурсів** (процесори, обчислювальні блоки)
2) Складається **розклад** (schedule) виконання операцій алгоритму на ресурсах
3) Операції і змінні **призначаються** на ресурси, визначаються лінії зв'язку, формується структурна (функціональна) схема обчислювача

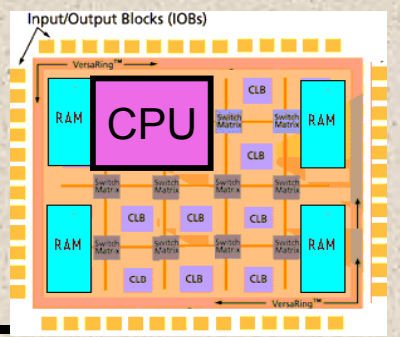


Синтез обчислювача

Оптимізація структури:

1) Формується множина ефективних структурних рішень
2) Структурні рішення сортуються по переважності за критерієм оптимальності
3) Найбільш переважна структура вибирається як оптимальна





Приклад синтезу блока для обчислення спецфункції

– розробити функціональну схему і VHDL-опис блока, що обчислює функцію:

$$\sin(X) = X - X^3/6 + X^5/120 = X - a*X^3 + b*X^5$$

Інтерфейс блока:

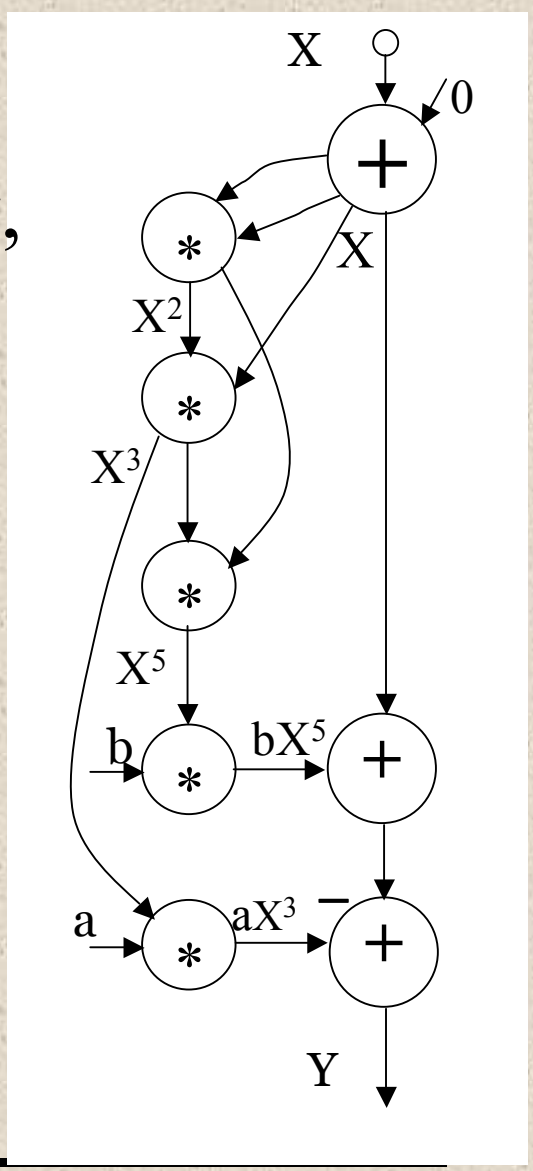
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.STD_LOGIC_arith.all;
entity X_Y_SIN is port(CLK : in STD_LOGIC;      --синхросерія
        RST : in STD_LOGIC; --сигнал начального встановлення
        START : in STD_LOGIC;       --сигнал запуску обчислень
        X : in STD_LOGIC_VECTOR(15 downto 0); --вхідне дане
        Y : out STD_LOGIC_VECTOR(15 downto 0); --вихідне дане
        RDY : out STD_LOGIC);      --сигнал готовності результату
end X_Y_SIN;
```

Приклад синтезу блока

$$Y = X - a*X^3+b*X^5$$

Початкові дані – граф алгоритму,
– критерій оптимальності =
мінімум апаратних витрат

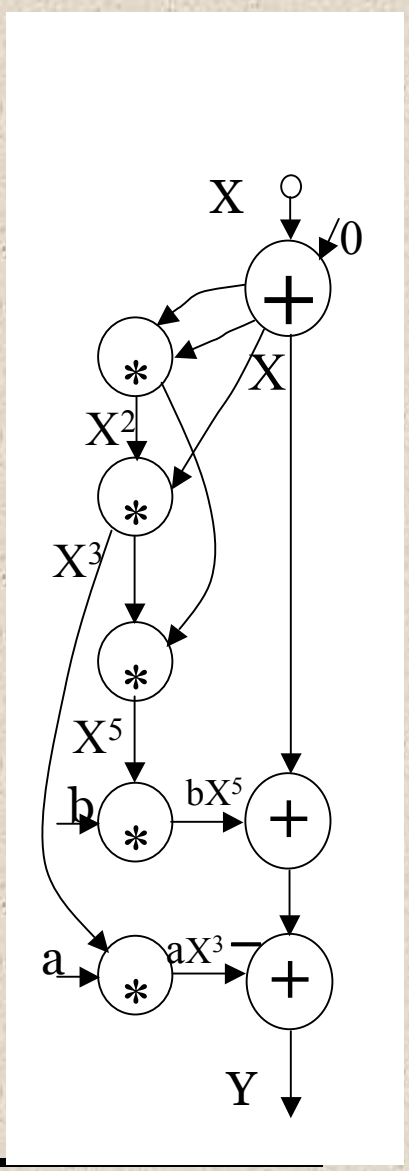
Ресурси: 1 блок множення,
1 суматор,
регістри результата Y
і проміжних результатів



Приклад синтезу блока

Розклад виконання

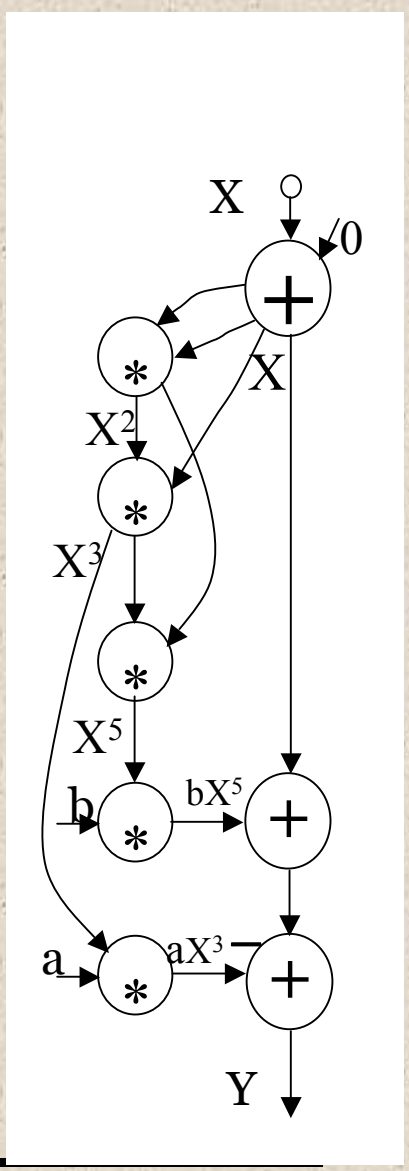
№ такт	Суматор S	Бл. множ. P	Рег X2	Рег X3	Рег Y
1	X				
2	X	$X^2 = X*X$			
3	X	$X^3 = X^2*X$	X^2		
4	X	$X^5 = X^2*X^3$		X^3	
5	X	$b*X^5$		X^3	
6	$X + b*X^5$	$a*X^3$			
7	$Y=X + b*X^5 - a*X^3$				
8					Y



Приклад синтезу блока

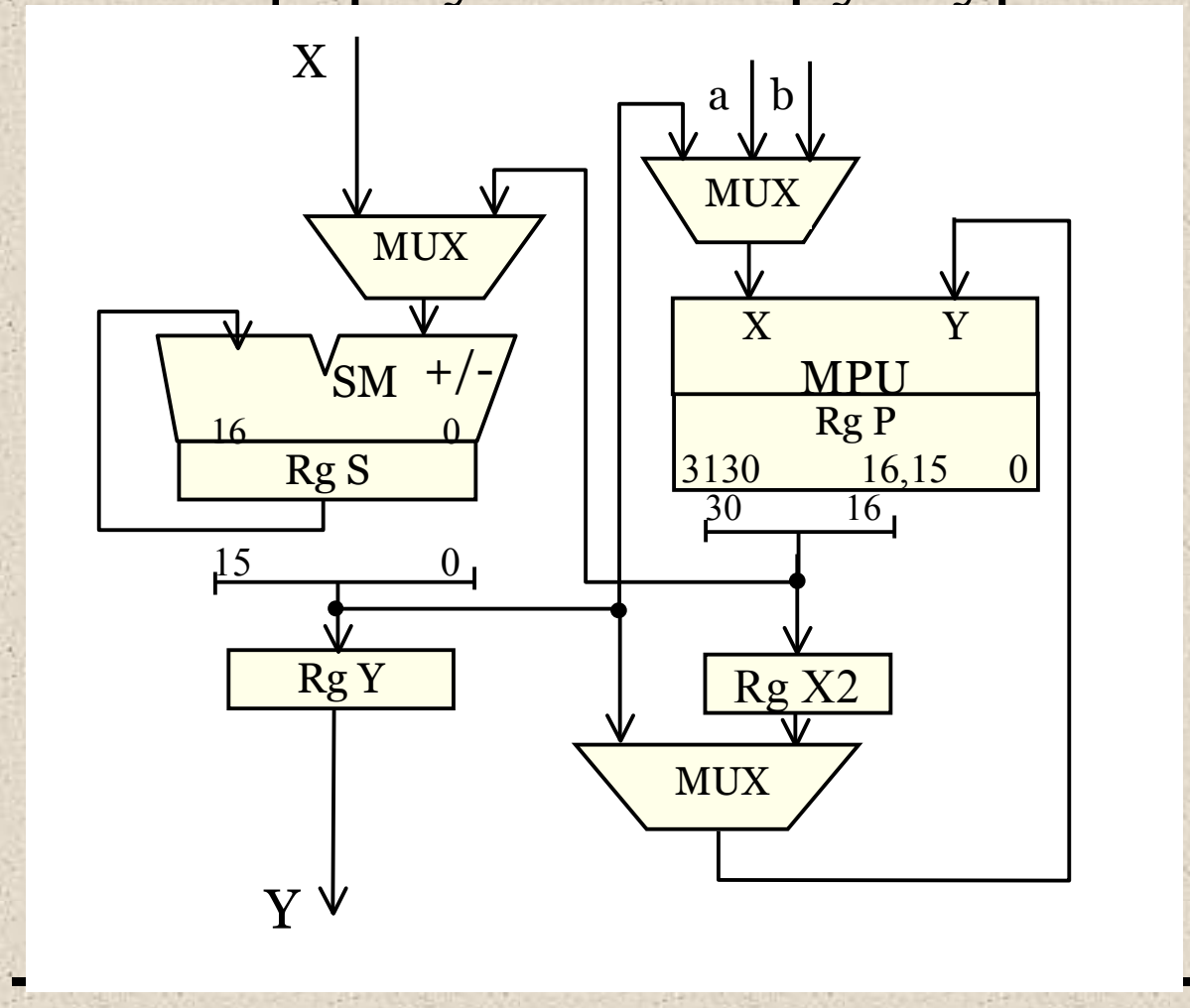
Оптимізація розкладу:
покращення завантаження регістрів

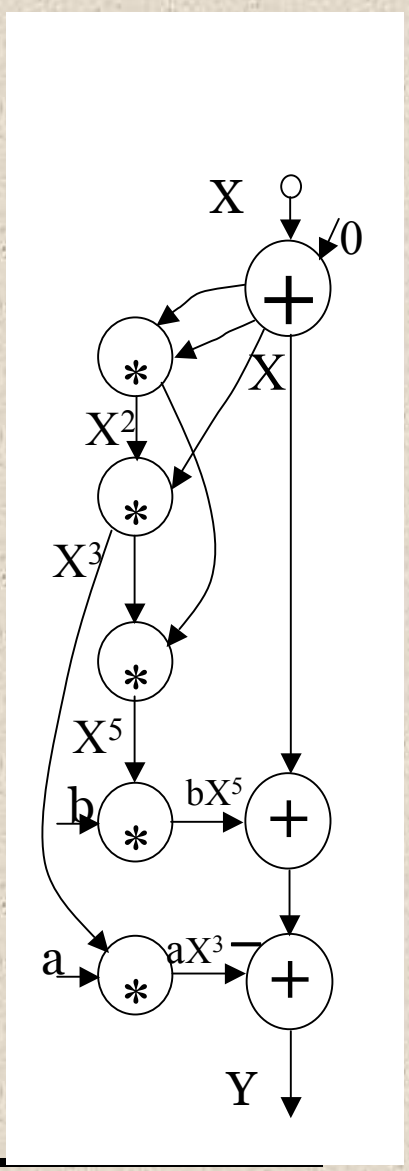
№ такт	Суматор S	Бл. множ. P	Рег X2	Рег X3	Рег Y
1	X				
2	X	$X^2 = X*X$			
3	X	$X^3 = X^2*X$	X^2		
4	X	$X^5 = X^2*X^3$	X^3	X^3	
5	X	$b*X^5$	X^3	X^3	
6	$X + b*X^5$	$a*X^3$			
7	$Y = X + b*X^5 - a*X^3$				
8					Y



Приклад синтезу блока

Призначення на ресурси:
формування структури





Приклад синтезу блока

Опис архітектури на VHDL

Завдання сигналів і констант

```
architecture X_Y_SIN of X_Y_SIN is
        constant a:SIGNED(15 downto 0):= --константа a
        SIGNED(conv_std_logic_vector(integer(1.0/6.0*2.0**15),16));
        constant b:SIGNED(15 downto 0):=  --константа b
        SIGNED(conv_std_logic_vector(integer(1.0/120.0*2.0**15),16));
        signal  s:SIGNED(16 downto 0);--суматор
        signal  p:SIGNED(31 downto 0);--блок множення
        signal  x2,x3:SIGNED(15 downto 0);--проміжний результат
        signal ct2:natural range 0 to 7;      --лічильник станів
   begin
```

Приклад синтезу блока

Опис архітектури на VHDL

Лічильник тактів

```
FSM:process(CLK,RST) begin
                if RST='1' then
                        ct2<=7; RDY<='0';
                elsif CLK='1' and CLK'event then
                        if ct2=7 then
                                RDY<='1';  -- результат готовий
                        end if;
                        if START='1' then
                                ct2<=0;     -- пуск обчислень
                                RDY<='0';
                        elsif ct2<7 then      -- лічильник рахує
                                ct2<=ct2+1;  -- до 7 і зупиняється
                        end if;
                end if;
        end process;
```

Опис архітектури на VHDL:
Власне обчислення за тактами

```
RALU:process(CLK,RST) begin
  if RST='1' then s<=(others=>'0'); p<=x"(others=>'0');
          x2<=(others=>'0');  Y<=(others=>'0');
  elsif CLK='1' and CLK'event then
        case ct2 is
        when 0=> s<= signed(SXT((X),17));          -- вхідне дане X
        when 1=> p<= s(15 downto 0)*s(15 downto 0); --X^2
        when 2=> p<= p(30 downto 15)*s(15 downto 0);--X^3
                  x2<=p(30 downto 15);                    -- X^2
        when 3=> p<= p(30 downto 15)*x2;          --X^5
                  x2<=p(30 downto 15);                -- X^3
        when 4=> p<= p(30 downto 15)*b;      --b*X^5
        when 5=> p<= x2*a;                   --a*X^3
                 s<= s + p(31 downto 15);     --X+b*X^5
        when 6=> s<= s - p(31 downto 15);     --  X+b*X^5-a*X^3
        when others=>Y<=std_logic_vector(s(15 downto 0));--результат
      end case;
  end if;
end process;   end X_Y_SIN;
```

Приклад синтезу блока

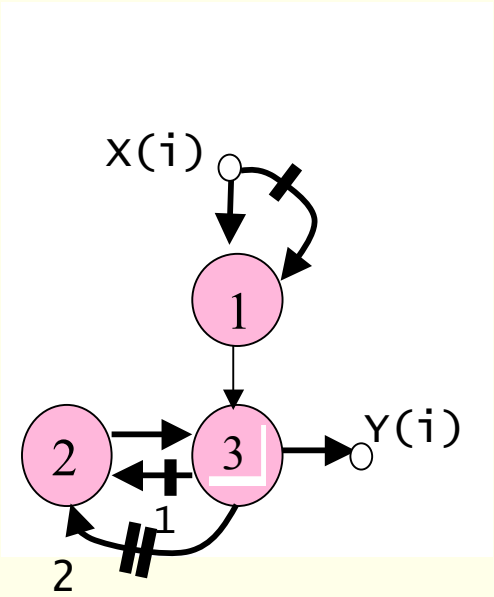
Аналіз результатів:

№ такт	Суматор S	Бл. множ. P	Рег X2	Рег X3	Рег Y
1	X				
2	X	$X^2 = X*X$			
3	X	$X^3 = X^2*X$	X^2		
4	X	$X^5 = X^2*X^3$	X^3		
5	X	$b*X^5$	X^3		
6	$X + b*X^5$	$a*X^3$			
7	$Y=X + b*X^5 - a*X^3$				
8					Y

Суматор – 1
Блок множення – 1
Регістрів – 4
Мультиплексорів - 3
Період виконання алгоритму – 8 тактів
Завантаження суматора – 25%
Завантаження блоку множення – 62,5%

Конвейерне виконання алгоритму

Граф синхронних потоків даних (ГСПД, SDF) – модель конвейерного обчислювача,

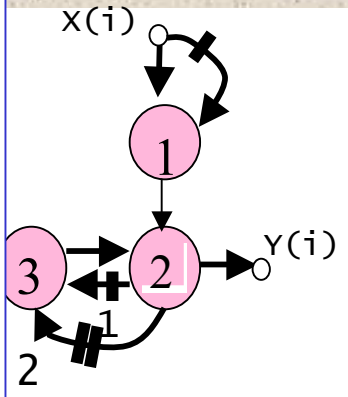


який виконує обчислення з періодом L=1 такт.

Якщо вхідні дані поступають потоком в кожному такті, то і результати видаються в кожному такті.

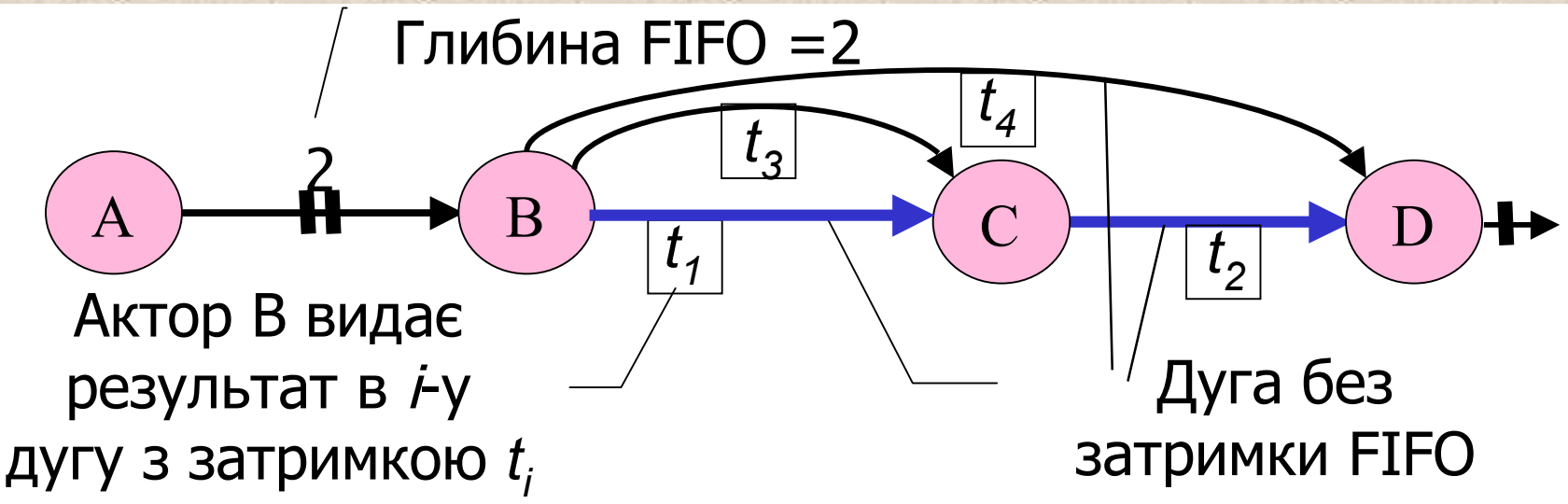
ГСПД відповідає модель обчислювача, стилем для синтезу описана на VHDL

```
ST0:process(CLK,RST) begin --процесс входной вершины
      if RST='1' then
         x_1<=0;                  -- начальное состояние X(i-1)
      elsif CLK='1' and CLK'event then
         x_1<=X; ;                -- регистр X(i-1)
      end if;
   end process;
   ST1: process(X,x_1) begin  --процесс вершины 1
      a<=x_1 + X;                 --операция в вершине 1
   end process;
   ST2: process(CLK,RST,a,b,y_0) begin--процесс вершины 2
      if RST='1' then
         y1_1<=0;      -- начальное состояние Y(i-1) 1-го FIFO
         y2_1<=0;      -- начальное состояние Y(i-1) 2-го FIFO
         y2_2<=0;      -- начальное состояние Y(i-2) 2-го FIFO
      elsif CLK='1' and CLK'event then
         y1_1<=y_0;               -- регистр Y(i-1) 1-го FIFO
         y2_1<=y_0;               -- регистр Y(i-1) 2-го FIFO
         y2_2<=y2_1;              -- регистр Y(i-2) 2-го FIFO
      end if;
       y_0<= a + b;               --операция в вершине 2
       Y<=y_0;                    --выходное данное
   end process;
   ST3: process(y1_1,y2_2) begin--процесс вершины 3
      b<= y1_1 + y2_2;              --операция в вершине 3
   end process;
```



Швидкодія ГСПД

Граф синхронних потоків даних (SDF):

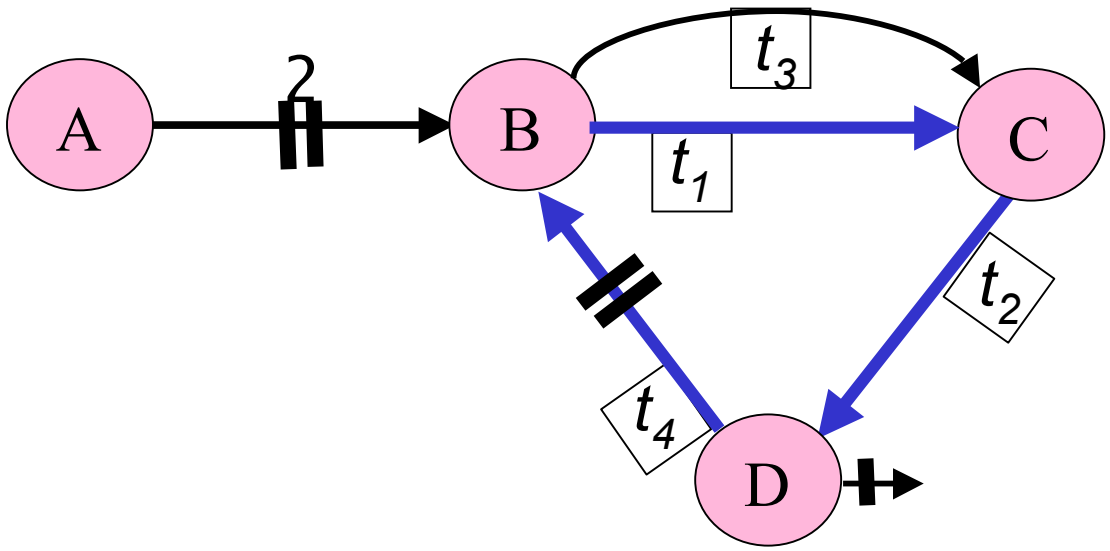


Довжина маршрута $T_{1Цj} = \sum t_i = t_1 + t_2$

Тривалість циклу $T_Ц = \max(T_{1Цj}) = \max((t_1 + t_2),(t_3 + t_2),t_4)$

Швидкодія ГСПД

Граф синхронних потоків даних (SDF):

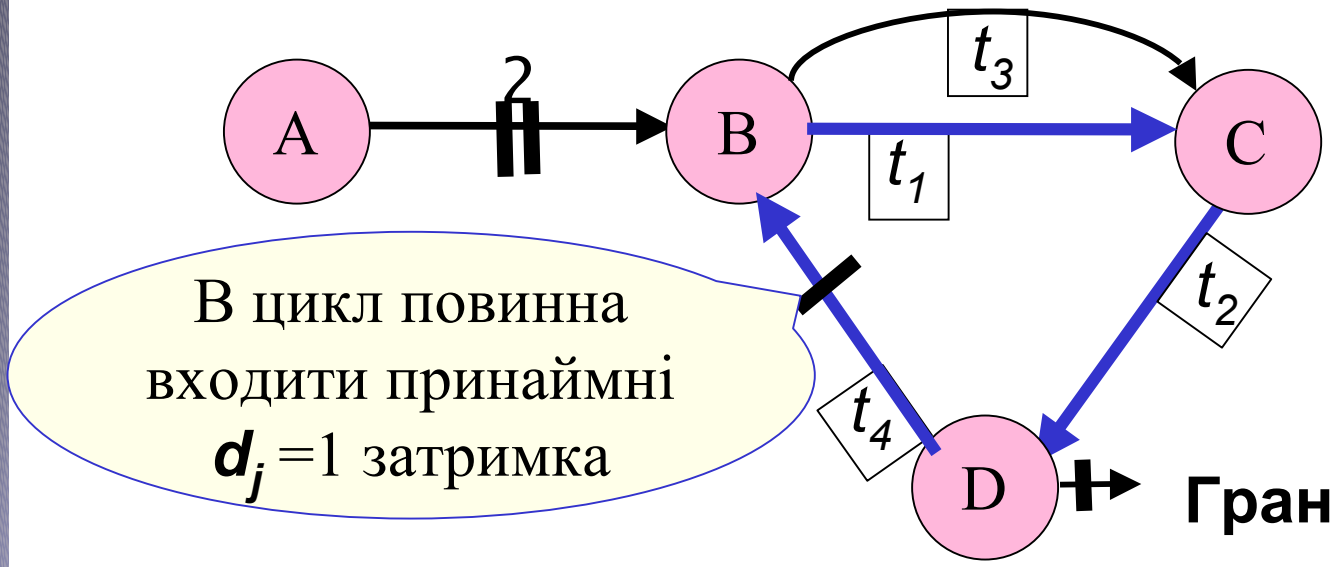


Границя *i* –го циклу $T_{Цj} = (1/d_j)\Sigma t_i = (t_1 + t_2 + t_4)/2$

- теоретична границя тривалості тактового інтервалу

Швидкодія ГСПД

Граф синхронних потоків даних (SDF):



Границя $T_{Цj} = (1/d_j)\Sigma t_i$

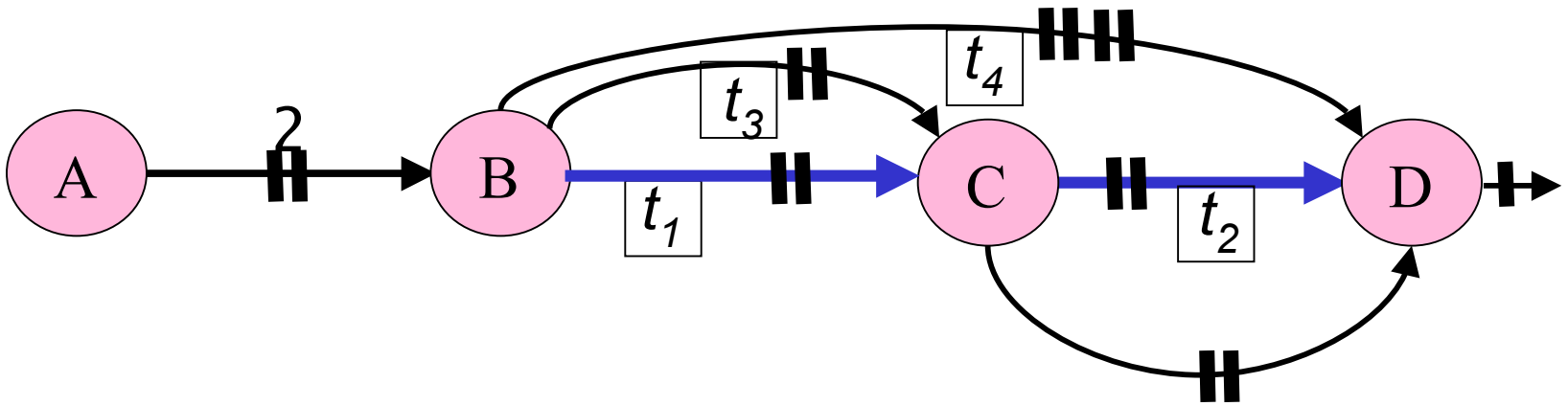
Границя ітерації $T_{ЦE} = \max(T_{Цj})$

Критичний цикл $T_{Цi} = T_{ЦE.}$

ГСПД з максимальною швидкодією повинен мати якнайбільшу кількість затримок FIFO у всіх своїх циклах

Швидкодія ГСПД

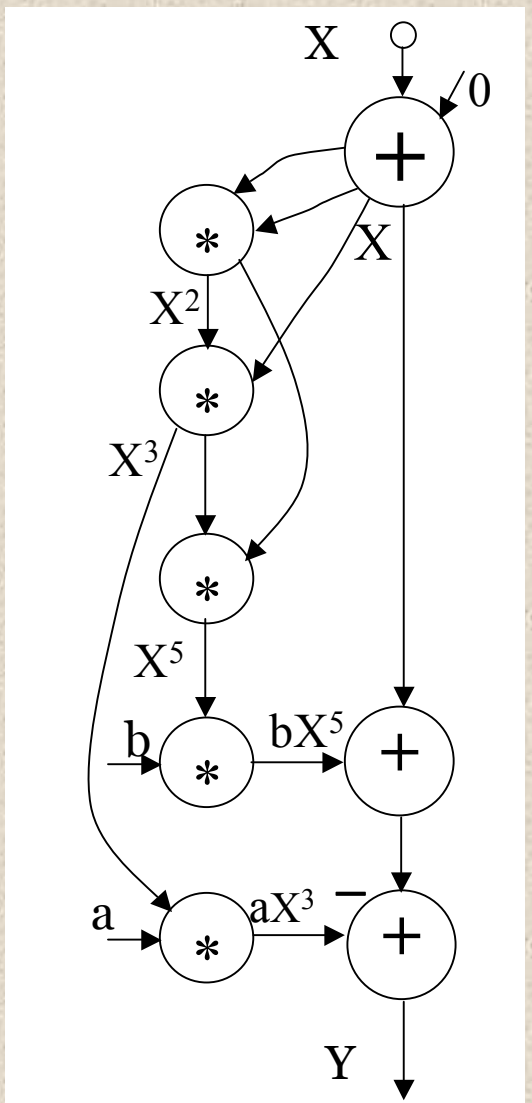
Граф синхронних потоків даних (SDF):



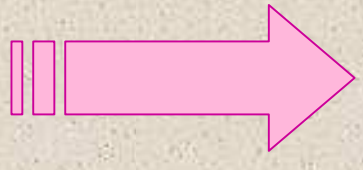
Якщо алгоритм – ациклічний, то в дуги його ГСПД можна додати як завгодно велику кількість регістрів затримки і для нього границя ітерації може бути **самою малою**.

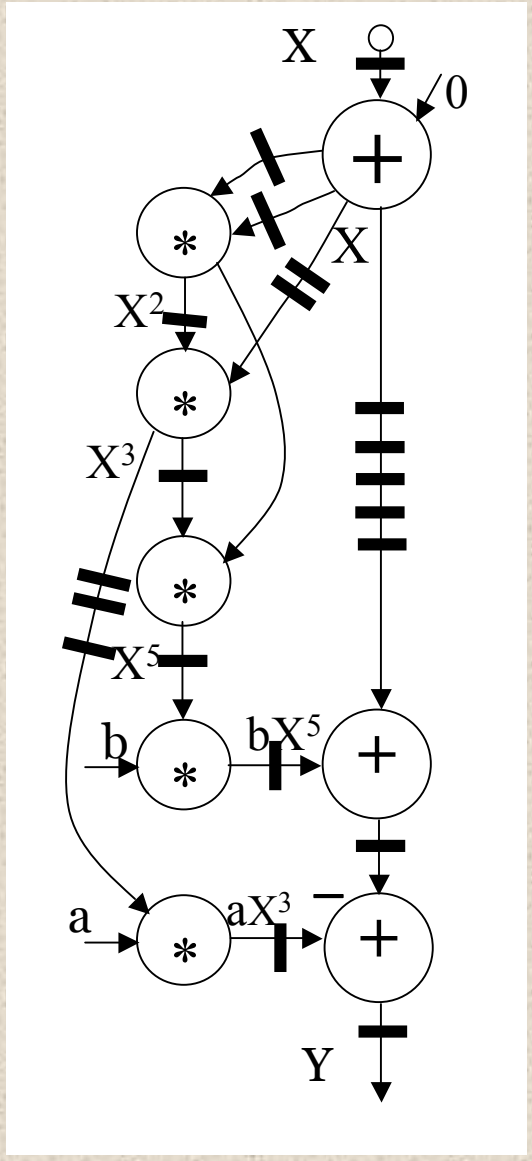
Урівноваження ациклічного графу – додавання до його дуг такої кількості затримок, щоб кількість затримок у будь-якому маршруті між двома вершинами була однаковою.

Перетворення графа алгоритму в ациклічний ГСПД



Урівноваження

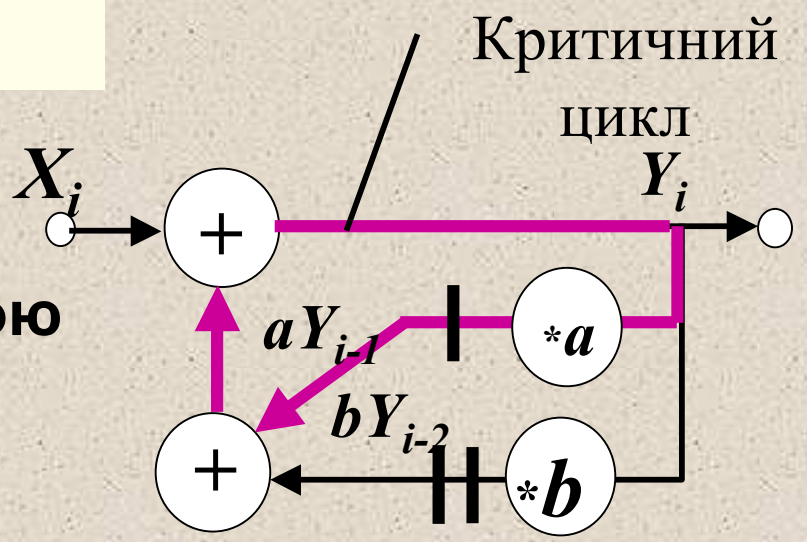




Виконує обробку потоків з періодом L=1 такт

Оптимізація графів алгоритму

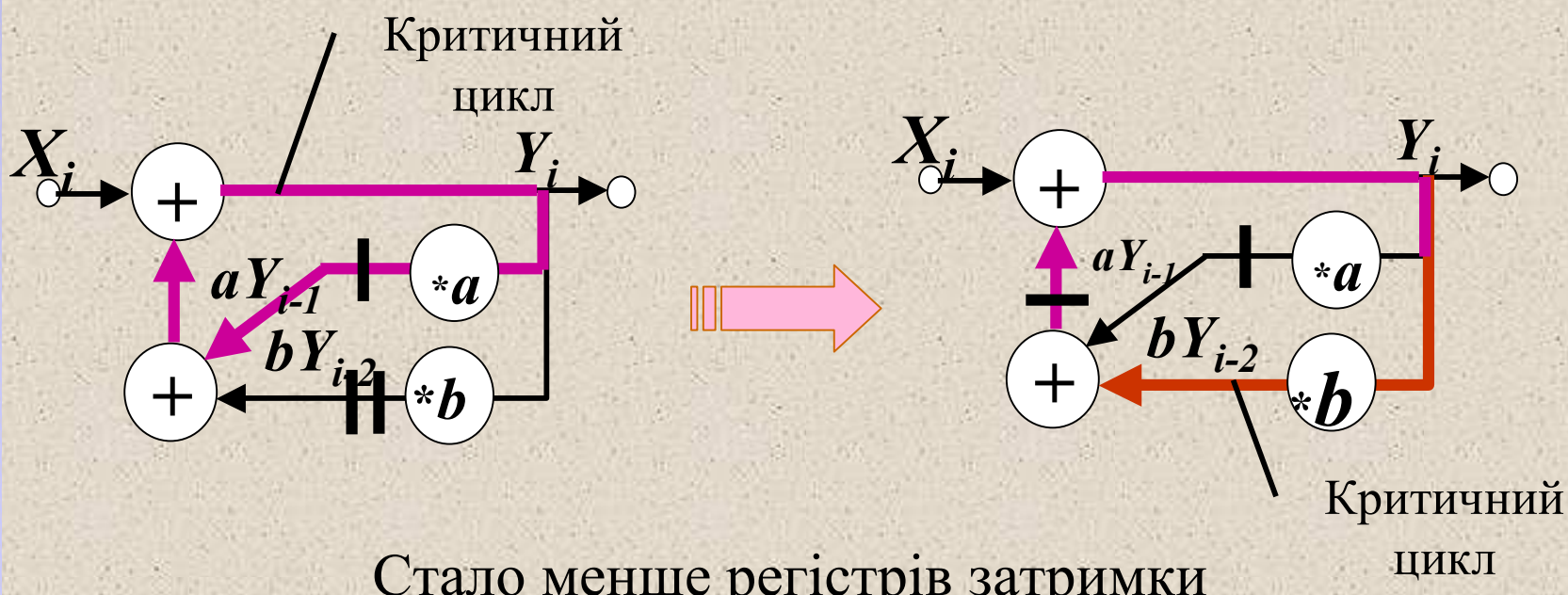
Мета оптимізації – наблизити
тривалість циклу $T_Ц = \max(T_{1Цj})$
до границі циклу $T_{Цj} = (1/d_j)\Sigma t_i$,
мінімізувати
границю ітерації $T_{ЦE} = \max(T_{Цj})$
і критичний цикл $T_Ц = T_{ЦE.}$



Приклад фільтра з передаточною функцією $H_{(Z)} = 1/(1-aZ^{-1}-bZ^{-2})$,
і різнецевим рівнянням
$Y_i = X_i + aY_{i-1} + bY_{i-2}$
Параметри: $T_Ц = t_M + 2t_+$ **і** $T_{ЦE} = T_Ц$

Оптимізація графів алгоритму

Ресинхронизація (retiming) –
Затримки в дугах ГСПД переставляються таким чином, щоб результати обчислень залишались такими самими. Корректне перетворення –
Сума затримок в кожному циклі – та сама, як і до перетворення

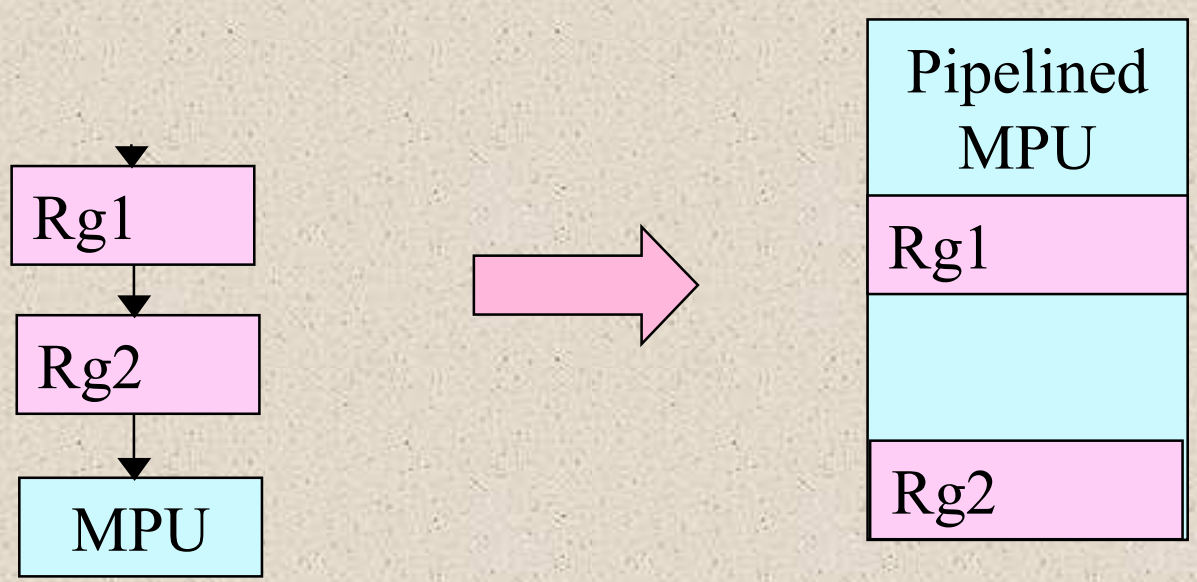


Стало менше регістрів затримки

Оптимізація графів алгоритму

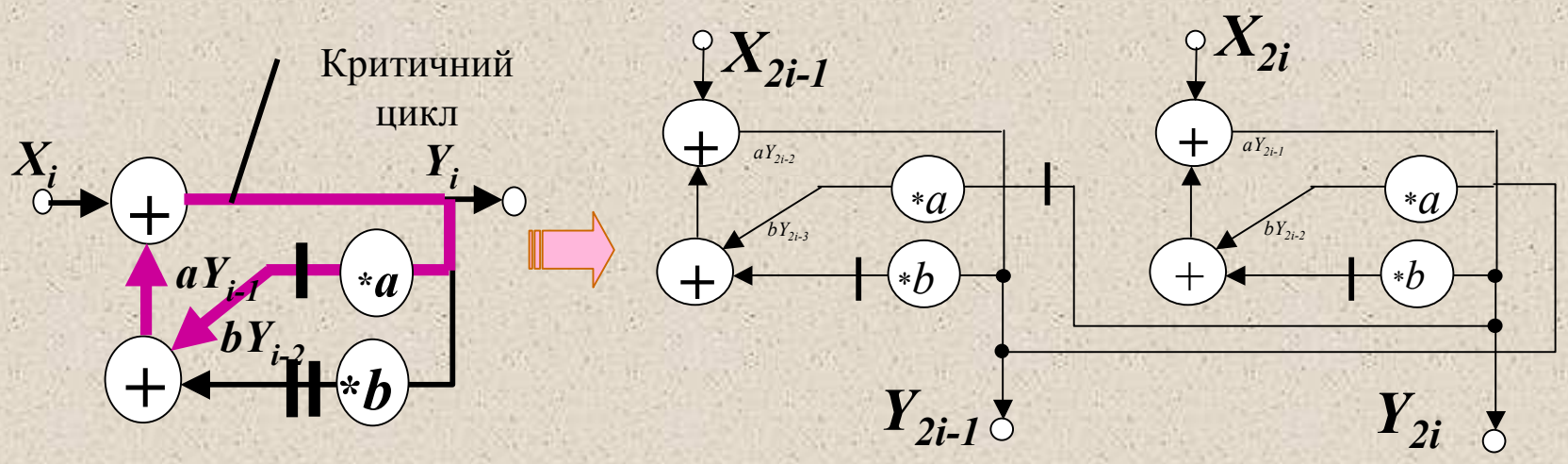
Ресинхронізація (retiming) – в VHDL

```
attribute register_balancing: string;
attribute register_balancing of Rg1,Rg2: is “yes”;
```



Оптимізація графів алгоритму

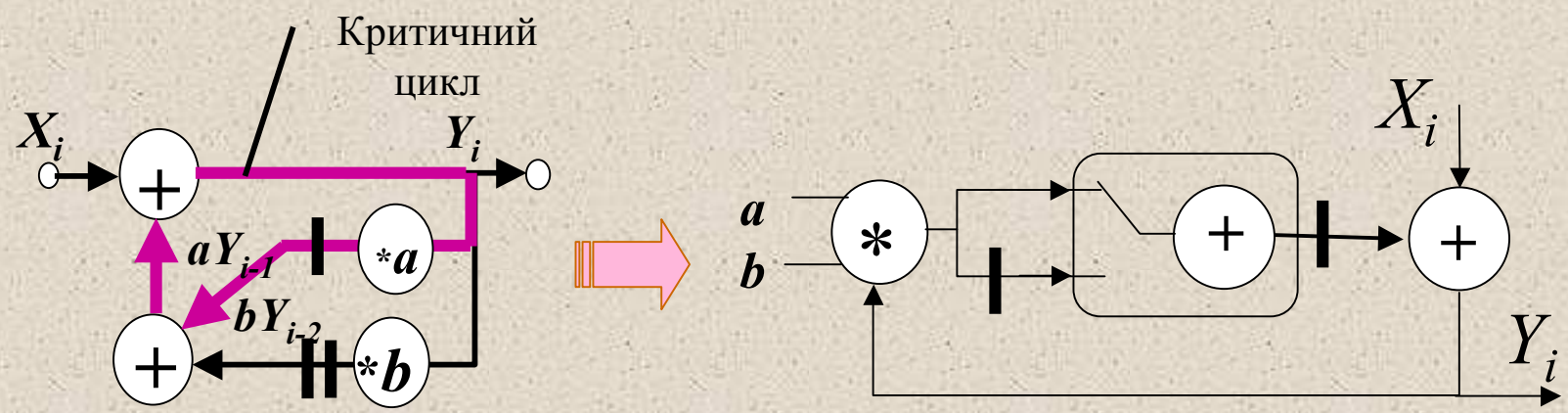
Розкручування ГСПД (unfolding)



В розкрученому ГСПД $T_Ц = 2t_M + 4t_+$, *але вдвічі більше операцій виконується паралельно – для парних і непарних даних*

Оптимізація графів алгоритму

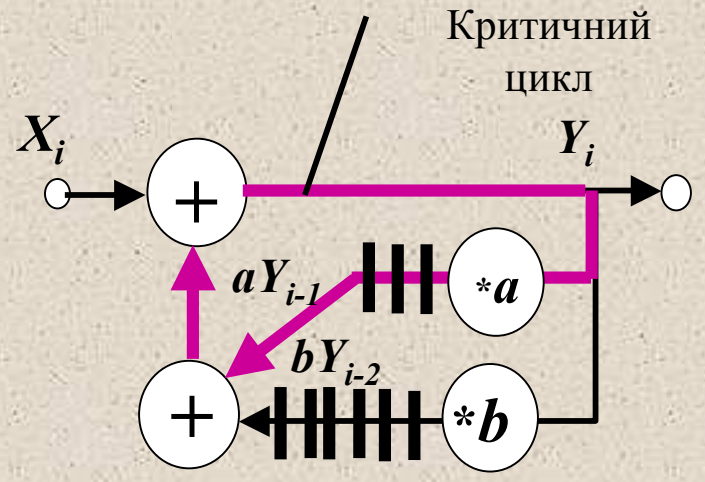
Згручування ГСПД (folding)



В згрученому ГСПД $T_Ц = t_M + 2t_+ + t_{MX}$, *але період виконання алгоритму L=2 і майже в L разів менші апаратні витрати*

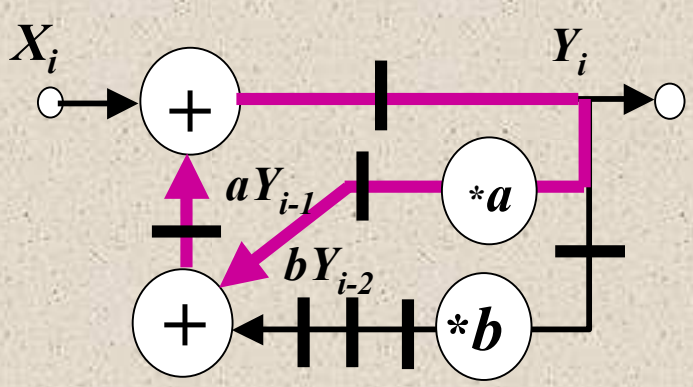
Оптимізація графів алгоритму

Згручування ГСПД (folding)



ГСПД після згручування 3-х графів

$T_Ц = t_M + 2t_+$ і $T_{ЦЕ} = T_Ц/3$



ГСПД після згручування і ресинхронізації

$T_Ц = t_M$ і $T_{ЦЕ} = (t_M + 2t_+)/3$

Періодичні алгоритми

Класифікація графів потоків даних