Конвейер

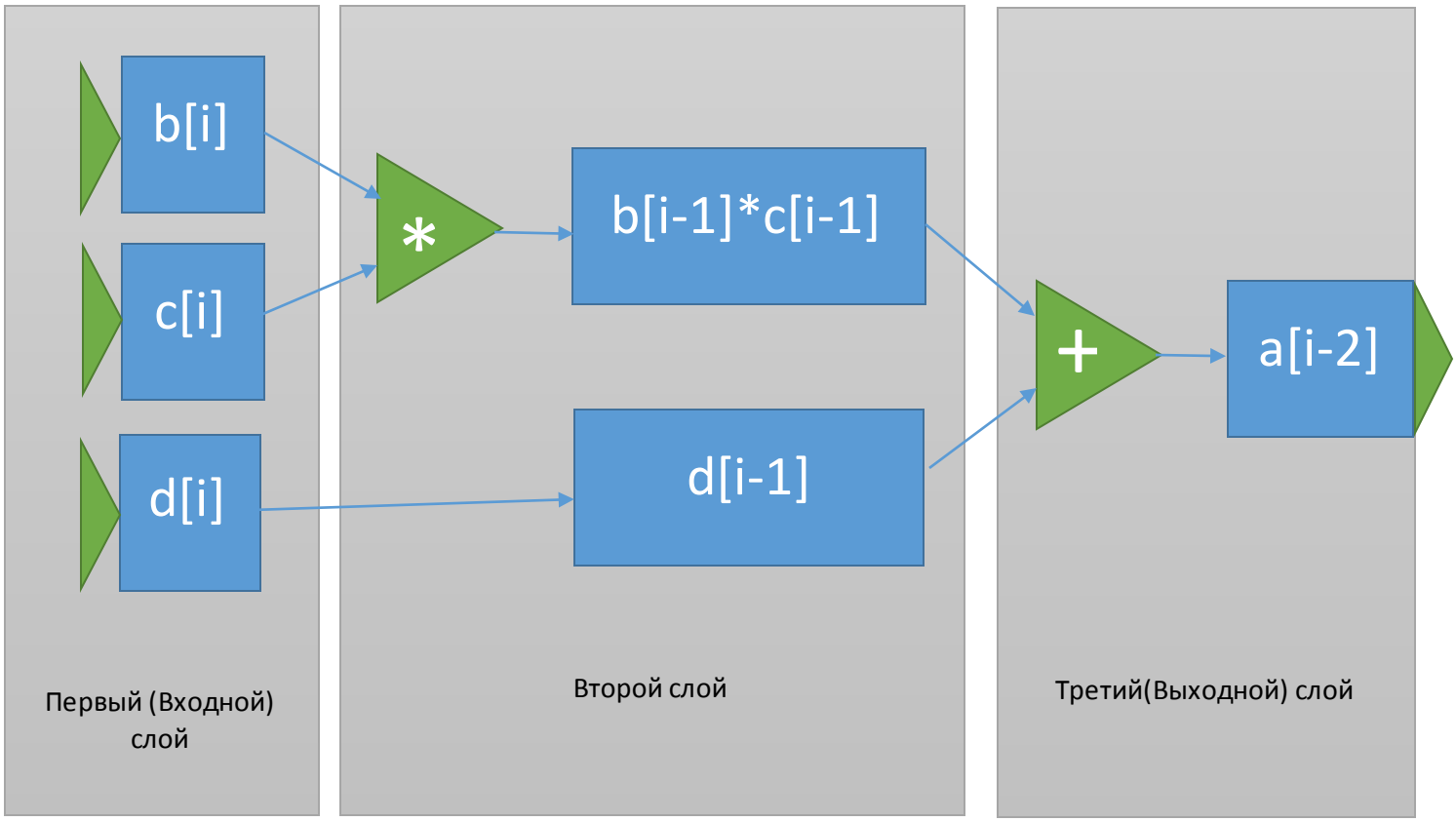
Конвейер – это способ распараллеливания вычислений, заключающийся в том, что каждая операция работает одновременно с другими, но над другой порцией данных, и завершив вычисления каждый элементарный вычислитель в конвейере передаёт результат дальше по конвейеру и, сразу же, берёт новую порцию данных с предыдущего этапа конвейера.

Пример конвейера.

Код на C++:

```
for(int i = 0; i < N; i++) {
        a[i] = b[i]*c[i] + d[i];
}
```

Граф вычислений:



Конвейер состоит из нескольких слоёв, каждый из которых имеет внутри себя вычислитель, на логических функциях, и регистр, в который сохраняется результат вычислений этого слоя.

```
module pipeline
(
        // Служебные сигналы
        input clock,
        input reset,
        // Входные значения
        input  [3:0] b,
        input  [3:0] c,
        input  [3:0] d,
        // флаги валидности для входных значений
        input        b_vld,
        input        c_vld,
        input        d_vld,
        // Выходное значение
        output [7:0] a
        // Флаг валидности для выходного значения
        output       a_vld
);


        //----------  Первый слой----------

        // Буферы для хранения входные значений
        reg  [3:0] b_reg;
        reg  [3:0] c_reg;
        reg  [3:0] d_reg;
        // Буферы для хранения флагов валидности буферов b_reg, c_reg, d_reg, соответственно.
        reg      b_reg_vld;
        reg      c_reg_vld;
        reg      d_reg_vld;

        // Сохранение входных данных в буфер
        always @(posedge clock) begin
                if (!reset) begin
                        b_reg <= 0;
                        c_reg <= 0;
                        d_reg <= 0;
                end else begin
                        if (b_vld) b_reg <= b;
                        if (c_vld) c_reg <= c;
                        if (d_vld) d_reg <= d;
                end
        end

        // Сохранение флагов валидности входных данных
        always @(posedge clock) begin
                if (!reset) begin
                        b_reg_vld <= 0;
                        c_reg_vld <= 0;
                        d_reg_vld <= 0;
                end else begin
                        b_reg_vld <= b_vld;
                        c_reg_vld <= c_vld;
                        d_reg_vld <= d_vld;
                end
        end
```

```
//------------------  Второй слой-------------------------

// Объявляем линию, которая будет передавать результат умножения b_reg на c_reg
wire [7:0] b_mul_c = b_reg * c_reg;
// Флаг валидности операции умножения. Равен 1 только если оба множителя валидны.
wire       b_mul_c_vld = b_reg_vld && c_reg_vld;
// Регистр для хранения результата умножения и
reg  [7:0] b_mul_c_reg;
// Буферный регистр для "d", что бы подержать его где-то, пока умножается b_reg на c_reg
reg  [3:0] d_buf_reg;
// Буферы для хранения флагов валидности
reg        b_mul_c_reg_vld;
reg        d_buf_reg_vld;

// сохранение умножения "b*c" и "d" в буффер.
always @(posedge clock) begin
        if (!reset) begin
                b_mul_c_reg <= 0;
                d_buf_reg <= 0;
        end else begin
                if (b_mul_c_vld) b_mul_c_reg <= b_mul_c;
                if (d_reg_vld) d_buf_reg <= d_reg;
        end
end

// Сохранение флагов валидности
always @(posedge clock) begin
        if (!reset) begin
                b_mul_c_reg_vld <= 0;
                d_buf_reg_vld <= 0;
        end else begin
                b_mul_c_reg_vld <= b_mul_c_vld;
                d_buf_reg_vld <= d_reg_vld;
        end
end
```

```
        //------------------  Третий слой----------------------

        // Объявляем линию, которая будет передавать результат умножения b_reg на c_reg
        wire [7:0] a_int = b_mul_c_reg + d_buf_reg;
        // Флаг валидности операции сложения. Равен 1 только если оба слагаемых валидны.
        wire       a_int_vld = b_mul_c_reg_vld && d_buf_reg_vld;
        // Регистр для хранения "a"
        reg [7:0] a_reg;
        // Буферы для хранения флагов валидности
        reg       a_reg_vld;

        // сохранение сложения "b*c + d"
        always @(posedge clock) begin
                if (!reset) begin
                        a_reg <= 0;
                end else begin
                        if (a_int_vld) a_reg <= a_int;
                end
        end

        // Сохранение флагов валидности
        always @(posedge clock) begin
                if (!reset) begin
                        a_reg_vld <= 0;
                end else begin
                        a_reg_vld <= a_int_vld;
                end
        end


        // Вывод результата в порт
        assign a = a_reg;
        assign a_vld = a_reg_vld;
end module
```

На VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_signed.all;

-- Объявление интерфейса модуля

entity pipeline is
	-- Объявление порта модуля. Входные и выходные линии, их имена и типы.
	port
	(
	-- Служебные сигналы
	clock	: in std_logic,
	reset	: in std_logic,
	-- Входные значения
	b		: in std_logic_vector(3 downto 0);
	c		: in std_logic_vector(3 downto 0);
	d		: in std_logic_vector(3 downto 0);
	-- флаги валидности для входных значений
	b_vld	: in std_logic;
	c_vld	: in std_logic;
	d_vld	: in std_logic;
	-- Выходное значение
	a		: out std_logic_vector(7 downto 0);
	-- Флаг валидности для выходного значения
	a_vld	: out std_logic
	);

end entity;

-- Определение архитектуры модуля
architecture rtl_pipeline of pipeline is
	-- Буферы для хранения входные значений
	signal b_reg: std_logic_vector(3 downto 0);
	signal c_reg: std_logic_vector(3 downto 0);
	signal d_reg: std_logic_vector(3 downto 0);
	-- Буферы для хранения флагов валидности буферов b_reg, c_reg, d_reg, соответственно.
	signal b_reg_vld		: std_logic;
	signal c_reg_vld		: std_logic;
	signal d_reg_vld		: std_logic;



	-- Объявляем линию, которая будет передавать результат умножения b_reg на c_reg
	signal b_mul_c : std_logic_vector(7 downto 0);
	-- Флаг валидности операции умножения. Равен 1 только если оба множителя валидны.
	signal b_mul_c_vld	: std_logic;
	-- Регистр для хранения результата умножения и
	signal b_mul_c_reg;
	-- Буферный регистр для "d", что бы подержать его где-то, пока умножается b_reg на c_reg
	signal d_buf_reg;
	-- Буферы для хранения флагов валидности
	signal b_mul_c_reg_vld;
	signal d_buf_reg_vld;

	-- Регистр для хранения "a"
	signal a_reg : std_logic_vector(7 downto 0);
	-- Буферы для хранения флагов валидности
	signal a_reg_vld : std_logic;
```

```
        -- Объявляем линию, которая будет передавать результат умножения b_reg на c_reg
        signal a_int : std_logic_vector(7 downto 0);
        -- Флаг валидности операции сложения. Равен 1 только если оба слагаемых валидны.
        signal a_int_vld : std_logic;
begin

        --Сохранение входных данных в буфер
        process(clock, reset)
        begin
                if (clock'event and clock = '1') then
                        if (reset = '0') then
                                b_reg <= x"0";
                                c_reg <= x"0";
                                d_reg <= x"0";
                        elsif
                                if (b_vld = '1') then
                                        b_reg <= b;
                                end if;
                                if (c_vld = '1') then
                                        c_reg <= c;
                                end if;
                                if (d_vld = '1') then
                                        d_reg <= d;
                                end if;
                        end if;
                end if;
        end process;


        -- Сохранение флагов валидности входных данных
        process(clock, reset) begin
                if (clock'event and clock = '1') then
                        if (reset = '0') then
                                b_reg_vld <= '0';
                                c_reg_vld <= '0';
                                d_reg_vld <= '0';
                        else
                                b_reg_vld <= b_vld;
                                c_reg_vld <= c_vld;
                                d_reg_vld <= d_vld;
                        end if;
                end if;
        end;


        ---------- Второй слой----------

        -- Объявляем линию, которая будет передавать результат умножения b_reg на c_reg
        b_mul_c <= b_reg * c_reg;
        -- Флаг валидности операции умножения. Равен 1 только если оба множителя валидны.
        b_mul_c_vld = b_reg_vld && c_reg_vld;

        -- сохранение умножения "b*c" и "d" в буффер.
        process (clock, reset) begin
                if (clock'event and clock = '1') then
                        if (reset = '0') then
                                b_mul_c_reg <= x"0";
                                d_buf_reg <= x"0";
                        else
                                if (b_mul_c_vld = '1') then
                                        b_mul_c_reg <= b_mul_c;
                                end if;
```

```
                        if (d_reg_vld  = '1') then
                                d_buf_reg  <= d_reg;
                        end if;
                end if;
        end if;
end;

-- Сохранение флагов валидности
process (clock, reset) begin
        if (clock'event and clock = '1') then
                if (reset = '0') then
                        b_mul_c_reg_vld  <= x"0";
                        d_buf_reg_vld  <= x"0";
                else
                        b_mul_c_reg_vld  <= b_mul_c_vld;
                        d_buf_reg_vld  <= d_reg_vld;
                end if;
        end if;
end;



---------- Третий слой----------

-- Объявляем линию, которая будет передавать результат умножения b_reg на c_reg
a_int <= b_mul_c_reg  + d_buf_reg;
-- Флаг валидности операции сложения. Равен 1 только если оба слагаемых валидны.
a_int_vld  <= b_mul_c_reg_vld  and d_buf_reg_vld;

-- сохранение сложения "b*c + d"
process (clock, reset) begin
        if (clock'event and clock = '1') then
                if (reset = '0') begin
                        a_reg  <= x"00";
                else
                        if (a_int_vld  = '1') then
                                a_reg  <= a_int;
                        end if;
                end if;
        end if;
end;

-- Сохранение флагов валидности
process (clock, reset) begin
        if (clock'event and clock = '1') then
                if (reset = '0') begin
                        a_reg_vld  <= 0;
                else
                        a_reg_vld  <= a_int_vld;
                end if;
        end if;
end;


-- Вывод результата  в порт
a <= a_reg;
a_vld  <= a_reg_vld;

end rtl_pipeline;
```