**ЭУМКД https://drive.google.com/file/d/17BoZBqaSHMQ8nfXxtoUEADgm74sCfBRG/view?usp=sharing**

**Лабораторная работа №1**

Используя Quartus Prime Lite Edition, реализовать схему АЛУ, реализующего операции сложения и вычитания знаковых целых чисел в дополнительном коде.

Входы: 5-ти разрядные A[4..0] и B[4..0] - исходные данные и одноразрядный Op, который определяет выполняемую операцию.

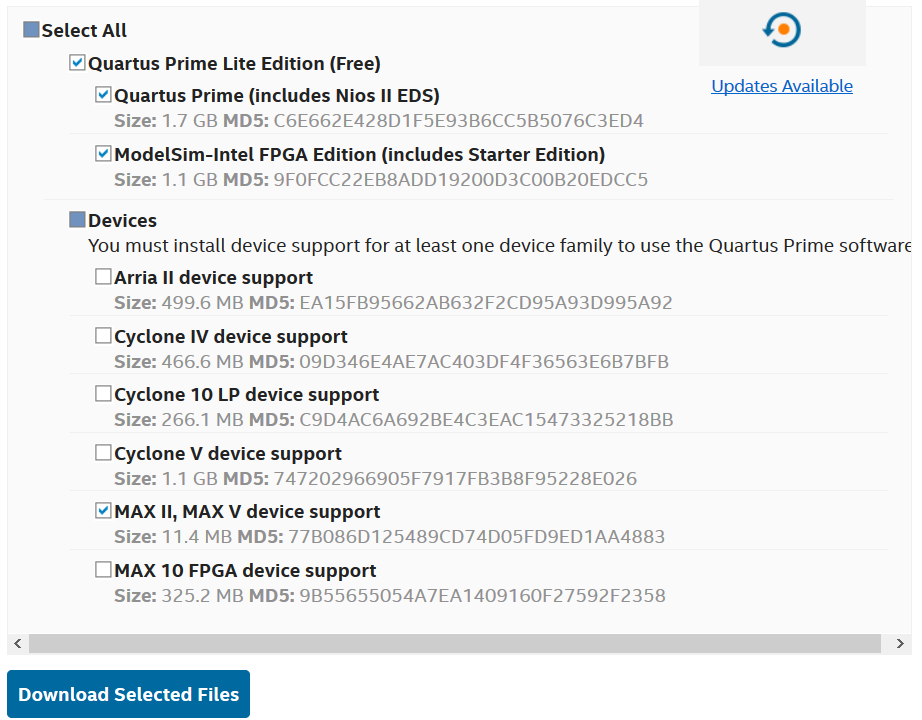
Выходы: 5-ти разрядный Q[4..0] и одноразрядный O, показывающий знаковое переполнение.

Для защиты ЛР предоставляется VWF, содержащий результаты для всех возможных входных сигналов.

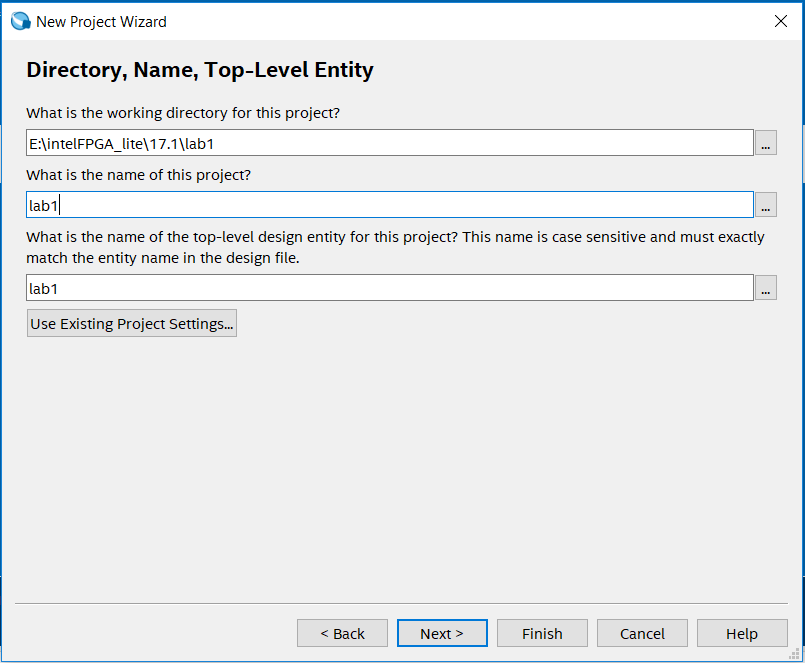
***Мини-Гайд по Quartus***

**Скачивание**

http://dl.altera.com/?edition=lite

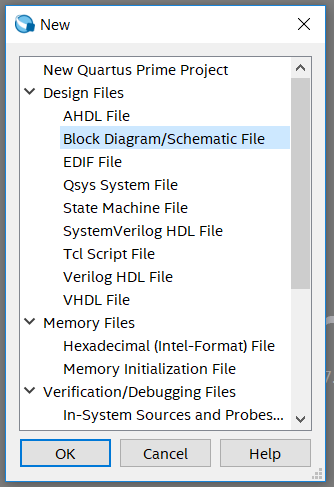


**Создание проекта**



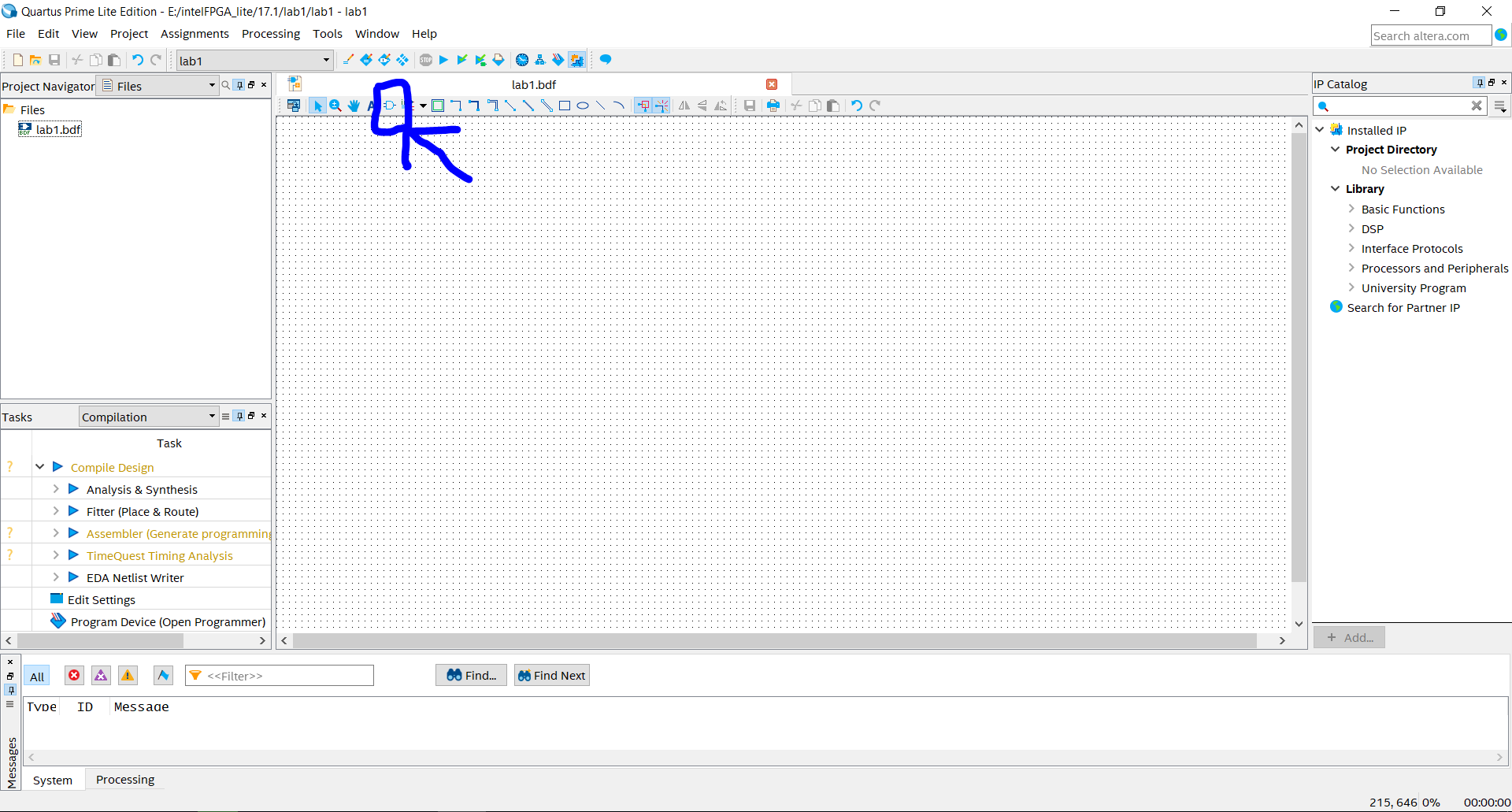
Тут можно просто нажать Finish, если вы всё установили верно, то всё будет хорошо.

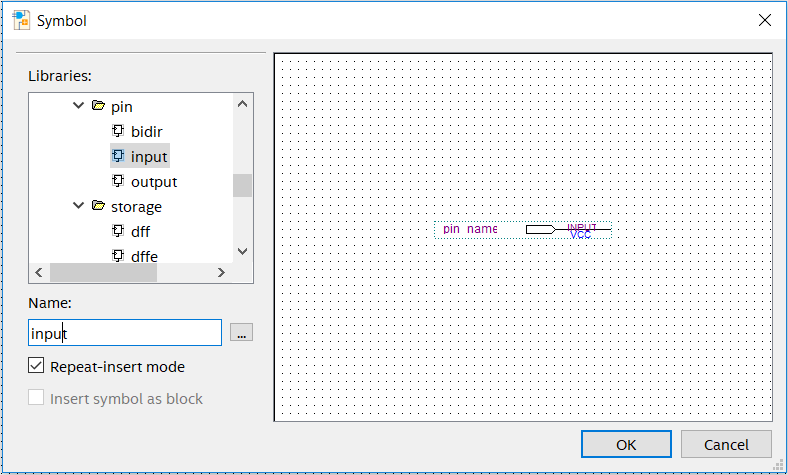
**Создание блока**



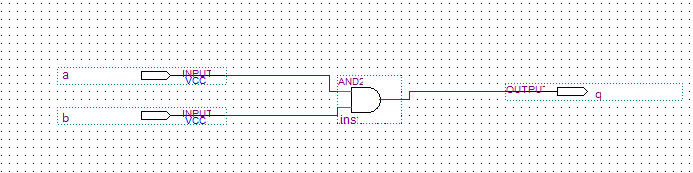
ВАЖНО: у вас должен существовать блочный файл, совпадающий с именем проекта, который будет вашим результирующим устройством.

**Добавление элементов**

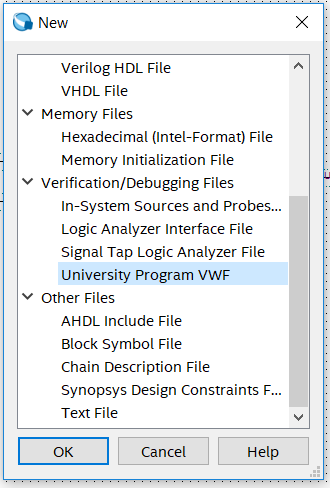




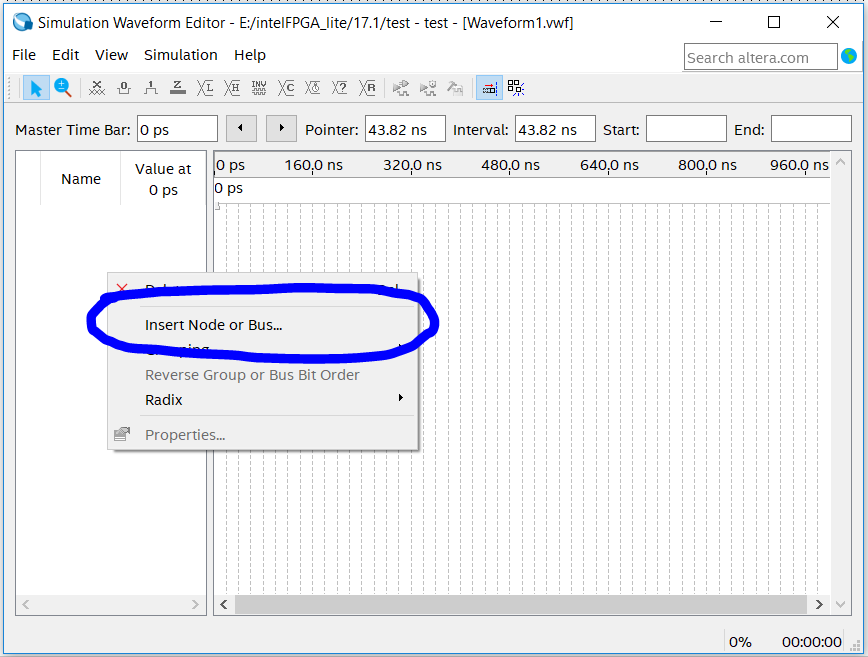
**Пример простой схемы**

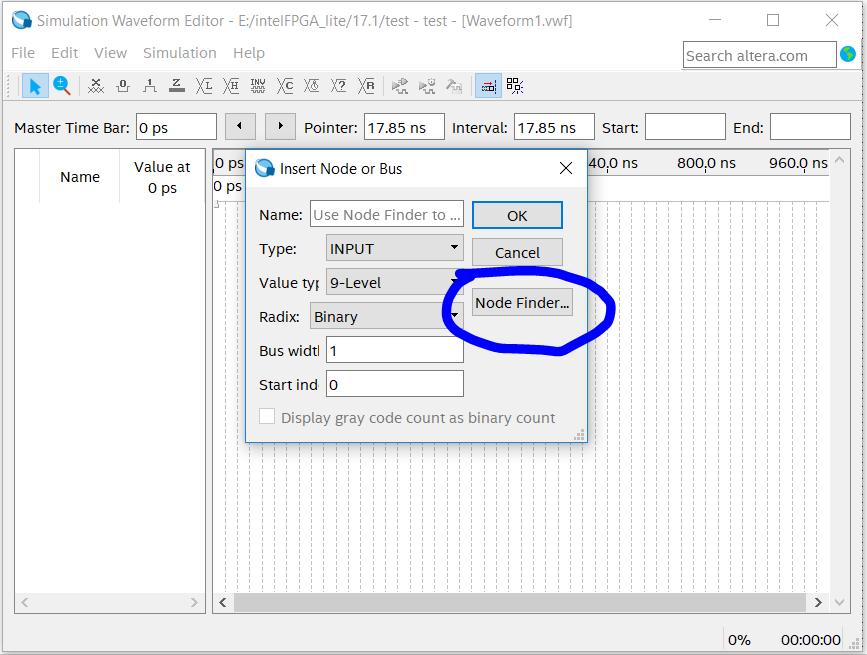


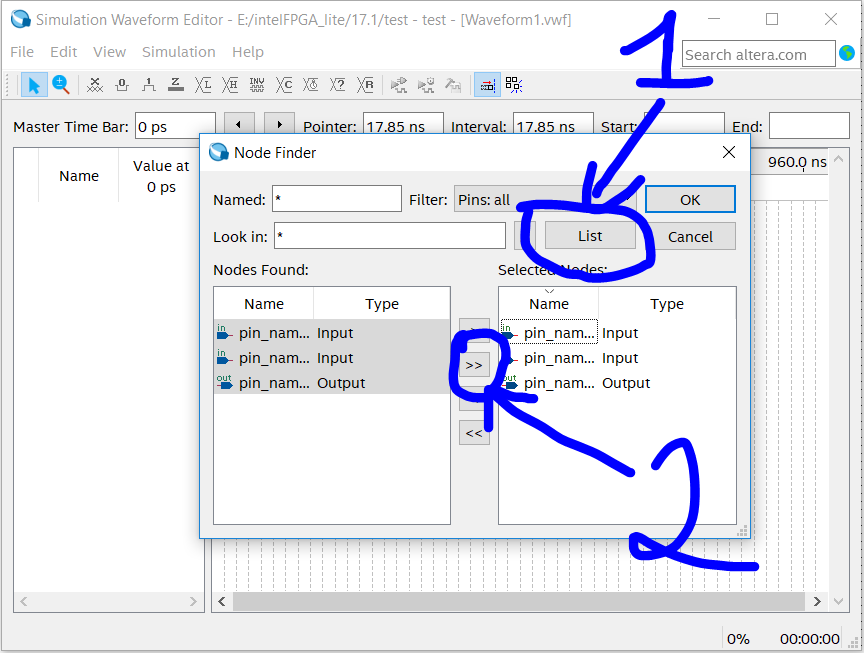
**Создание тестирующего файла**



**Добавление входов/выходов**



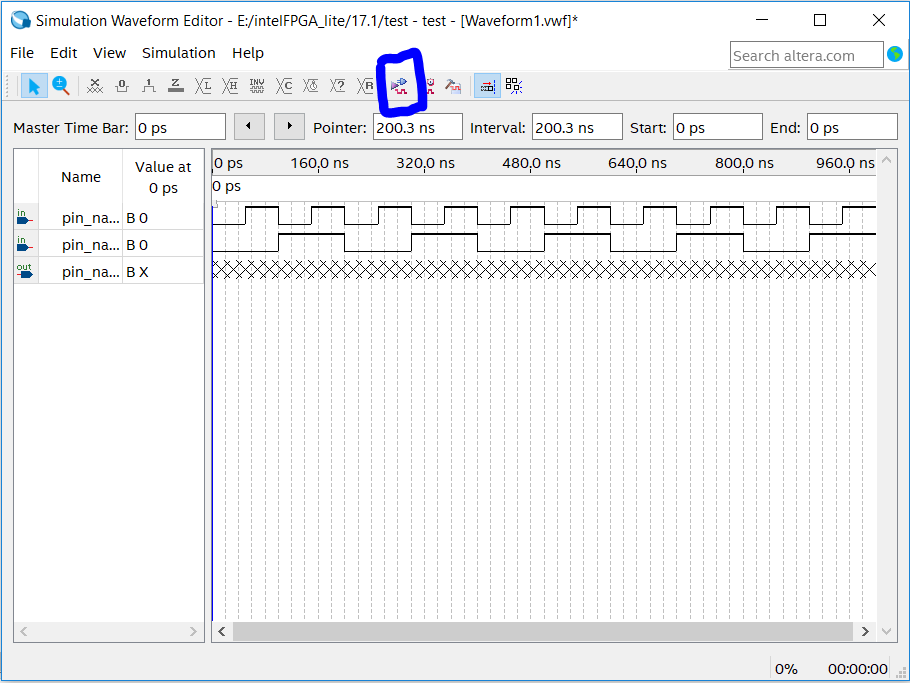




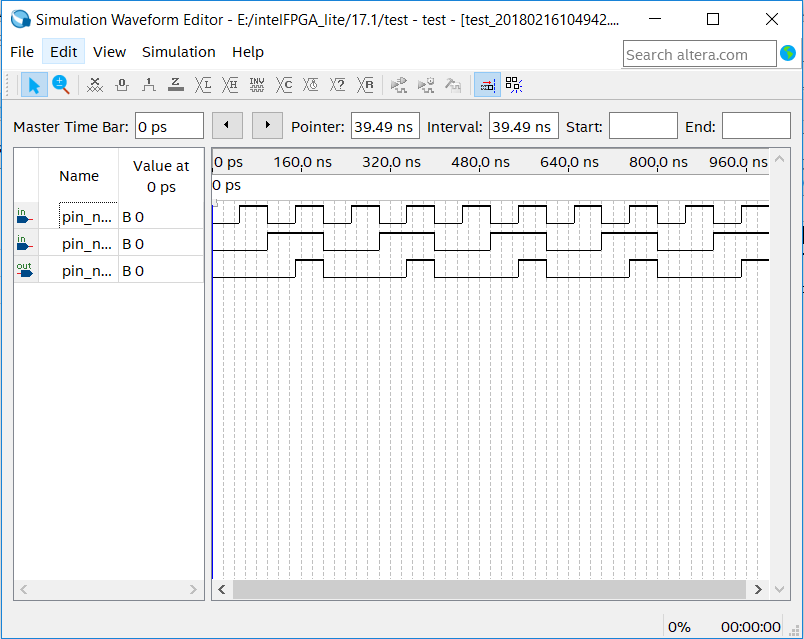
*Установка значений:*

*  - неизвестное значение. Используется для output’ов
*  - выделенная область устанавливается в логический ноль/еденицу
*  - счётчик

**Запуск симуляции**



**Результат**

****

**Лабораторная работа №2**

Используя Quartus Prime Lite Edition, и ЛР1 реализовать схему АЛУ, реализующего операцию умножения знаковых чисел с фиксированной точкой в дополнительном коде на основе алгоритма Бута.

Устройство должно работать в синхронном режиме, причём, после каждого синхроимпульса, состояние устройства изменяется на таковое, после выполнения очередного шага в алгоритме Бута.

Входы: 4-ти разрядные A[3..0] и B[3..0] - исходные данные, C - тактовый импульс.

Выходы: 8-ти разрядный Q[7..0] - результат и

Бит Ready - состояние выхода Q (1 - Q = A\*B, иначе - 0).

Теория на защиту: Алгоритм Бута(см. ЭУМКД)

**Лабораторная работа №3**

Написать программу, находящую решение квадратного уравнения **ax2 + bx + c = 0,** используя инструкции **SSE.**

**Теория на защиту: IEEE 754**

**Лабораторная работа №4**

Обработать массивы из 8 вещественных элементов используя SIMD инструкции **AVX** по следующему выражению (в зависимости от варианта):

1. F[i]=(A[i]+B[i])\*C[i]+D[i] , i=1...8;

2. F[i]=(A[i]+B[i])\*C[i]-D[i] , i=1...8;

3. F[i]=A[i]-B[i]\*C[i]+D[i] , i=1...8;

4. F[i]=A[i]-B[i]\*C[i]-D[i] , i=1...8;

5. F[i]=A[i] \*C[i]+B[i] \*D[i] , i=1...8;

6. F[i]=A[i] \*B[i] +C[i] -D[i] , i=1...8.

7. F[i]=A[i] -B[i] +C[i] \*D[i] , i=1...8.

8. F[i]=A[i] -B[i] +C[i] -D[i] , i=1...8.

9. F[i]=A[i] \*B[i] +C[i] -D[i] , i=1...8.

10. F[i]=(A[i]+B[i])\*(C[i]+D[i]) , i=1...8;

11. F[i]=(A[i]+B[i])\*(C[i]-D[i] ), i=1...8;

12. F[i]=(A[i]-B[i])\*(C[i]+D[i] ), i=1...8;

13. F[i]=(A[i]-B[i])\*(C[i]-D[i]) , i=1...8;

14. F[i]=(A[i] \*C[i])+(B[i] \*D[i]) , i=1...8;

15. F[i]=(A[i] \*B[i]) +(C[i] -D[i]) , i=1...8.

16. F[i]=(A[i] -B[i]) +(C[i] \*D[i]) , i=1...8.

17. F[i]=(A[i] -B[i]) +(C[i] -D[i] ), i=1...8.

18. F[i]=(A[i] \*B[i] )+(C[i] -D[i]) , i=1...8.

19. F[i]=A[i]+(B[i])\*C[i])+D[i] , i=1...8;

20. F[i]=A[i]+(B[i])\*C[i])-D[i] , i=1...8;

21. F[i]=A[i]-(B[i]\*C[i])+D[i] , i=1...8;

22. F[i]=A[i]-(B[i]\*C[i])-D[i] , i=1...8;

23. F[i]=A[i] \*(C[i])+B[i]) \*D[i] , i=1...8;

24. F[i]=A[i] \*(B[i]) +C[i] )-D[i] , i=1...8.

25. F[i]=A[i] –(B[i]) +C[i]) \*D[i] , i=1...8.

26. F[i]=A[i] –(B[i]) +C[i]) -D[i] , i=1...8.

27. F[i]=A[i] \*(B[i] )+C[i]) -D[i] , i=1...8.

Размер элемента массива D в 2 раза больше A, B, C.

Теория на защиту: SIMD, Расширения процессора, AVX, SSE