Факультет Программной инженерии и компьютерной техники

Лабораторная работа №3

Высокоуровневый синтез функциональных блоков СнК

по дисциплине «Информационная безопасность»

Вариант - 4

Группа: P3432

Выполнили:

Глотов Егор Дмитриевич

Чмурова Мария Владиславовна

Ефимов Арслан Альбертович

Преподаватель:

Быковский Сергей Вячеславович

г. Санкт-Петербург

2025 г.

Цель

Получить знания и навыки использования технологии высокоуровневого синтеза для разработки IP ядер

Изображение полученного аппаратного ускорителя с его интерфейсом

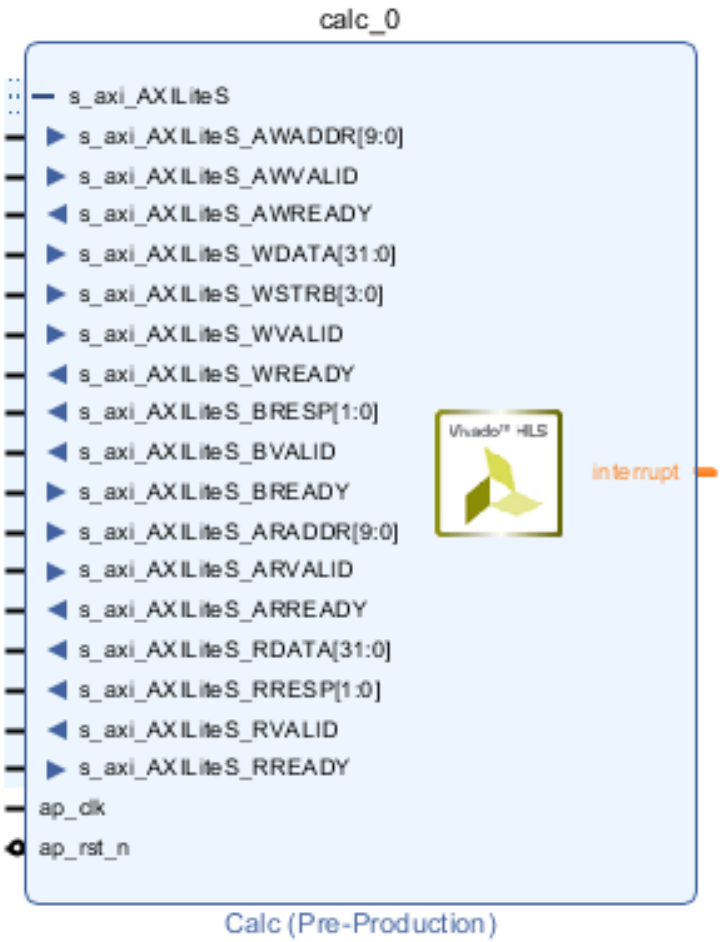

Рисунок 1. Полученный аппаратный ускоритель

Метрики для полученного аппаратного ускорителя без директив

1. Утилизация ресурсов FPGA

Utilization Estimates

- **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	3	0	279	-
FIFO	-	-	-	-	-
Instance	6	-	276	250	-
Memory	1	-	0	0	0
Multiplexer	-	-	-	134	-
Register	-	-	194	-	-
Total	7	3	470	663	0
Available	270	240	126800	63400	0
Utilization (%)	2	1	~0	1	0

Рисунок 2. Утилизация ресурсов

2. Время вычисления алгоритма при тактовой частоте в 100MHz:

$$35.845 - 12.365 = 23{,}48\ [\text{мкс}]$$

3. Временная диаграмма с результатами симуляции

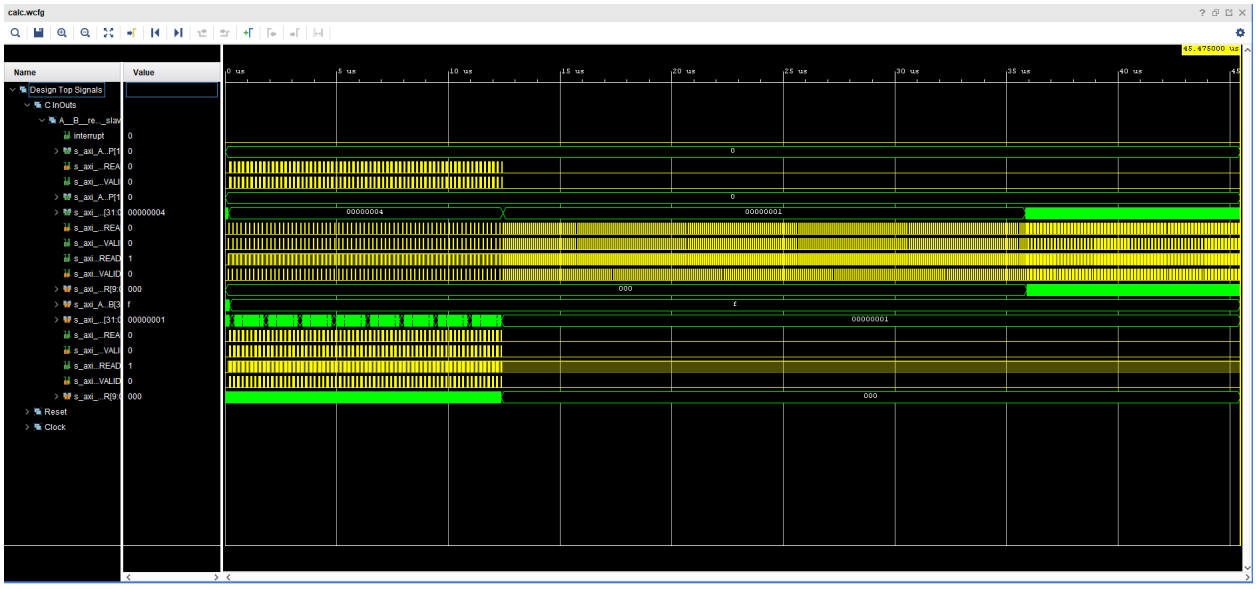

Рисунок 3. Временная диаграмма №1

Метрики для полученного аппаратного ускорителя с директивой unrolled loops

1. Утилизация ресурсов FPGA

Utilization Estimates

- **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	3	0	1158	-
FIFO	-	-	-	-	-
Instance	6	-	276	250	-
Memory	1	-	0	0	0
Multiplexer	-	-	-	839	-
Register	-	-	670	-	-
Total	7	3	946	2247	0
Available	270	240	126800	63400	0
Utilization (%)	2	1	~0	3	0

Рисунок 4. Утилизация ресурсов FPGA

2. Время вычисления алгоритма при тактовой частоте в 100MHz

$$35.665 - 12,445 = 23,22 \text{ [мкс]}$$

3. Временная диаграмма с результатами симуляции

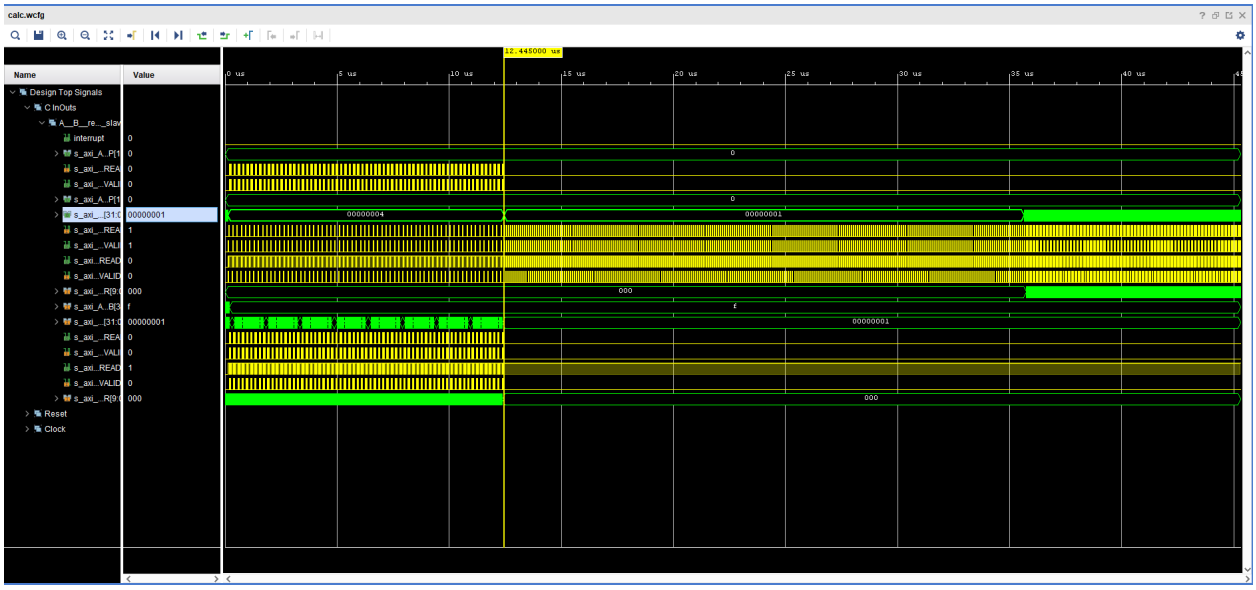

Рисунок 5. Временная диаграмма №2

Метрики для полученного аппаратного ускорителя с директивой pipelined loops

1. Утилизация ресурсов FPGA

Utilization Estimates

- **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	192	0	3635	-
FIFO	-	-	-	-	-
Instance	6	-	276	250	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	1029	-
Register	-	-	5455	-	-
Total	8	192	5731	4914	0
Available	270	240	126800	63400	0
Utilization (%)	2	80	4	7	0

Рисунок 6. Утилизация ресурсов FPGA

2. Время вычисления алгоритма при тактовой частоте в 100MHz

$$14.425 - 12.445 = 1.98 \text{ [мкс]}$$

3. Временная диаграмма с результатами симуляции

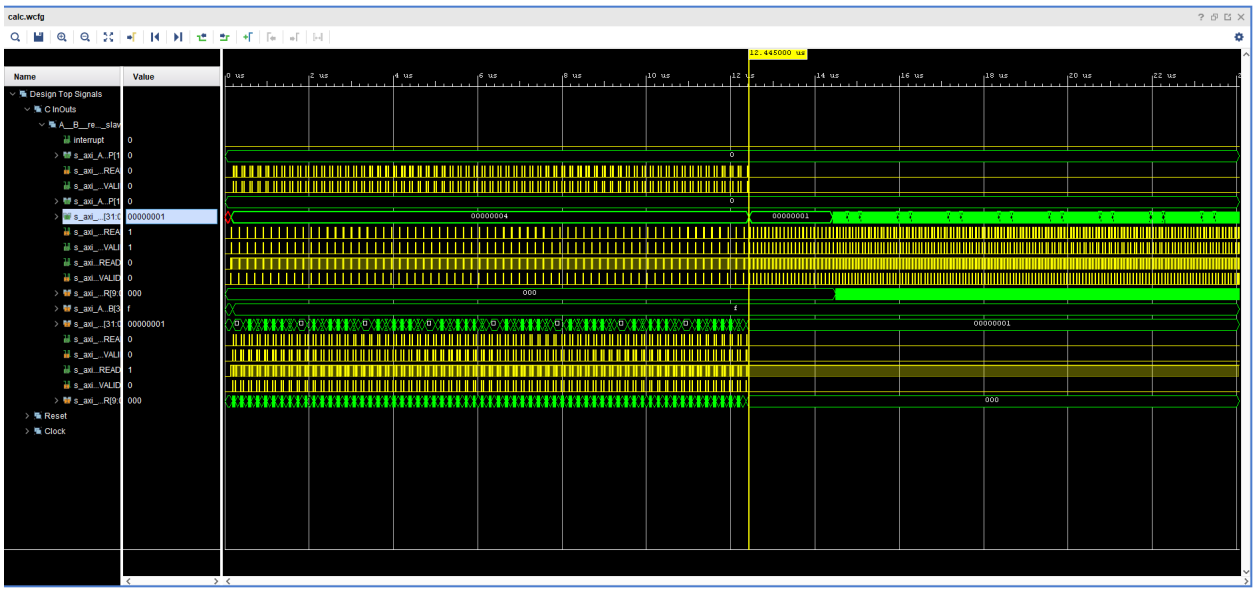

Рисунок 7. Временная диаграмма №3

Исходный код алгоритма на си

matrix.c:

```
#include "matrix.h"

void calc(
    data_t A[MATRIX_SIZE * MATRIX_SIZE],
    data_t B[MATRIX_SIZE * MATRIX_SIZE],
    data_t result[MATRIX_SIZE * MATRIX_SIZE]
) {
#pragma HLS INTERFACE s_axilite port=return
#pragma HLS INTERFACE s_axilite port=A
#pragma HLS INTERFACE s_axilite port=B
#pragma HLS INTERFACE s_axilite port=result

    data_t tmp[MATRIX_SIZE * MATRIX_SIZE];

    for (int i = 0; i < MATRIX_SIZE; i++) {
        for (int j = 0; j < MATRIX_SIZE; j++) {
            data_t sum = 0;
            for (int k = 0; k < MATRIX_SIZE; k++) {
                sum += A[i * MATRIX_SIZE + k] *
                       B[k * MATRIX_SIZE + j];
            }

            tmp[i * MATRIX_SIZE + j] = sum;
        }
    }

    for (int k = 0; k < MATRIX_SIZE; k++) {
        for (int l = 0; l < MATRIX_SIZE; l++) {
            result[k * MATRIX_SIZE + l] =
                tmp[k * MATRIX_SIZE + l] -
                B[k * MATRIX_SIZE + l];
        }
    }
}
```

Верификационное окружение

matrix_test.c:

```
#include <stdio.h>
#include <stdlib.h>
#include "matrix.h"

void init_ones(data_t *A) {
    for (int i = 0; i < MATRIX_SIZE; i++)
        for (int j = 0; j < MATRIX_SIZE; j++)
            A[i * MATRIX_SIZE + j] = 1;
}
void init_sequence(data_t *B) {
    for (int i = 0; i < MATRIX_SIZE; i++)
        for (int j = 0; j < MATRIX_SIZE; j++)
            B[i * MATRIX_SIZE + j] = j + 1;
}
int main() {
    int i, j;

    data_t A[MATRIX_SIZE * MATRIX_SIZE];
    data_t B[MATRIX_SIZE * MATRIX_SIZE];
    data_t result[MATRIX_SIZE * MATRIX_SIZE];

    init_ones(A);
    init_sequence(B);

    calc(A, B, result);

    FILE *fp = fopen("out.dat", "w");
    if (!fp) {
        perror("Cannot open out.dat");
        return 1;
    }

    for (i = 0; i < MATRIX_SIZE; i++) {
        for (j = 0; j < MATRIX_SIZE; j++) {
            fprintf(fp, "%d %d %d\n",
                    i, j, result[i * MATRIX_SIZE + j]);
        }
    }

    fclose(fp);

    printf("Comparing against out.golden.dat\n");

    if (system("diff -w out.dat out.golden.dat")) {
        printf("*********************************************\n");
        printf("FAIL: Output DOES NOT match the golden output\n");
        printf("*********************************************\n");
        return 1;
    } else {
        printf("*********************************************\n");
        printf("PASS: The output matches the golden output!\n");
        printf("*********************************************\n");
        return 0;
    }
}
```

Вывод

В ходе лабораторной работы получили знания и навыки использования технологии высокоуровневого синтеза для разработки IP ядер. Нам удалось создать аппаратные ускорители без использования и с использованием директив unrolled loops и pipelined loops.

В итоге получили следующие результаты работы алгоритма при тактовой частоте 100MHz: $23{,}48$ [мкс], 23.22 [мкс] и 1.98 [мкс] соответственно. Как видно из времени выполнения алгоритмов, работа с директивой pipelined намного быстрее, чем без директив и с директивой unrolled, поэтому, несмотря на немного повышенное потребление ресурсов ресурсов (DSP, FF, LUT) с директивой pipelined, мы выбираем именно этот вариант из-за самой высокой скорости работы.