Университет ИТМО

Системы на кристалле

Лабораторная работа №2

Вариант 3

Выполнили: Калугина Марина

Саржевский Иван

Группа: P3402

г. Санкт-Петербург

2020 г.

Содержание

Цель работы.

Получить базовые навыки использования средств высокоуровневого синтеза в процессе проектирования СнК.

Задание

1. Спроектировать и описать функциональность аппаратного ускорителя для алгоритма КИХ-фильтра из первой лабораторной работы на языке С, пригодную для синтеза в аппаратный блок.
2. Провести синтез аппаратного ускорителя.
3. Разработать тестовое окружение для проверки функциональности синтезированного аппаратного ускорителя.
4. Оценить следующие характеристики:
    a. Время выполнения алгоритма при частоте тактового сигнала в 100 МГц.
    b. Число занимаемых ресурсов ПЛИС (XC7A100T-1CSG324C)
    c. Время и занимаемые ресурсы ПЛИС с использованием следующих оптимизаций: раскрутка циклов, конвейеризация циклов.

Ход работы

Исходный код алгоритма

```
#include "fir.h"

void fir_filter(data_t x[N], data_t y[N-2], data_t p)
{
    if (p < 3 || p > 10)
      return;

     FIR_LOOP: for (int j = 0; j < 11 - p; ++j)
     {
      int sum =
                      x[0] * ((j <= 0 && 0 < j + p)? 1 : 0)
                    + x[1] * ((j <= 1 && 1 < j + p)? 1 : 0)
                    + x[2] * ((j <= 2 && 2 < j + p)? 1 : 0)
                    + x[3] * ((j <= 3 && 3 < j + p)? 1 : 0)
                    + x[4] * ((j <= 4 && 4 < j + p)? 1 : 0)
                    + x[5] * ((j <= 5 && 5 < j + p)? 1 : 0)
                    + x[6] * ((j <= 6 && 6 < j + p)? 1 : 0)
                    + x[7] * ((j <= 7 && 7 < j + p)? 1 : 0)
                    + x[8] * ((j <= 8 && 8 < j + p)? 1 : 0)
                    + x[9] * ((j <= 9 && 9 < j + p)? 1 : 0);

      y[j] = sum / p;
     }
}
```

Исходный код тестового окружения

```
#include <stdio.h>
#include "fir.h"

void erase_res(data_t res[N-2])
{
    for (data_t i = 0; i < 8; ++i)
      res[i] = 0;
}

int main () {
    FILE        *fp;

    data_t x[10] = { 13, 5, 3, 4, 6, 2, 3, 5, 5, 5 };
    data_t y[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };
    data_t exp_y[8] = { 7, 4, 4, 4, 3, 3, 4, 5 };

    fp=fopen("out.dat","w");
    for (data_t p = 3; p < 11; ++p)
    {
      fir_filter(&x, &y, p);
      fprintf(fp,"%d %d %d %d %d %d %d %d\n", y[0], y[1], y[2],
y[3], y[4], y[5], y[6], y[7], y[8]);
      erase_res(&y);
    }
    fclose(fp);

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

```
out.gold.dat:
7 4 4 4 3 3 4 5
6 4 3 3 4 3 4 0
6 4 3 4 4 4 0 0
5 3 3 4 4 0 0 0
5 4 4 4 0 0 0 0
5 4 4 0 0 0 0 0
5 4 0 0 0 0 0 0
5 0 0 0 0 0 0 0
```

Так как конкретные числовые значения на входе не влияют на логику алгоритма и тайминги, было принято решение протестировать алгоритм на одном входном наборе данных но со всеми возможными значениями размера окна.

Характеристики

Время выполнения алгоритма при частоте тактового сигнала в 100 МГц.

Так как у нас известны значения частоты и задержки, то для получения значения времени выполнения достаточно перемножить эти значения:

Таким образом, время выполнения алгоритма при частоте тактового сигнала в 100МГц: от 10 до 3370 нс.

Минимальное граничное значение равное 1 получается в случае, когда p выходит за рамки допустимых значений. В таком случае, происходит выход из программы за один такт.

Latency		Interval		
min	max	min	max	Type
1	337	1	337	none

Число занимаемых ресурсов ПЛИС

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	884
FIFO	-	-	-	-
Instance	-	-	394	238
Memory	-	-	-	-
Multiplexer	-	-	-	268
Register	-	-	324	-
Total	0	0	718	1390
Available	270	240	126800	63400
Utilization (%)	0	0	~0	2

Алгоритму необходимо 718 триггеров и 1390 LUT-элементов

Время и занимаемые ресурсы ПЛИС с использованием следующих оптимизаций: раскрутка циклов, конвейеризация циклов.

Раскрутка циклов

После раскрутки циклов, получились такие результаты занимаемого времени от 10 до 470 нс

Latency		Interval		
min	max	min	max	Type
1	47	1	47	none

Занимаемое число ресурсов:

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	7725
FIFO	-	-	-	-
Instance	-	-	3152	1904
Memory	-	-	-	-
Multiplexer	-	-	-	392
Register	-	-	1838	-
Total	0	0	4990	10021
Available	270	240	126800	63400
Utilization (%)	0	0	3	15

Конвейеризация циклов

После конвейеризация циклов, получились такие результаты занимаемого времени от 10 до 780 нс

Latency		Interval		
min	max	min	max	Type
1	78	1	78	none

Занимаемое число ресурсов:

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	886
FIFO	-	-	-	-
Instance	-	-	2283	1738
Memory	-	-	-	-
Multiplexer	-	-	-	134
Register	0	-	393	64
Total	0	0	2676	2822
Available	270	240	126800	63400
Utilization (%)	0	0	2	4

Вывод

Тип оптимизации	Максимальная оценка времени выполнения в тактах	Процент использованных FF	Процент использованных LUT
Без оптимизаций	337	~0	2
LOOP UNROLL	47	3	15
PIPELINE	78	2	4

В ходе выполнения лабораторной работы был реализован и оптимизирован алгоритм КИХ-фильтра.

В результате оптимизации при раскрутке циклов удалось уменьшить время выполнения в 8 раз, и чуть больше чем в 4 раза удалось уменьшить время выполнения при конвейеризации. При этом, ожидаемо, увеличилось и число занимаемых ресурсов. Например, процент использованных LUT’ов при LOOP UNROLL равен 15-ти процентам, а это значит, что на одну плату может уместиться всего 6 таких алгоритмов. Таким образом, чем больше мы выигрываем во времени выполнения алгоритма, тем больше занимаем ресурсов ПЛИС.