|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |  |
| МИНОБРНАУКИ РОССИИ | | | | | | |  |
| Федеральное государственное бюджетное образовательное учреждение  высшего профессионального образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | | | |  |
|  | Институт информационных технологий (ИТ) | | | | | | |
|  | Кафедра инструментального и прикладного программного обеспечения (ИППО) | | | | | | |
| **Отчет по лабораторной работе №2**  По дисциплине «Разработка ПАОН и АС»  Тема: «Разработка модели аппаратного ускорителя вычислений для программно-аппаратного комплекса» | | | | |
|  | | | | |
|  | | | | |
| Выполнил студент группы ИКМО-01-19 | | | Косиков М.И. | |
| Преподаватель | | | Тарасов И.Е. | |
|  | |  | |  | |
|  | |  | |

Москва 2019

**Цель работы:** разработка модели аппаратного ускорителя вычислений на языке программирования высокого уровня.

**Введение**

В настоящее время к программно-аппаратным комплексам могут предъявляться различающиеся требования по характеристикам, в зависимости от их назначения и области применения. Поэтому на практике невозможно разработать универсальную аппаратную платформу, одинаково хорошо подходящую для множества отличающихся областей применения. Например, для настольных компьютеров, ноутбуков, смартфонов и роутеров требуются процессоры, отличающиеся производительностью, энергопотреблением и возможностью выполнения операций определенных типов.

Кроме того, даже в составе одного комплекса часто выполняются задачи, относящиеся к разным классам по составу выполняемых операций. Например, в персональном компьютере (десктопе, ноутбуке) требуются как вычисления универсального характера с широким набором поддерживаемых операций, так и специфические операции над объектами в трехмерном пространстве для построения трехмерных изображений на экране дисплея. Целесообразно разделить такие задачи, реализовав в составе комплекса несколько вычислительных узлов. Можно убедиться, что в современных компьютерах используются как процессоры общего назначения (CPU), так и графические процессоры (GPU), ориентированные на выполнение операций, характерных для построения трехмерных изображений на двумерном дисплее.

В отличие от *гомогенной* («однородной») архитектуры, подразумевающей использование множества одинаковых вычислительных устройств, такие комплексы имеют *гетерогенную* («неоднородную») архитектуру.

Универсальной платформой для реализации аппаратных ускорителей являются микросхемы программируемой логики с архитектурой FPGA. Они представляют собой матрицу программируемых логических ячеек с конфигурируемыми соединениями. Кроме базовых ячеек, на микросхеме FPGA размещены также блоки статической памяти, аппаратные компоненты «умножение с накоплением» и высокоскоростные последовательные приемопередатчики. Поэтому с применением FPGA можно разрабатывать ускорители, предназначенные для следующих задач:

1. На базе компонентов «умножение с накоплением» – цифровые фильтры, блоки для операций с матрицами, ускорители преобразования Фурье, нейросети.

2. На базе логических ячеек – вспомогательные контроллеры, устройства аппаратной защиты, преобразователи интерфейсов, устройства сбора данных.

Высокоскоростные последовательные приемопередатчики являются единственным способом для передачи сигналов со скоростями 3, 6 и более Гбит/с (до 58 Гбит/с на базе 7 нм FPGA). Такие компоненты используются в основном в магистральном сетевом оборудовании, разработка которого требует специальных технологий проектирования печатных плат и приборов для отладки высокоскоростных аналоговых цепей.

Эффективно продемонстрировать взаимодействие процессора и аппаратного ускорителя можно на примере ускорения цифровой обработки сигналов. Работа цифрового фильтра (фильтра *с конечной импульсной характеристикой*, сокращенно КИХ-фильтра, также FIR – Finite Impulse Response) состоит в вычислении выхода по формуле:

y = k1x1 + k2x2 + … knxn  (1)

где x1, x2,… xn – отсчеты входного сигнала в моменты времени t-1, t-2, … t-n соответственно;

k1, k2,… kn – коэффициенты.

Расчету коэффициентов фильтра для достижения заданных характеристик посвящены разделы теории цифровой обработки сигналов. Существует множество классов фильтров, однако все они требуют вычислений по формуле (1). Нетрудно убедиться, что фильтр с N коэффициентами требует N умножений и N-1 сложений. Для процессора общего назначения это означает необходимость выполнения как минимум 2N – 1 операций. Кроме того, в следующий момент времени фильтр должен прочитать из внешнего источника сигнала новое значение x1, а предыдущие значения сдвинуть на один элемент массива (при этом теряется отсчет xn). Для этого также потребуются такты процессора.

В то же время можно создать цифровую схему, выполняющую вычисления по (1) за один такт, одновременно выполняя и сдвиг значений x. Такая схема может иметь независимый от процессора вход x и выход y, принимая только значения k1…kn, чтобы обеспечить реализацию фильтров с разными наборами коэффициентов.

Моделирование ускорителя программно-аппаратного комплекса в виде аппаратного цифрового фильтра можно выполнить с помощью одного из языков высокого уровня. Модель фильтра должна содержать массивы x[N] и k[N], входную переменную x и выходную переменную y. Для имитации взаимодействия с процессором должна быть предусмотрена функция, задающая значение выбранного коэффициента в массиве k.

На рис. 1 показан пример амплитудно-частотной характеристики КИХ-фильтра, построенного в специализированной утилите FIR Compiler, входящей в САПР Vivado. Утилита предназначена для создания аппаратных фильтров, поэтому результаты моделирования могут быть впоследствии проверены на базе FPGA.

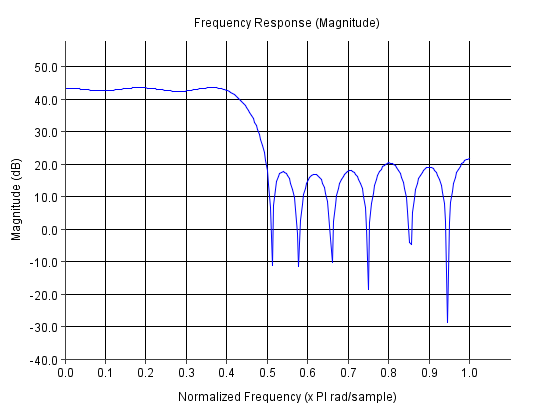


Рис. 1. Пример АЧХ КИХ-фильтра.

Коэффициенты этого фильтра имеют следующие значения:

6, 0, -4, -3, 5, 6, -6, -13, 7, 44, 64, 44, 7, -13, -6, 6, 5, -3, -4, 0, 6

По горизонтальной оси на рис. 1 отложена частота в радианах на цифровой отсчет. Значение 1.0 соответствует одному радиану на точку цифрового представления сигнала, т.е. один период входного сигнала должен иметь 2π (~6.28) точек. Соответственно, точка графика с частотой 0,1 соответствует входному синусоидальному сигналу с 62.8 точками на каждый период и т.д.

Для воспроизведения АЧХ фильтра необходимо последовательно подавать на него синусоидальные сигналы, приведенные к целочисленному формату. В процессе их получения необходимо умножать значение синуса на выбранную амплитуду (100 или 1000), чтобы избежать округления до нуля значений синуса, вычисляемого в диапазоне -1…1. Для демонстрации реакции фильтра входной сигнал для каждой выбранной частоты следует подавать в течение некоторого времени, чтобы пронаблюдать поведение выхода y фильтра. По вертикальной оси отложено отношение амплитуд выходного и входного сигналов, выраженное в децибелах.

При моделировании фильтра необходимо иметь в виду, что при умножении двоичных чисел разрядность результата может превысить 32. В общем, для представления результата умножения требуется разрядность, равная сумме разрядностей операндов. Кроме того, если последовательно складывать числа, требуемая разрядность также может возрасти. Например, при сложении 2 чисел может потребоваться дополнительный бит для представления результата, при сложении 4 чисел – 2 бита, и т.д. Однако разрядность, требуемая для конкретного набора коэффициентов, зависит от фактического заполнения разрядной сетки этих коэффициентов.

**Исходные данные**

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Амплитуда | 1000 |
| Количество этапов фильтрации | 21 |
| Количество тактов | 120 |
| Максимальное значение частоты в радианах | 2 |
| Шаг частоты в радианах | 0.1 |
| Массив коэффициентов фильтра | 6, 0, -4, -3, 5, 6, -6, -13, 7, 44, 64, 44, 7, -13, -6, 6, 5, -3, -4, 0, 6 |

**Результат**

Результаты моделировании фильтра по исходным данным отображены на графике.

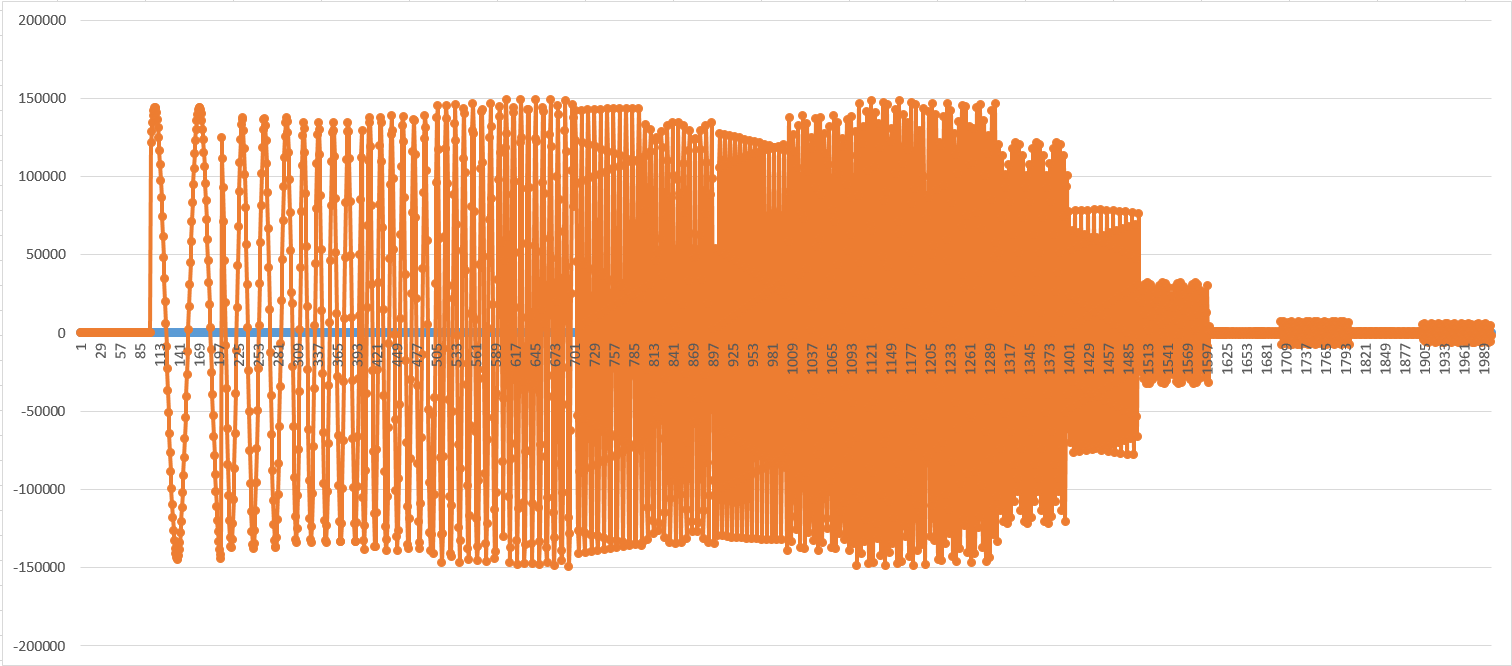


Рис. 2. Результат работы АЧХ КИХ-фильтра.

**Выводы**

В ходе данной работы разработана программа, которая моделирует работу цифрового фильтра и записывает результаты преобразований в Excel-файл.

Моделирование ускорителя программно-аппаратного комплекса в виде аппаратного цифрового фильтра выполнено на языке программирования C# с помощью IDE Microsoft Visual Studio.

**Приложение**

1. Класс реализации фильтра

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace FSP\_lab\_2

{

struct ArgumentAndFunc

{

public double y;

public int t;

}

class FilterSound

{

const int MathConst = 1000;

// Массив коэффициентов фильтра [k[21]]

int[] arrayOfCoef;

// lineTime - отрезок количества тактов. [100]

long lineTime;

// filterLine - количество этапов фильтрации [21]

long filterLine;

double w;

double stepW;

const string path = @"C:\lab2\note.txt";

public FilterSound(int[] arrayOfCoef, long lineTime, long filterLine, double w, double stepW)

{

this.arrayOfCoef = arrayOfCoef;

this.lineTime = lineTime;

this.filterLine = filterLine;

this.w = w;

this.stepW = stepW;

if (arrayOfCoef.Length != filterLine)

{

throw new Exception("Длина массива должна соответствовать количеству этапов фильтрации");

}

}

private double GetSin(double w, double t)

{

double result = MathConst \* Math.Sin(w \* t);

return result;

}

private double GetY(double[] arrayOfX)

{

double sum = 0;

//Расчёт Y - количество коэффициентов фильтра на аргумент

for (int i = 0; i < filterLine; i++)

{

sum += arrayOfCoef[i] \* arrayOfX[i];

}

return sum;

}

public void GetSoundPerformance()

{

List<ArgumentAndFunc[]> schedule = new List<ArgumentAndFunc[]>();

for (double i = 0; i < w; i+=stepW)

{

schedule.Add(DuringLineTime(i));

}

OnExcel(schedule);

}

private ArgumentAndFunc[] DuringLineTime(double w)

{

List<ArgumentAndFunc> arrayOfY = new List<ArgumentAndFunc>();

for (int displacement = 0; displacement < lineTime; displacement++)

{

ArgumentAndFunc newStruct = new ArgumentAndFunc();

// Создаём массив ИКСов по getSin, считаем по массиву Y по функции getY и записываем Y в arrayOfY

newStruct.y = GetY(GetArrayOfX(displacement, w));

// displacement - в данном контексте количество тактов

newStruct.t = displacement;

arrayOfY.Add(newStruct);

}

return arrayOfY.ToArray();

}

public void cleanOrCreateExcel()

{

using (FileStream fstream = new FileStream(path, FileMode.OpenOrCreate))

{

}

using (FileStream fstream = new FileStream(path, FileMode.Truncate))

{

}

}

private double[] GetArrayOfX(int displacement, double w)

{

// Получение 20-ти иксов.

List<double> resultArray = new List<double>();

// displacement - это сдвиг, то есть начальное и конечное значение на оси времени, к которому прибавляется filterLine

// Начальное значение зависит от времени, из duringLineTime, a конечное от времени + filterLine

for (int t = displacement; t < filterLine + displacement; t++)

{

// Добавление их в массив.

// X = 1000\* sin(w, t)

resultArray.Add(GetSin(w, t));

}

return resultArray.ToArray();

}

private void OnExcel(List<ArgumentAndFunc[]> data)

{

// запись в файл

using (StreamWriter sw = new StreamWriter(path, false, System.Text.Encoding.Default))

{

foreach (var array in data)

{

foreach (var item in array)

{

string str = item.t + ";" + item.y + "\n";

sw.Write(str);

Console.Write(str);

}

}

}

}

}

}

1. Класс запуска приложения

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FSP\_lab\_2

{

class Program

{

static void Main(string[] args)

{

int[] k = {6, 0, -4, -3, 5, 6, -6, -13, 7, 44, 64, 44, 7, -13, -6, 6, 5, -3, -4, 0, 6 };

FilterSound fsp = new FilterSound(k,100, 21, 2, 0.1);

// Добавить функцию отчистки

fsp.cleanOrCreateExcel();

fsp.GetSoundPerformance();

}

}

}