УДК 681.5.08, 681.513.2, 681.518.3

Методика реализации уравнений имитатора аэродинамических воздействий на программируемой логической интегральной схеме

Константинов А.А.

Московский авиационный институт (национальный исследовательский университет), МАИ, Волоколамское шоссе, 4, Москва, А-80, ГСП-3, 125993, Россия

e-mail: sasha_konst@pochta.ru

Аннотация

Целью работы является разработка методики, посредством которой можно эффективно разработать программное обеспечение для реализации математического аппарата имитатора аэродинамических воздействий, выполняемого на ПЛИС.

Для разработки методики был проведен системный анализ аэродинамического уравнения, который входит в состав имитатора аэродинамических воздействий (ИАВ).

При разработке методики использовались существующие рекомендации по программированию ПЛИС в LabVIEW.

Применение данной методики, при разработке имитатора аэродинамических воздействий позволило увеличить скорость работы алгоритма, для восьми взаимосвязанных каналов, с 17,8 кГц до 416,6 кГц, что соответствует увеличению производительности в 24 раза. При получении результатов использовалось оборудование National Instruments: восьми канальный аналого-цифровой преобразователь (АЦП) с разрядностью 16 бит и частотой дискретизации 750кГц и

восьми канальный цифро-аналоговый преобразователь (ЦАП) с разрядностью 16 бит и частотой дискретизации 1МГц.

Предлагаемая методика может быть использована при разработке различных измерительно-информационных и управляющих систем, с высоким требованием к быстродействию, точности и многоканальности, с явной синхронизацией исполнения во времени.

Ключевые слова: программируемая логическая интегральная схема, цифро-аналоговый преобразователь, имитатор аэродинамических воздействий, аналогово-цифровой преобразователь, метод электромеханического моделирования

В основе имитатора аэродинамических воздействий (ИАВ) лежит метод электромеханического моделирования (ЭММ). Идея метода состоит в воспроизведении аэродинамических сил на механическую конструкцию в стендовых условиях, без реального воздушного потока. Для этого необходимо, в частности, заменить распределенные аэродинамические силы, действующие на упругую конструкцию при ее колебаниях в потоке, искусственными (эквивалентными) сосредоточенными силами с помощью электродинамических силовозбудителей. Этот метод занимает промежуточное положение между полностью экспериментальными и полностью аналитическими методами. Фактически испытывается упругая конструкция – натурный летательный аппарат (ЛА) или его динамически подобная модель (ДПМ). В то же время аэродинамические силы являются искусственными и должны вычисляться, в режиме жесткого реального времени, для всех «полетных» условий, возбуждающие силы должны

распределяться между силовозбудителями и воспроизводиться специальным электронным блоком.

Схема эксперимента с ЭММ изображена на рис.1. Для возбуждений колебаний конструкции используются сигналы датчиков, преобразованные вычислительным блоком (ВБ) ИАВ.

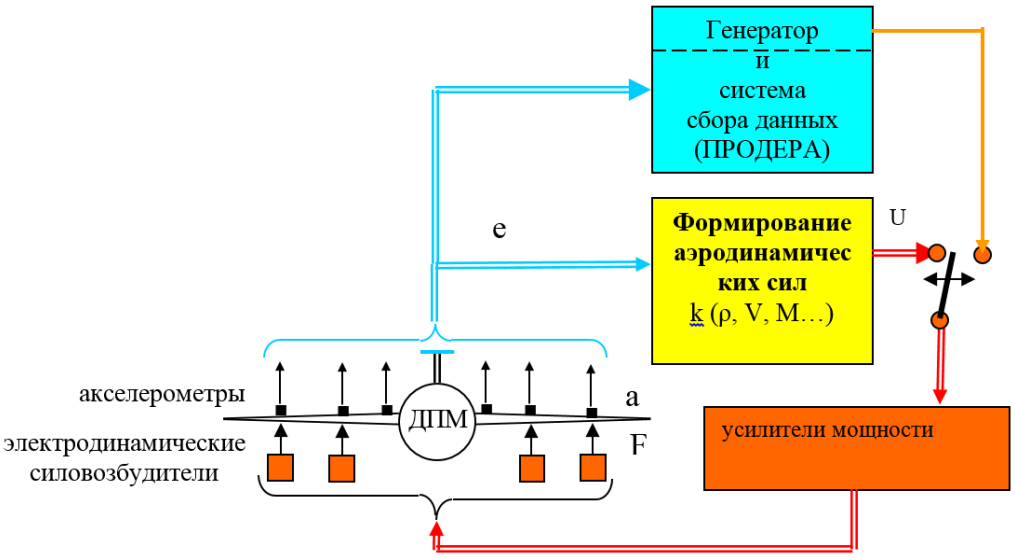

Рис. 1. Схема испытаний методом ЭММ.

На рис.1 приведены следующие обозначения: a - ускорения точек конструкции, F - усилия электродинамических силовозбудителей, e - электрические сигналы датчиков ускорения, U - электрический сигнал на выходе из имитатора аэродинамических воздействий.

ИАВ обеспчевивает непрерывное формирование из комбинации входных сигналов акселерометров ряда выходных напряжений, пропорциональных сосредоточенным аэродинамическим силам на несущую поверхность летательного аппарата (ЛА), к которым сводятся воздействия со стороны потока воздуха. В таком случае преобразование сигналов реализуется в соответствии с аэродинамический

теорией, определяющей зависимость каждой силы от мгновенных величин виброскорости и виброперемещения отдельных точек ЛА.

Реализация законов управления осуществляетя изменением коэффициентов преобразования в соответствии с заданным режимом управления величинами общих (основных) относительных коэффициентов: скорости набегающего потока v, плотности , числа Струхаля Sh и числа Маха M, либо их комбинации.

Типовые режимы управления управления:

1. Изменение одного из коэффициентов при постоянных остальных;
2. Парное – одновременное изменение двух коэффициентов при постоянных остальных;
3. Одновременное измеенение трех или четырех коэффициентов.

Все соотношения между коэффициентами при их изменении выбираются и задаются по результатам предварительных расчетов.

Многие расчетные схемы приводят к линейной системе дифференциальных уравнений, описывающих малые колебания упругой конструкции в потоке:

$$M\ddot{y}(t)+H\dot{y}(t)+Gy(t)=f^{A}(t), \qquad (1)$$

где M,H и G - матрицы инерции, демпфирования и жесткости соответственно, t - текущий момент времени, y - мгновенное значение вектора перемещений точек измерения конструкции; f^{A} - мгновенное значение вектора сосредоточенных аэродинамических сил, который может быть вычислен на основе любой аэродинамической теории и процедуры преобразования распределенных сил в ряд энергетически эквивалентных сосредоточенных сил. При вывешивании

механической конструкции на стенде, значения вектора сосредоточенных аэродинамических сил становятся тождественно равными значениям вектора сил возбуждения [1, 2]:

$$f(t) = f^{A}(t), \tag{2}$$

где f - вектор сил возбуждения.

При использовании датчиков ускорения, дифференциальное уравнение (1) приводится к интегральной форме:

$$Ma(t) + \frac{H}{T}\int a(t)dt + \frac{G}{T^2}\iint a(t)dt^2 = f(t), \tag{3}$$

где a - мгновенное значение ускорения точек конструкции.

Вибродатчики, установленные на конструкции, преобразуют ускорения соответствующих точек в электрические сигналы e. Эти сигналы преобразуются вычислительным устройством в выходные напряжения u. После усиления мощности эти напряжения u преобразуются в силы f электродинамическими силовозбудителями. Соответствующие преобразования описываются уравнениями:

$$e = R\,a\ ;\ u = Ke\ ;\ f = Qu, \tag{4}$$

где R и Q - диагональные матрицы калибровочных коэффициентов датчиков и силовозбудителей соответственно; K - матрица аэродинамических преобразований.

Чтобы обеспечить высокое быстродействия при реализации уравнения (3), его расчет необходимо перенести на программируемую логическую интегральную схему (ПЛИС). Для ввода сигналов ускорения в ИАВ используются аналого-

цифровые преобразователи (АЦП), а значение выходного сигнала преобразуется к аналоговому виду посредством цифро-аналогового преобразователя (ЦАП).

Для того, чтобы реализовать уравнение на ПЛИС, необходимо его привести к дискретному виду и провести масштабирование входных и выходных сигналов, исходя из разрядности АЦП и ЦАП, для численного представления на ПЛИС.

По правилам линейной алгебры, элемент вектора выходного сигнала рассчитывается как:

$$u_i(t_c) = \sum_j^n g_{ij} y_j(t_c) + \sum_j^n h_{ij} \dot{y}_j(t_c) + \sum_j^n m_{ij} \ddot{y}_j(t_c), \qquad (5)$$

где t_c - текущий момент времени, g_{ij} - элемент матрицы жесткости G, h_{ij} - элемент матрицы демпфирования H, m_{ij} - элемент матрицы инерции, y_j - перемещение точек исследуемого объекта в единицах измеряемой физической величины, $\dot{y}_j$ - скорость движения точек исследуемого объекта в единицах измеряемой физической величины, $\ddot{y}_j$ - ускорения точек исследуемого объекта в единицах измеряемой физической величины, n - число точек измерения, i - номер канала измерения и возбуждения, j - номер столбца матриц G и H.

Значение измеряемой физической величины внутри ПЛИС описывается в виде:

$$e_i(t_c) = [A_i \cdot U_i(t_c) \cdot 2^{-N_{ADCi}}], \qquad (6)$$

где i - номер канала АЦП, $e_i(t_c)$ - числовое представление исходного сигнала датчика после оцифровки в текущий момент времени, A_i - размах диапазона

измерения, $U_i(t_c)$ - текущее значение измеренного сигнала, N_{ADCi} - разрядность канала АЦП.

Значение ускорения в точке рассчитывается:

$$\ddot{y}_i^*(t_c) = \left\lfloor f\left(e_i(t_c) - e_i^0\right)\right\rfloor, \tag{7}$$

где $\ddot{y}_i^*$ - цифровое значение ускорения в точке, e_i^0 - цифровое значение уставки (смещение нуля), f - функция фильтрации, квадратные скобки – округление до ближайшего целого.

Параметры фильтра выбираются на основании технических требований при реализации ИИУК.

Значение скорости в точке рассчитывается:

$$\dot{y}_i^*(t_c) = \left[\dot{s}_i^*(t_c)\cdot 2^{n_i^h - N_{ADCi} + r}\right], \tag{8}$$

где $\dot{y}_i^*$ - цифровое значение скорости в точке, $\dot{s}_i^*(t_c)$ - промежуточная сумма, пропорциональная скорости точек конструкции в текущий момент времени, n_i^h - количество значащих разрядов интеграторов скорости, r - параметр, описывающий тип данных АЦП ($r = 0$ - беззнаковый тип данных, $r = 1$ - знаковый тип данных).

$$\dot{s}_i^*(t_c) = \dot{s}_i^*(t_{c-1}) + \ddot{y}_i^*(t_c), \tag{9}$$

Значение перемещения в точке рассчитывается:

$$y_i^*(t_c) = \left[s_i^*(t_c)\cdot 2^{n_i^g - (N_{ADCi} + r)}\right] \tag{10}$$

где y_i^* - цифровое значение перемещения в точке, $s_i^*(t_c)$ - промежуточная сумма, пропорциональная перемещению точек конструкции в текущий момент времени, n_i^g - количество значащих разрядов интеграторов перемещения.

$$s_i^*(t_c) = s_i^*(t_{c-1}) + \dot{s}_i^*(t_{c-1}) + \left\lfloor \ddot{y}_i^*(t_c) \cdot 0{,}5 \right\rfloor \tag{11}$$

Цифровое значение силового воздействия:

$$u_i^*(t_c) = \left[\left(\sum_j^n g_{ij} y_j^*(t_c) + \sum_j^n h_{ij} \dot{y}_j^*(t_c) + \sum_j^n m_{ij} \ddot{y}_j^*(t_c) \right) \cdot 2^{-(W_i - N_{DACi} + r)} \right], \tag{12}$$

где W - разрядность расширенной разрядной сетки результата, N_{DACi} - разрядность i-ого канала ЦАП.

$$W_i = n_i^k + \log_2 n + 1, \tag{13}$$

где n_i^k - разрядность блока перемножения вектора строчки на вектор-столбец.

Перед тем, как приступить к реализации алгоритма на ПЛИС необходимо проанализировать математические операторы и функции, лежащие в его основе, а также учесть аппаратные ограничения, которые накладывает архитектура ПЛИС.

Ограничения, накладываемые архитектурой ПЛИС:

- Все векторы должны быть заранее определенной длины (фиксированной);
- Все векторы должны быть одномерными;
- ПЛИС не поддерживает вычисления с плавающей запятой на аппаратном уровне;
- Алгоритм ограничен емкостью ПЛИС.

Для реализации программного обеспечения ПЛИС, в качестве примера, взята среда программирования LabVIEW.

Методика реализации программы для ПЛИС:

1) Разбивка формул на отдельные типовые математические операции;
2) Разделение параметров на быстро и медленно меняющиеся;
3) Перенос расчета медленно меняющихся параметров на хост;
4) Предварительный выбор способа реализации каждой математической операции на ПЛИС;
5) Программное распараллеливание независимых операций (данное положение важно для синхронного исполнения параллельных веток);
6) Замена элементов лицевой панели константами на блок-диаграмме (если константа не представляет собой меняющийся параметр);
7) Свертка связанных констант и в одно значение;
8) Выбор типа данных для каждой операции;
9) Предварительная разработка программы ПЛИС;
10) Отладка программы в режиме моделирования;
11) Предварительный расчет скорости работы алгоритма;
12) Предварительная оценка аппаратной емкости алгоритма;
13) Компиляция программы ПЛИС и запуск на целевом устройстве;
14) Проверка программы ПЛИС на соответствие требованиям;
15) При необходимости и возможности – корректировка программы ПЛИС:

a. Увеличение производительности программы –замена отдельных участков кода на более быстрые;
b. Корректировка по емкости участка кода – замена отдельных участков кода на менее ресурсоемкие.

Разбивка формул на отдельные типовые математические операции.

Разбивка формулы на составляющие необходима, для составления перечня математических операций, которые необходимо реализовать. При этом используются различные математические преобразования и теории для упрощения выражений, рассматриваются различные формы записи и соответственно программные реализации математических операций.

Разделение параметров на быстро и медленно меняющиеся

Разделим все параметры, которые участвуют в расчетах и рассчитываются сами или опрашиваются в режиме реального времени, на быстро меняющиеся и медленно меняющиеся.

Разделение осуществляется для каждого аппаратного модуля (вычислителя).

Быстро меняющимися параметрами назовем те, которые меняются в темпе поступающей информации и жестко привязаны к временным отсечкам вычислителя (аппаратного узла, контроллера, процессора), и в обязательном порядке должны быть реализованы в рамках этого модуля (назовем такой вычислитель «целевым устройством»).

Медленно меняющимися параметрами назовем те, которые могут не иметь жесткой привязки к временным отсечкам вычислителя, и расчет которых осуществляется со скоростью значительно меньшей, чем быстро меняющихся.

Перенос расчета медленно меняющихся параметров на хост

Быстро меняющиеся параметры зачастую участвуют в контуре управления и контуре системы автоматической защиты, в связи с чем требуется безотказность работы. Безотказность работы достигается разгрузкой «ответственного вычислителя» от лишних, для него, расчетов.

Такая разгрузка возможна, если архитектура комплекса разделена функционально и аппаратно на несколько уровней (например древовидная архитектура). При этом медленно меняющиеся параметры и операции над ними переносятся на уровень выше (назовем его «хост»), а результаты передаются на целевое устройство посредством каналов передачи данных. [3]

Осуществлять обмен данными с ПЛИС можно несколькими способами. Наиболее часто встречающиеся:

1) Программный опрос (поллинг);
2) Обмен данными по прерываниям;
3) Канал FIFO DMA.

Предварительный выбор способа реализации каждой математической операции на ПЛИС

Выбираются палитры (библиотеки) LabVIEW и их функции, посредством которых реализуется каждая математическая операция. Для одинаковых операций изначально рассматривается один вариант реализации.

Для отобранных библиотек и их функций, можно построить карты распределения ресурсов и определить скорости работы каждой операции (произведя ряд компиляций простейших программ с их использованием с различными конфигурациями). Полученные данные интерполируются полиномами малых степеней - одномерными для скалярных величин, и двухмерными для операций с векторами, для последующей оценки ресурсоемкости операции с произвольной разрядностью и глубиной.

Так, к примеру, рекомендуется избегать непосредственной операции деления на ПЛИС.

Программное распараллеливание независимых операций

Независимыми операциями называются те, результаты и входные сигналы которых не зависят друг от друга. Такие операции можно выполнить параллельно, уменьшив время исполнения кода.

Узлы и функции с различным быстродействием целесообразно размещать в разных параллельных циклах, тогда медленно исполняемые функции не будут тормозить выполнение быстрых.

При наличии длинных цепочек последовательно связанных узлов, эффективно применять конвейеризацию алгоритма, а также применять Single Cycle Timed Loop (однотактовые циклы).

Также целесообразно, выносить в отдельный цикл обмен данными между вычислителями, а получаемые и принимаемые данные передавать в математический аппарат посредством областей памяти, расположенных на ПЛИС. Однако, следует учесть, что компоненты программы, которые совместно используются в различных ветвях алгоритма, являются разделяемыми. Если не предпринять специальных мер, то не исключено, что один и тот же ресурс в один и тот же момент времени может быть затребован различными параллельно исполняемыми циклами – при этом возникает подмена или даже потеря данных до того, как они будут обработаны. Одновременное обращение к ресурсам может повлиять на детерминизм исполнения задач, послужить причиной некорректного выполнения операций.

Для достижения максимально возможного детерминизма и повышения быстродействия следует избегать использования разделяемых ресурсов, однако на практике в приложениях ПЛИС редко это удается.

Наиболее часто в качестве разделяемых ресурсов выступают как компоненты ПЛИС и модуля ввода-вывода, так и компоненты программ:

- каналы ввода-вывода;
- блоки памяти, используемые просто для хранения данных – Memory или для быстрого обмена данными – FIFO (first input – first output);
- локальные переменные и нереентерабельные subVI (подпрограммы и функции).

В LabVIEW существует возможность избежать проблемы разделяемого ресурса, связанной с использованием subVI. Для этого достаточно сделать subVI реентерабельным – любой вызов subVI из разных мест основной программы создает

свою собственную копию подпрограммы и обрабатывает свои потоки данных, исключая тем самым конфликты. В ПЛИС реентерабельный subVI представляется копиями схем, причем количество копий равно количеству источников запросов на этот subVI.

Очевидно, что применение реентерабельных subVI не только освобождает от необходимости решать проблему разделяемого ресурса, но и существенно повышает быстродействие приложения. Однако это приводит к значительному увеличению количества используемых логических вентилей, т.е. неэкономному расходованию ресурсов ПЛИС.

В тех случаях, когда нужны нестандартные решения, исключения конфликтов добиваются программными средствами, используя функции палитры синхронизации. [3]

При необходимости синхронного ввода данных от каналов (синхронный опрос), функция опроса входов должна вызываться один раз на все каналы. При использовании для каждого канала своего экземпляра функции опроса каналов, происходит их поочередный вызов, что заметно уменьшает скорость работы с функциями ввода/вывода в рамках всего приложения.

Замена элементов лицевой панели константами на блок-диаграмме

При наличии элементов лицевой панели, которые хранят единственное значение, которое не может быть изменено в процессе работы программы, их заменяют константами на блок-диаграмме. Это связано с тем, что массивы объектов лицевой

панели, а также массивы данных расходуют большое количество логических элементов ПЛИС. В связи с этим не рекомендуется создавать на лицевой панели массивы индикаторов или управляющих элементов. Для хранения массивов данных лучше использовать блочную память ПЛИС – в объектах типа Memory или FIFO.[3]

Свертка связанных констант в одно значение

При наличии целой цепочки операций над константами (часто последовательно организованной), целесообразно ее заменить заранее вычисленным результатом. То же справедливо для медленно меняющихся параметров, передающих значение результата вычислений ПЛИС, с хоста.

Выбор типа данных для каждой операции

При выборе типов данных для каждой операции необходимо учитывать диапазон изменения входных и выходных параметров. Так, для числа в диапазоне от 0 до 3 целесообразно выбрать тип данных fix-point формата fxp<+2,2>, вместо типовых I16, U16. Если же исходные данные представлены в форматах, требующих большего числа элементов памяти, то на соответствующем этапе обработки целесообразно выполнить преобразование типов данных.

В некоторых случаях, для повышения производительности рекомендуется объединять данные малой разрядности в слова. [3]

Рекомендуется избегать сложных и громоздких кластеров, данные которых, при выполнении функций, частично не задействуются. Учитывая, что трассировка на ПЛИС кластера представляет собой связанный пучок сигнальных линий,

незадействованные данные кластера попросту тратят ресурсы. Гораздо эффективнее производить поэлементную трассировку. Данная операция требует от программиста больше времени, однако положительно сказывается на объеме затраченных ресурсов.

Предварительная разработка программы ПЛИС

Осуществляется разработка программы, реализующей математический аппарат с учетом п.1-п.8, для ПЛИС.

Отладка программы в режиме моделирования

В данном пункте осуществляется отладка алгоритма ПЛИС, и устранение всех явных и неявных ошибок, возникающих во время программирования, адаптации математического аппарата для архитектуры ПЛИС в режиме моделирования. При этом следует учесть, что во время моделирования, не идет симуляция работы ПЛИС. Т.е. параллельные операции будут выполняться независимо, если на процессоре достаточно свободных потоков для осуществления параллелизма. Иначе, процедуры будут выполняться стандартным образом – последовательно. Для наиболее точного определения места возникающих ошибок следует использовать режим анимации, для наблюдения непосредственного следования потоков данных, и анализу получающихся на каждом шаге математических операций значений.

Предварительный расчет скорости работы алгоритма

Предварительный расчет скорости работы алгоритма производится анализом графа соединений математических операций (анализ маршрутов потоков данных, на основе предварительно разработанной программы в LabVIEW), где вершины графа – математические операции, а значения дуг, исходящих из этих вершин – время работы математической операции

Алгоритм нахождения времени выполнения алгоритма описывается следующей рекуррентной формулой [4]:

$$d(1) = 0,$$

$$d(j) = \max_{(i,\,j)} \left(d(i) + c(i,\,j)\right), \qquad (13)$$

где $d(j)$ - время выполнения алгоритма от начальной точки до текущей математической операции, $d(i)$ - время выполнения алгоритма до вершины i, $c(i,\,j)$ - время выполнения алгоритма от вершины i до вершины j, $j = 2,\ldots, n+1$, n - число математических операций.

Предварительная оценка аппаратной емкости алгоритма

Для предварительной оценки емкости алгоритма необходимо суммировать емкость всех математических операций. Математические операции, которые расположены в цикле, необходимо учитывать единожды.

Данное оценивание необходимо, при определении емкости ПЛИС, требующейся для реализации данного алгоритма.

Компиляция программы ПЛИС и запуск на целевом устройстве

Для генерации исполнительного файла ПЛИС необходимо настроить параметры компилятора Xilinx. В LabVIEW, для этого, необходимо создать в дереве проектов файл спецификации к программе ПЛИС. В качестве основного критерия оптимизации компиляции доступен один из вариантов – по быстродействию или сложности структуры ПЛИС. По умолчанию выбран вариант оптимизации по быстродействию.

Таким образом, улучшение производительности может привести к неэкономному расходованию ресурсов ПЛИС, но вышеперечисленные приемы проектирования позволяют улучшить качество программного кода по обоим критериям.

После создания исполнительного файла ПЛИС, его необходимо загрузить на целевое устройство посредством контроллера реального времени или ПК.

Проверка программы ПЛИС на соответствие требованиям

Для проверки программы ПЛИС на соответствие требованиям, необходимо разработать методику проверки на соответствие требованиям. В каждом конкретном случае подобная методика – уникальна, однако есть несколько типовых параметров, которые необходимо оценивать достаточно часто:

1) Скорость выполнения программы расчета;
2) Фазовый сдвиг между входным и выходным сигналами в заданном частотном диапазоне;
3) Амплитудные искажения сигнала;
4) Емкость алгоритма по затрачиваемым ресурсам ПЛИС.

Корректировка программы ПЛИС

a) Увеличение производительности программы
   a. Уменьшение длины путей прохождения данных в блок-диаграмме;
   b. Компоновка внутри параллельных циклов функций, не замедляющих выполнение друг друга;
   c. Использование реентерабельных функций и узлов;
   d. Упаковка данных для пересылок между циклами и между программой ПЛИС программой хоста;
   e. Использование синхросигналов повышенной частоты для структур Timed Loop.
b) Уменьшение ресурсоемкости программы
   a. Аккуратное применение ресурсоемких функций и возможная их замена;
   b. Минимизация массивов и объектов лицевой панели;
   c. Использование структур циклов Single Cycle Timed Loop;
   d. Использование циклов для последовательного выполнения однотипных операций.

Применение данной методики, при разработке имитатора аэродинамических воздействий позволило увеличить скорость работы алгоритма, для восьми взаимосвязанных каналов, с 17,8 кГц до 416,6 кГц, что соответствует увеличению производительности в 24 раза. При получении результатов использовалось оборудование National Instruments: восьми канальный аналого-цифровой преобразователь (АЦП) с разрядностью 16 бит и частотой дискретизации 750кГц и

восьми канальный цифро-аналоговый преобразователь (ЦАП) с разрядностью 16 бит и частотой дискретизации 1МГц.

Библиографический список

1. Карклэ П.Г., Малютин В.А., Мамедов О.С. и другие. О современных методиках наземных испытаний самолетов в аэроупругости. // Труды Центрального Аэрогидродинамического института. 2012. №2708. С. 1-35.
2. Карклэ П.Г., Смыслов В.И. Электромеханическое моделирование в задачах аэроупругости // Полет. 2008. №10. С.25-31.
3. Баран Е. Д. LabVIEW FPGA. Реконфигурируемые измерительные и управляющие системы. М.: Издательство ДМК Пресс, 2009. – 448 с.
4. Писарук Н. Н. Кратчайшие пути в графах. URL: http://pisaruk.narod.ru/slides/shortPath.pdf (дата обращения 29.09.14).