Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра «Систем автоматики, автоматизированного управления и проектирования»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ С.В. Ченцов

« ____» 06. 2018 г.

БАКАЛАВРСКАЯ РАБОТА

15.03.04 – Автоматизация технологических процессов и производств

РАЗРАБОТКА СХЕМЫ СЛЕЖЕНИЯ ЗА ЗАДЕРЖКАМИ НАВИГАЦИОННОГО СИГНАЛА С ПОМОЩЬЮ CORDIC-ПРОЦЕССОРА

			зав. кафедрой, канд. техн. наук Д.В. Капулин
Руководитель	_____	__.06. 2018 г.	
Выпускник	_____	__.06. 2018 г.	Н.Н. Овчинников
Нормоконтролер	_____	__.06. 2018 г.	Т.А. Грудинова

Красноярск 2018

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка схемы слежения за задержками навигационного сигнала с помощью CORDIC процессора» содержит 73 страницы текстового документа, 3 таблицы, 3 приложения, 24 использованных источников.

Ключевые слова: СХЕМА СЛЕЖЕНИЯ, НАВИГАЦИОННЫЙ СИГНАЛ, CORDIC, КОДОВАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ.

Цель ВКР: Разработка технического решения системы слежения с использованием средств навигации.

Задачи ВКР:

– анализ существующих методов навигационных определений и алгоритмов их реализации;

– разработка структуры системы слежения;

– выбор инструментария для создания системы;

– разработка CORDIC процессора.

В результате выполнения ВКР был проведен анализ существующих вариантов реализации, была разработана структура системы слежения на основе CORDIC процессора, осуществлен выбор инструментария и проведены испытания посредством компьютерного моделирования.

Реализованы:

– генератор псевдослучайной последовательности навигационного сигнала;

– коррелятор навигационных сигналов;

– сумматоры различных архитектур для возможной вариабельности;

– схема слежения за задержкой навигационного сигнала на основе CORDIC-процессора.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методы и средства автоматизации являются неотъемлемой частью технологических процессов и всего производства. При этом нарастает потребность в автоматизации различных технологических объектов, в том числе подвижных. Такие объекты могут перемещаться в достаточно небольшом помещении среди людей, что требует обеспечить необходимость безаварийного такого процесса, а также реализовать достаточно точную систему слежения за местоположением объекта. В таких условиях применение известных методов и средств навигации гражданского назначения с возможным отклонением в 5–15 метров затруднительно.

Задачи автоматизации управления движущимися объектами сводятся к диспетчеризации и мониторингу в тех местах, где это возможно, со сравнительно небольшой точностью (порядка 10 метров). Разработать полностью автономную систему не представляется возможным, но можно сделать шаг навстречу к созданию такой системы посредством внедрения навигации в сферу автоматизации. В данном случае потребуется учесть задержку навигационного сигнала, возникающую за счёт скорости распространения сигнала от источника, также не следует забывать о времени, необходимом сигналу для преодоления препятствий.

Исходя из изложенного, актуальным является разработка схемы слежения за задержкой навигационного сигнала с помощью CORDIC процессора. Такая навигационная технологическая система должна отвечать на вопросы: «Где находится объект?» и «Куда движется объект?». CORDIC-процессор в данном случае позволит вычислять сложные функции посредством простых операций суммирования и сдвига. Сама система в дальнейшем может быть установлена на технику, например, на погрузчик, который сможет в автоматическом режиме доставлять готовые изделия на склад, а также привозить необходимые запчасти со склада в цех.

Предложенный в данной выпускной квалификационной работе метод позволит:

1) Реализовать систему слежения в виде интегральной схемы с архитектурой класса «Система на кристалле».

2) Выбрать направление оптимизации работы системы слежения (по времени или по ресурсам).

3) Провести декомпозицию систему с целью аппаратной оптимизации ресурсов.

Целью работы является разработка технического решения схемы слежения за задержками навигационного сигнала на основе CORDIC процессора.

Для достижения обозначенной цели следует решить задачи:

1) Выполнить анализ и детализировать требований к разрабатываемой системе.

2) Выполнить анализ существующих методов и алгоритмов решения навигационной задачи, провести сравнение с требованиями.

3) Выбрать инструментарий проектирования.

4) Провести проектирование структуры и разработать систему слежения в целом и отдельных её частей.

5) Провести компьютерное моделирование разработанной системы слежения с целью подтверждения корректности и адекватности применения выбранных проектных решений.

1 Анализ предметной области

1.1 Общие сведения и задачи определения местоположения подвижных объектов

Спутниковые радионавигационные системы стали важной и неотъемлемой частью деятельности человека также, как и системы спутниковой связи. Спутниковая навигация и координация во времени стали использоваться практически во всех сферах жизни будь то различные технические системы, наука, образование, экономика, производство т. д. Главной задачей при разработке современных навигационных устройств является создание аппаратных вычислительных блоков, способных работать на достаточно больших скоростях 100–200 МГц для обеспечения высокой точности определения местонахождения объекта. Основной проблемой при этом является закрытость таких систем, так как данная аппаратура ориентирована на специализированное применение. К другим недостаткам можно отнести стоимость разработки как самой аппаратуры, так и требуемого программного обеспечения. Но при этом данные системы позволяют отслеживать объект с точностью до нескольких миллиметров.

Реализация системы слежения, предложенная в данной работе, может быть представлена на архитектуре класса «Система на кристалле» что позволит выполнять её на заказных платах. Также в предложенном методе рассматривается аппаратная оптимизация и ее вариация в двух направлениях: скорость расчёта или требуемый объём памяти.

Для работы на скоростях более 100 МГц необходима реализация на программируемых логических схемах конвейерной архитектуры, которая имеет достаточно большие расходы на выравнивание задержек вычислительных блоков. Но её использование способствует:

– кодированию/декодированию сигналов в реальном времени;

– понижению/повышению рабочей частоты сигнала;

– обработке широкополосных сигналов.

Для разработки навигационной аппаратуры потребителя потребуются такие составные блоки обработки как:

1) Схема преобразования частоты.

2) Коррелятор псевдослучайной последовательности.

3) Схема слежения за сигналом.

Конвейерный сумматор, как и конвейерный умножитель, составляют основу для схем преобразования частот. Они способны работать на частоте свыше 100 МГц и выполнять вычисления за 1 такт. Также умножитель применяется для корреляционной обработки навигационного сигнала на промежуточной частоте. После понижения частоты можно будет использовать умножители, работающие на частотах менее 10 МГц. В этом случае будет доступна возможность дополнительной аппаратной автоматизации. Современные программируемые логические интегральные схемы имеют множество DSP (англ. Digital Signal Processing) слоёв, которые также используются для составления вычислительных блоков. Другой важной составляющей данной аппаратуры является цифровой генератор литерных частот. Он нужен для понижения несущей частоты сигнала. Такой генератор можно построить на основе параллельной Flash-памяти и CORDIC процессора.

Генератор литерных частот (ГЛЧ) на основе параллельной Flash-памяти является самым простым схемным решением, но так как он требует предварительного сохранения всех выборок рабочих литерных частот, то для его реализации на один частный канал от 100 до 200 МГц требуется внешняя микросхема Flash-памяти. Кроме этого, такой ГЛЧ может работать только на фиксированных частотах для которых были сохранены выборки.

Способ реализации ГЛЧ на основе CORDIC процессора, работающем в режиме вращения, идентичен конвейерному умножителю по аппаратным

ресурсам, позволяет получить составляющие гармонического сигнала SIN и COS. Сам CORDIC работает по системе уравнений:

$$\begin{cases} x^{i+1} = x^{i} - d_i y^{i} 2^{-i}; \\ y^{i+1} = y^{i} - d_i x^{i} 2^{-i}; \\ z^{i+1} = z^{i} - d_i \tan^{i} 2^{-i}, \end{cases} \tag{1.1}$$

где $x = \cos(z)$;

$y = \sin(z)$;

z – целевой угол;

$d \in \{-1,1\}$.

Такой способ создаёт псевдовращение, и вектор частот после вращения увеличивается в K раз (K=1.64676). Чтобы компенсировать данный эффект, длина первоначального вектора выбирается как $1/K$ (0.60725). Также реализовав CORDIC на конвейерной архитектуре можно получать значения $\sin(z)$ и $\cos(z)$ всего за один такт, работая при этом на частотах свыше 100МГц [19].

К несомненным достоинствам создания ГЛЧ с помощью CORDIC процессора является возможность динамического изменения рабочей частоты. Также возможно изменение начальной фазы ГЛЧ для аппаратного выравнивания сигналов поступающих на отдельные участки системы. Но и данная реализация не обошлась без недостатков, основным из которых является достаточно большая стоимость из-за того, что на частотах более 100 МГц есть необходимость в конвейеризации вычислений. Данный недостаток возможно частично устранить, используя слои DSP, которых требуется не более 10.

1.2 Методы решения навигационных задач

Основной навигационной задачей в спутниковой радионавигационной системе (СРНС) является определение пространственных координат объекта, его скорости, а также текущего времени. При решении данной задачи должен быть определён вектор состояния потребителя:

$$П=|x\ y\ z\ t\ V_x\ V_y\ V_z|^{т}, \qquad (1.2)$$

где x, y, z – пространственные координаты объекта;

t – текущее время;

V_x, V_y, V_z – составляющие вектора скорости объекта.

Элементы вектора П недоступны для измерения с помощью радиосредств. У принятого сигнала могут быть измерены отдельные параметры, такие как задержка или доплеровское смещение частоты. Измеряемый навигационный параметр радиосигнала называется радионавигационным, а соответствующий геометрический параметр – навигационным. Следовательно, задержка сигнала т и доплеровское смещение частоты $f_д$ являются радионавигационными параметрами, а соответствующие им дальность до объекта Д и скорость V_p есть навигационные параметры. Связь между данными параметрами можно представить соотношениями:

$$\begin{cases} Д = c\text{т}, \\ V_p = f_д \lambda, \end{cases} \qquad (1.3)$$

где c – скорость света;

λ – длина волны излучаемого навигационного сигнала.

Геометрическое место точек пространства с одинаковым значением навигационного параметра называют поверхностью положения. Пересечение

двух поверхностей положения, каждая из которых соответствует своему навигационному параметру с заданным значением, определяет линию положения – геометрическое место точек пространства. для которых два заданных навигационных параметра имеют одинаковые значения. Местоположение определяется координатами точки пересечения трех поверхностей положения или двух линий положения. В ряде случаев (из-за нелинейности поверхностей положения) две линии положения могут пересекаться в двух или более точках. При этом однозначно найти местоположение можно только, используя дополнительную поверхность положения или иную информацию о местоположении объекта.

Для решения навигационной задачи, т. е. для нахождения вектора потребителя П, используют функциональную связь между навигационными параметрами и компонентами вектора потребителя. Соответствующие функциональные зависимости принято называть навигационными функциями. Конкретный вид навигационных функций обусловлен многими факторами: типом НП, характером движения НС и потребителя, выбранной системой координат и т. д.

Навигационные функции для пространственных координат потребителя можно получить с помощью различных разновидностей дальномерных, разностно-дальномерных, угломерных методов определения координат и их комбинаций. Для записи навигационных функций, включающих составляющие вектора скорости потребителя, используют соответствующие методы определения скорости объекта

1.2.1 Дальномерный метод

Данный метод является наиболее простым, основан на пассивных измерениях расстояния $Д_i$ между i-м навигационным спутником и объектом. При использовании метода дальность является навигационным параметром, а

поверхность положения – сфера с радиусом $Д_i$. Уравнение такой сферы можно описать уравнением:

$$П = \sqrt{|(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2|}, \quad (1.4)$$

где x_i, y_i и z_i – известные координаты i-го навигационного спутника, принимая во внимание время на распространение сигнала;

x, y, z – координаты объекта.

Координаты x, y, z показывают местоположение объекта и определяются как пересечение трёх сфер. Следовательно, для применения дальномерного метода нужно измерить дальности до трёх навигационных спутников [14].

То есть для данного метода навигационная функция является системой из трёх уравнений типа (1.4). Из-за того, что данная система уравнений является нелинейной, возникает проблема неоднозначности определения координат объекта, от которой можно избавиться благодаря известной информации об объекте, такой как его радиальная скорость. Также в уравнении (1.4) подразумевается, что абсолютно все величины должны быть получены в один момент времени t. Но при построении данного метода в СРНС проявляются некоторые особенности. Самое главное – то что дальность $Д_i$ определяется посредством результатов отслеживания задержки $т_i$ сигнала, отправленного от навигационного спутника.

Рассмотрим составляющие вектора $x_i(t) = \{x_i(t), y_i(t), z_i(t)\}$ и $x(t) = \{x(t), y(t), z(t)\}$. Первый – вектор координат i-го навигационного спутника в момент времени t, второй – вектор координат объекта в этот же момент времени. Предположим, что объект и навигационный спутник работают в единой временной шкале. С борта навигационного спутника посылается сигнал, представленный в верхней части рисунка 1.

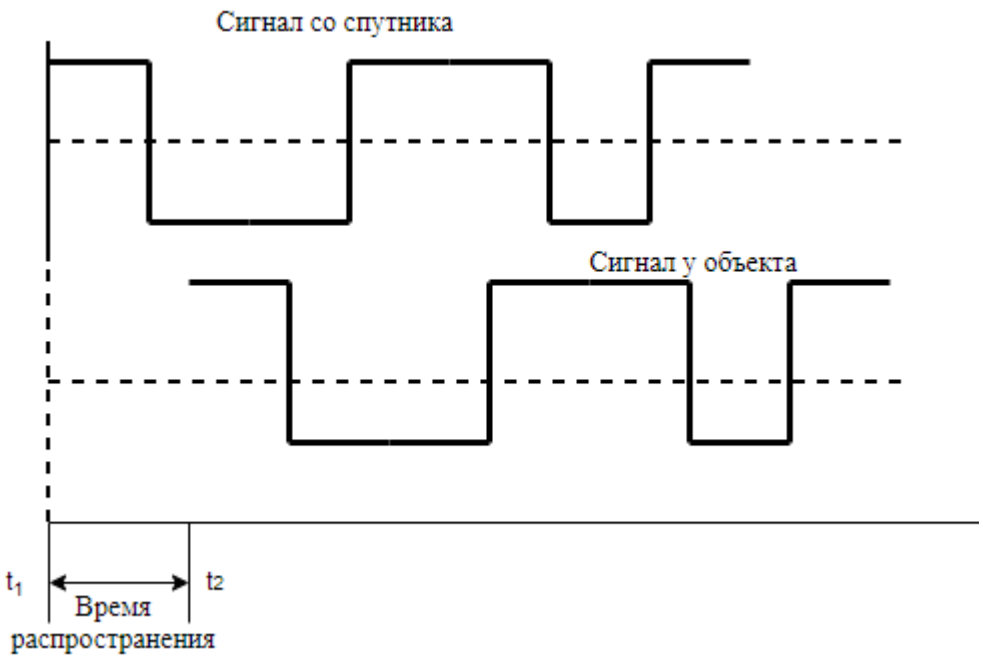

Рисунок 1 – Временная диаграмма излучения
и приёма навигационного сигнала

Данный вид огибающей соответствует дальномерному коду сигналов, используемых в спутниковой радионавигационной системе. Далее вместо огибающей сигнала будем говорить о дальномерном коде. В момент времени t_1 есть некоторая фаза излучаемого кода. В этот момент времени навигационный спутник и объект имеют координаты $x_i(t_1)$ и $x(t_1)$ соответственно. Спустя некоторый промежуток времени, в момент времени t_2 сигнал с фиксированной фазой достигает приёмника объекта, что можно увидеть на нижнем графике рисунка 1. В этот момент времени навигационный спутник и объект имеют координаты $x_i(t_2)$ и $x(t_2)$ соответственно. Задержка в приёмнике измеряется в момент времени t_2 и при известном значении t_1 она будет равна:

$$\tau_i(t_2) = t_2 - t_1 \ . \qquad (1.5)$$

Однако в то же время сигнал, отправленный с навигационного спутника, преодолел расстояние от точки $x_i(t_1)$ до точки $x(t_2)$. Это означает что время распространения сигнала, то есть его задержка, будет равно:

$$\text{т}_i(t_2) = || x_i(t_1) - x(t_2) || / c, \tag{1.6}$$

где ||…|| – эвклидова норма вектора (расстояние между векторами (1.4));

c – скорость света.

Подставляя в получившееся соотношение выражение (1.5) и заменяя t_2 текущем временем, получим формулу для определения задержки сигнала:

$$\text{т}_i(t) = || x_i(t - \text{т}_i(t) - x(t)) || / c. \tag{1.7}$$

Соотношение (1.7) является нелинейным уравнением относительно $\text{т}_i(t)$ [3].

Другой значимой особенностью, которую следует учитывать в спутниковой радионавигационной системе, является привязка времени излучения сигнала с борта навигационного спутника к бортовой шкале времени. При этом объект измеряет задержку сигнала по собственной шкале времени потребителя (ШВП). При наличии расхождения t' шкал времени возникает смещение измеренной дальности относительно истинной, и точность определения местоположения потребителя падает.

Пусть t – системная шкала времени, $t_{\text{нс}}$ – шкала времени навигационного спутника, $t_{\text{п}}$ – шкала времени потребителя (рисунок 2), причем ШВП сдвинута относительно бортовой шкалы времени спутника (БШВ) на t'. Тогда $t_{\text{нс}1}$ – момент излучения фиксированной фазы дальномерного кода с борта НС в БШВ, который соответствует моменту времени $t_{\text{п}1}$ в ШВП; $t_{\text{п}1}$ – момент времени приема той же фазы дальномерного кода приемником потребителя в ШВП.

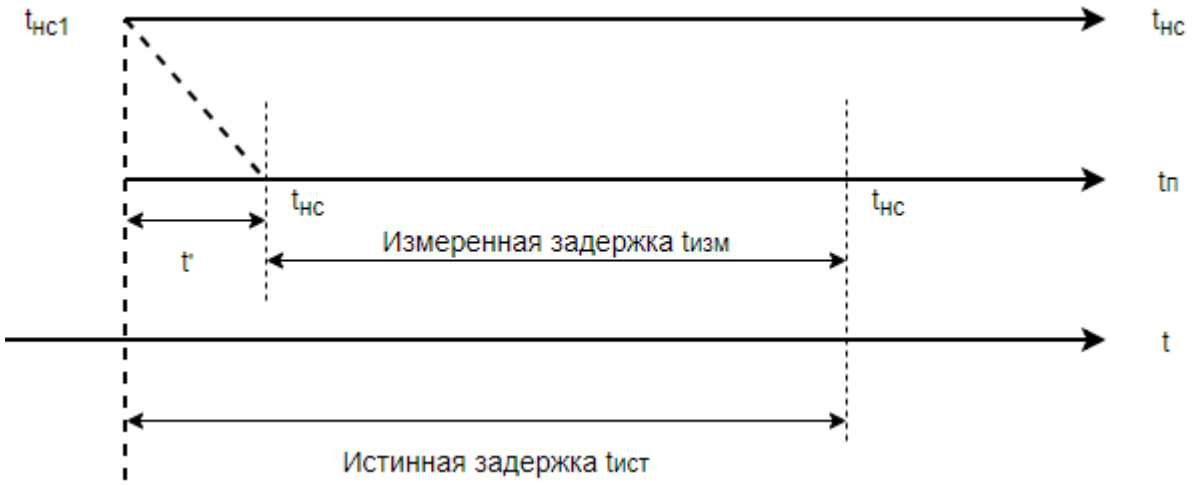

Рисунок 2 – Временная диаграмма излучения и приёма навигационного сигнала со сдвигом шкал времени

Истинное значение задержки сигнала $\tau_{i\text{ист}} = t_{\text{п}2} - t_{\text{сн}1}$, а измеренное объектом:

$$\tau_{i\text{изм}} = t_{\text{п}2} - t_{\text{п}1} = t_{\text{п}2} - (t_{\text{п}1} + t') = \tau_{i\text{ист}} - t'. \quad (1.8)$$

То есть измеренное потребителем значение задержки $\tau_{i\text{изм}}$, называемое псевдозадержкой, отличается от истинного $\tau_{i\text{ист}}$ на величину t', равную смещению шкал времени (БШВ и ШВП). Уменьшить влияние этого фактора можно, установив у потребителя высокостабильный эталон времени (частоты) и периодически проводя его калибровку по БШВ. Однако высокостабильные эталоны времени достаточно дороги и не могут быть использованы у массового потребителя. Другим способом уменьшения влияния расхождения шкал времени является использование псевдо-дальномерного метода определения координат.

1.2.2 Псевдодальномерный метод

Под псевдодальностью от какого-либо навигационного спутника до

объекта принято считать известную в свободном пространстве дальность Д'$_i$ = ct' до этого спутника, различающуюся от действительной дальности Д$_i$ на неизвестную, но неизменяющуюся за время нахождения навигационных параметров величину Д'. То есть уравнение для псевдодальности между объектом и навигационного спутника выглядит следующим образом:

$$Д'_i = Д_i + Д' = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + Д'. \quad (1.9)$$

В спутниковой радионавигационной системе псевдодальность Д'$_i$ = ct'_i вычисляется через время распространения сигнала от навигационного спутника до объекта в пространстве:

$$т'_i = t_{п(ШВП)} - t_{нсi(ШВП)}, \quad (1.10)$$

где $t_{п(ШВП)}$ – время на приёмнике сигнала в момент времени принятия этого сигнала;

$t_{нсi(ШВП)}$ – время излучения сигнала со спутника.

Следовательно,

$$Д'_i = cт'_i, \quad (1.11)$$

где т′ – разница между ШВП и БШВ [20][21].

Из-за того, что псевдодальмерный метод основан на нахождении псевдодальностей, за навигационный параметр следует принимать Д'$_i$. Поверхностью положения до сих пор считалась сфера с центром в точке масс навигационного спутника, однако радиус данной сферы изменён на неизвестную величину Д'. Нахождение псевдодальностей лишь до трёх спутников приводит к системе уравнений с четырьмя неизвестными x, y, z, Д'.

При решении такой системы уравнений получается неопределённый параметр и нужно провести дополнительный расчёт: измерить ещё одну псевдодальность до четвёртого спутника. Такая система из четырёх уравнений позволит найти точное решение, то есть местоположение объекта при использовании данного метода находится как точка пересечения четырёх поверхностей положения.

Псевдодальномерный метод предоставляет крайне жёсткие требования к системе навигационных спутников, так как он требует нахождение в зоне видимости всех четырёх спутников что возможно только в среднеорбитальной спутниковой радионавигационной системе. Использование низкоорбитальных спутниковых радионавигационных систем, как правило, обеспечивает периодическую видимость лишь одного или двух навигационных спутников из-за чего определить положение в данных системах можно, но не в режиме реального времени, так как измерения нескольких линий положения по сигналам одного навигационного спутника проводятся последовательно.

Псевдодальномерный метод не накладывает жёстких ограничений на значение погрешности $Д'_i = c т'_i$ и позволяет одновременно с определением местоположения вычислять отклонение шкалы времени объекта. При распространении сигнала от навигационного спутника до объекта возникают дополнительные неконтролируемые задержки сигнала, которые также могут быть отнесены к составляющей Д'.

1.2.3 Разностно-дальномерный метод

Данный метод основывается на измерениях разности дальностей от объекта до навигационных спутников, одного или сразу нескольких. Он аналогичен псевдодальномерному методу, так как его разумно использовать только в случае фактического измерения псевдодальности. Этот метод использует три разности $\Delta Д_{ij} = Д'_i - Д'_j$, соответствующих расстояниям до трех навигационных спутников. При условии неизменности Д’ разности

псевдодальностей одинаковы с разностями истинных значений, для расчёта которых необходимы всего три независимых уравнения. В данном случае навигационным параметром является $\Delta Д_{ij}$. Поверхности положения находятся благодаря условию $\Delta Д_{ij} = \text{const}$ и являются поверхностями двухполосного гиперболоида. Точность определения координат объекта по данному методу совпадает с точностью псевдодальномерного метода.

1.2.4 Радиально-скоростной (доплеровский) метод

Метод основан на измерении трех радиальных скоростей перемещения потребителя относительно трех навигационных спутников. Физической основой радиально-скоростного метода (РСМ) является зависимость радиальной скорости движения точки относительно НС от координат и относительной скорости НС. Дифференцируя (1.4) по времени, получаем

$$\Pi = \sqrt{|(x_i - x)(x'_i - x') + (y_i - y)(y'_i - y') + (z_i - z)(z'_i - z')|}, \qquad (1.12)$$

где $\{(x'_i - x')(y'_i - y')(z'_i - z')\}$ – характеризуют вектор относительной скорости;

$\{(x_i - x)(y_i - y)(z_i - z)\}$ – относительные координаты потребителя.

Из (1.12) следует, что для определения компонент $\{x', y', z'\}$ вектора скорости необходимо знать: векторы координат $X_i = \{x_i, y_i, z_i\}$ и скорости $M_i = \{x'_i, y'_i, z'_i\}$ навигационных спутников, а также координаты потребителя $\{x, y, z\}$. В результате интегрирования определяются «новые эквивалентные» измерения N_i, соответствующие разностно-дальномерному методу, но при этом разности дальностей формируются для одного и того же навигационного спутника в различные моменты времени [16].

Недостатком метода является невозможность их измерения координат объекта в реальном масштабе времени. Кроме того, в средневысотных

спутниковых радионавигационных системах реализация радиально-скоростного метода затруднена ввиду медленного изменения радиальной скорости. Это обусловило применение разностно-скоростного метода в таких спутниковых радионавигационных системах только для определения составляющих скорости потребителя. Другой возможностью получения информации о координатах $\{x,\ y,\ z\}$ является использование радиально-скоростного метода совместно с одним из известных (описанных выше) «дальномерных» методов.

Недостатком радиально-скоростного метода при определении скорости потребителя является необходимость наличия высокостабильного эталона частоты, так как любая нестабильность частоты приводит к неконтролируемому изменению доплеровского смещения частоты, а, следовательно, к дополнительным ошибкам измерения составляющих скорости потребителя.

1.2.5 Псевдорадиально-скоростной метод

Псевдорадиально-скоростной метод (псевдодоплеровский метод) позволяет определять вектор скорости потребителя в присутствии неизвестного смещения частоты сигнала, например, из-за нестабильности эталона частоты. При наличии такого смещения $\Delta f = Д'_i/\lambda$, а выражение для радиальной скорости можно представить в виде двух слагаемых:

$$Д''_i = Д_i + Д'_i = \frac{(x_i - x)(x'_i - x') + (y_i - y)(y'_i - y') + (z_i - z)(z'_i - z')}{Д_i} + Д'_i. \quad (1.13)$$

Для нахождения вектора скорости потребителя и поправки $Д'_i$, необходимо провести измерения по четырем НС и решить систему четырех уравнений вида (1.12). Для ее решения потребуется знание дальностей $Д'_i$ и координат потребителя. Эта информация может быть получена, например, из

псевдодальномерных измерений [13].

1.2.6 Разностно-радиально-скоростной метод

Сущность данного метода заключается в определении трех разностей $\Delta Д_{ij} = Д'_i - Д'_j$ двух радиальных скоростей навигационных спутников. При этом разности можно вычислять относительно одного или относительно различных спутников. По существу, при вычислении разностей могут использоваться и псевдо-радиальные скорости $Д''_i$, так как при таком вычитании компенсируется неизвестное смещение $Д'_i$ (в предположении, что это смещение одинаковое для различных спутников). Навигационные параметры находятся по формуле:

$$\Delta Д'_i = \frac{(x_i - x)(x'_i - x') + (y_i - y)(y'_i - y') + (z_i - z)(z'_i - z')}{Д_i} - \\ - \frac{(x_j - x)(x'_j - x') + (y_j - y)(y'_j - y') + (z_j - z)(z'_j - z')}{Д_j}. \qquad (1.14)$$

Поверхности положения представляют собой поверхности тела вращения, фокусами которых являются координаты центров масс *i*-го навигационного спутника [15].

Как и для «дальномерных» методов, точность определения составляющих вектора скорости в разностно-радиально-скоростном методе совпадает с точностью определения тех же составляющих в псевдорадиально-скоростном методе. Достоинством разностно-радиально-скоростного метода является его нечувствительность к нестабильностям эталонов частоты и другим неконтролируемым смещениям частоты, а его недостатком – невозможность оценки нестабильности эталонов частоты.

1.2.7 Комбинированные методы

Помимо перечисленных основных методов определения компонент вектора состояния потребителя П возможны комбинированные методы, использующие кроме СРНС дополнительные измерители координат, имеющиеся у потребителя. Так, в дальномерном методе при наличии у потребителя измерителя высоты Н можно вместо измерений трех дальностей до навигационных спутников ограничиться измерением двух дальностей. В этом случае навигационная функция будет включать два уравнения вида (1.4), а третье необходимое уравнение дает измеритель высоты:

$$(R_З + H)^2 = x^2 + y^2 + z^2 . \qquad (1.15)$$

Другой вариант использования комбинированных методов заключается в замене совокупности одновременных измерений на комбинацию одновременных и последовательных измерений или на совокупность только последовательных измерений, например, определение координат потребителя радиально-скоростным методом (1.11). В качестве другого примера можно привести псевдодальномерный метод, который можно реализовать, заменив четыре одновременных измерения по четырем НС на два последовательных измерения по двум НС или на четыре последовательных измерений до одного НС. Аналогичные комбинации возможны и для других методов [3].

2 Алгоритмы первичной обработки сигналов

2.1 Алгоритм поиска и обнаружения

Навигационная аппаратура подвижного объекта имеет многоканальный коррелятор, то есть содержит более одного канала обработки сигналов с видимых навигационных спутников, поэтому поиск сигналов со спутников может быть параллельным. Поиск сигнала для каждого навигационного спутника является последовательным просмотром возможных значений задержек и доплеровских значений частоты сигнала. В режиме поиска используются квадратурные составляющие, а задача обнаружения сигнала в элементарной ячейке поиска решается в соответствии с алгоритмом:

$$\sqrt{I'^2_p + Q'^2_p} \geq h, \tag{2.1}$$

где h – порог, выбираемый из условия обеспечения заданной вероятности правильного обнаружения [10].

Длина интервала накопления сигнала T_a при исследовании в одной элементарной ячейке напрямую зависит от соотношения силы сигнала к шуму q_c/n_0 и вероятности правильного обнаружения или пропуска сигнала $T_a \approx 1-10$ мс [8].

2.2 Алгоритм работы и схема слежения за фазой сигнала

При когерентном режиме работы аппаратуры объекта происходит слежение за фазой сигнала навигационного спутника. Сама схема слежения за фазой включает в себя фазовой дискриминатор (ФД), сглаживающий фильтр и генератор опорного сигнала. В зависимости от условий работы такой схемы, наилучшими (обеспечивающими требуемые точности слежения) являются разные типы фазовых дискриминаторов [9].

Сводка различных типов фазовых дискриминаторов и их основных характеристик представлена в таблице 1. На рисунке 3 представлена схема слежения за фазой с одним из возможных фазовых дискриминаторов. Такая схема имеет корреляторы синфазной I_p и квадратурной Q_p составляющих, блок дискриминирующей функции, сглаживающий фильтр и блок управляющих сигналов для управляемого цифрового генератора гармонического сигнала. В качестве сглаживающего фильтра в схеме слежения за фазой целесообразно использовать фильтр третьего порядка. Но для слабо динамичных потребителей в схеме слежения за фазой можно использовать и фильтр второго порядках [24].

Таблица 1 – фазовые дискриминаторы и их характеристики

Алгоритм работы (дискриминирующая функция)	Зависимость дискриминационной характеристики от фазовой ошибки	Общие свойства
$-th(I_p)Q_p$	–	Оптимальный фазовый дискриминатор при произвольных соотношениях сигнала к шуму. Крутизна дискриминационной характеристики пропорциональна амплитуде A. Требует достаточно больших вычислительных затрат.
$-sign(I_p)Q_p$	$\sin(\delta\varphi)$	Близок к оптимальному при большом соотношении сигнала к шуму. Крутизна дискриминационной характеристики пропорциональна амплитуде A. Требует минимальных вычислительных затрат.

Окончание таблицы 1

Алгоритм работы (дискриминирующая функция)	Зависимость дискриминационной характеристики от фазовой ошибки	Общие свойства
$-I_pQ_p$	$\sin(2\delta\varphi)$	Близок к оптимальному при малых значениях соотношения сигнала к шуму. Крутизна дискриминационной характеристики пропорциональна квадрату амплитуды A^2. Средние затраты на вычисление.
$-\mathrm{arctg}(Q_p/I_p)$	$\delta\varphi$	Главный угол арктангенса. Оптимален в случае оценок максимума функции правдоподобия [12]. Используется при произвольных соотношениях сигнала к шуму. Крутизна дискриминационной характеристики не зависит от амплитуды A. Имеет самые большие затраты на вычисление.

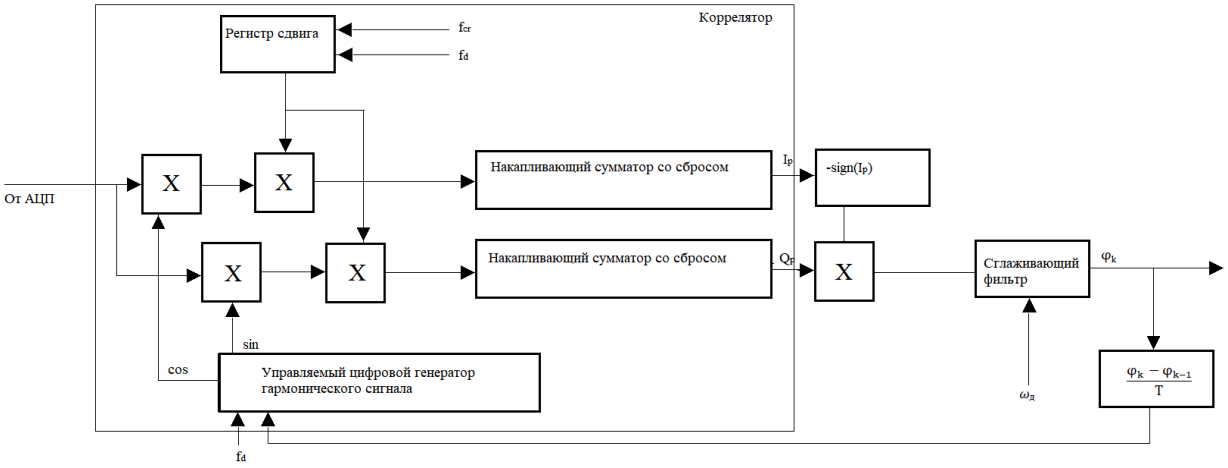

Рисунок 3 – Схема слежения за фазой

Для описания схемы слежения за фазой общего вида введём n-мерный вектор состояния $x_{\varphi,k}$ полагая при этом $\varphi'_k = C_\varphi^{\mathrm{T}} x'_{\varphi,k}$, $C_\varphi = |100...0|^{\mathrm{T}}$. Тогда для дискретной схемы слежения можно записать следующие дискретные уравнения:

$$x'_{\varphi,k} = \tilde{x}_{\varphi,k} + K_\varphi u_{\text{д}\varphi,k}, \tag{2.2}$$

$$\tilde{x}_{\varphi,k} = \Phi_\varphi x'_{\varphi,k-1}, \tag{2.3}$$

где $x'_{\varphi,k}$ – текущая оценка вектора состояния;

$\tilde{x}_{\varphi,k}$ – экстраполированная оценка вектора состояния;

K_φ – вектор-столбец коэффициентов усиления;

Φ_φ – переходная матрица сглаживающего фильтра в контуре следящей системы;

$u_{\text{д}\varphi,k}$ – процесс на выходе ФД.

Так, для схемы, представленной на рисунке 3, имеем:

$$u_{\text{д}\varphi,k} = -\mathrm{sign}(I_{p,k}(x'_{\varphi,k-1}))Q_{p,k}(x'_{\varphi,k-1}). \tag{2.4}$$

Уравнения (2.2), (2.3), рассматриваемые как уравнения линейного фильтра, на вход которого действует процесс $u_{\text{д}\varphi,k}$, описывают сглаживающий фильтр схемы слежения за фазой, который характеризуются переходной матрицей Φ_φ и вектором коэффициентов усиления K_φ. При наличии дополнительной информации о доплеровском смещении частоты принимаемого сигнала (например, от внешних измерительных датчиков) соответствующая оценка $\tilde{\omega}_{\text{д}}$ может быть введена в сглаживающем фильтре

(рисунок 3), что позволяет «снять» часть динамического возмущения, действующего на схему, и сузить ее полосу пропускания. Такая процедура позволяет повысить помехоустойчивость.

2.3 Алгоритм работы и схема слежения задержкой сигнала

Следящая система за задержкой сигнала, как и предыдущая схема слежения за фазой, включает в себя дискриминатор, фильтр и генератор опорного сигнала. Для формирования дискриминаторов схемы слежения за задержкой часто используют опережающие и запаздывающие квадратурные составляющие I_E, Q_E, I_L, Q_L:

$$I_{E(L)i,k} = \sum_{l=1}^{M} y(t_{k-1,l}) G_{\text{дк}}(t_{k-1,l} - (t'_{i(k-1),l} \pm \frac{\Delta t}{2})) \cos(\omega_{\text{пi}} t_{k-1,l} + \varphi'_{i(k-1),l}), \qquad (2.5)$$

$$Q_{E(L)i,k} = \sum_{l=1}^{M} y(t_{k-1,l}) G_{\text{дк}}(t_{k-1,l} - (t'_{i(k-1),l} \pm \frac{\Delta t}{2})) sin(\omega_{\text{пi}} t_{k-1,l} + \varphi'_{i(k-1),l}). \qquad (2.6)$$

Алгоритмы работы дискриминаторов представлены в таблице 2.

Таблица 2 – дискриминаторы схемы слежения за задержкой и их характеристики

Алгоритм работы (дискриминирующая функция)	Общие свойства
$\sqrt{I_E^2 + Q_E^2} - \sqrt{I_L^2 + Q_L^2}$	Близок к оптимальному при некогерентном приёме с большими соотношениями сигнала к шуму. Крутизна дискриминационной характеристики зависит от амплитуды *A*. Имеет хорошие характеристики при модуле ошибки менее 0.5*t*, но также имеет большие затраты на вычисления.

Окончание таблицы 2

Алгоритм работы (дискриминирующая функция)	Общие свойства
$\frac{\sqrt{I_E^2+Q_E^2}-\sqrt{I_L^2+Q_L^2}}{\sqrt{I_E^2+Q_E^2}+\sqrt{I_L^2+Q_L^2}}$	Крутизна дискриминационной характеристики не зависит от амплитуды A. Имеет хорошие характеристики при модуле ошибки менее $0.5t$, имеет очень большие затраты на вычисления.
$(I_E^2+Q_E^2)-(I_L^2+Q_L^2)$	Близок к оптимальному при некогерентном приёме при малых соотношениях сигнала к шуму. Крутизна дискриминационной характеристики зависит от квадрата амплитуды A^2. Имеет хорошие характеристики при модуле ошибки менее $0.5t$, средние затраты на вычисления.
$(I_E-I_L)I_p+(Q_E-Q_L)Q_p$	Близок к оптимальному при некогерентном приёме при малых соотношениях сигнала к шуму. Крутизна дискриминационной характеристики зависит от квадрата амплитуды A^2 . Имеет хорошие характеристики при модуле ошибки менее $0.5t$, минимальные затраты на вычисления.

На рисунке 4 представлена схема слежения за задержкой навигационного сигнала с одним из типов дискриминаторов задержки.

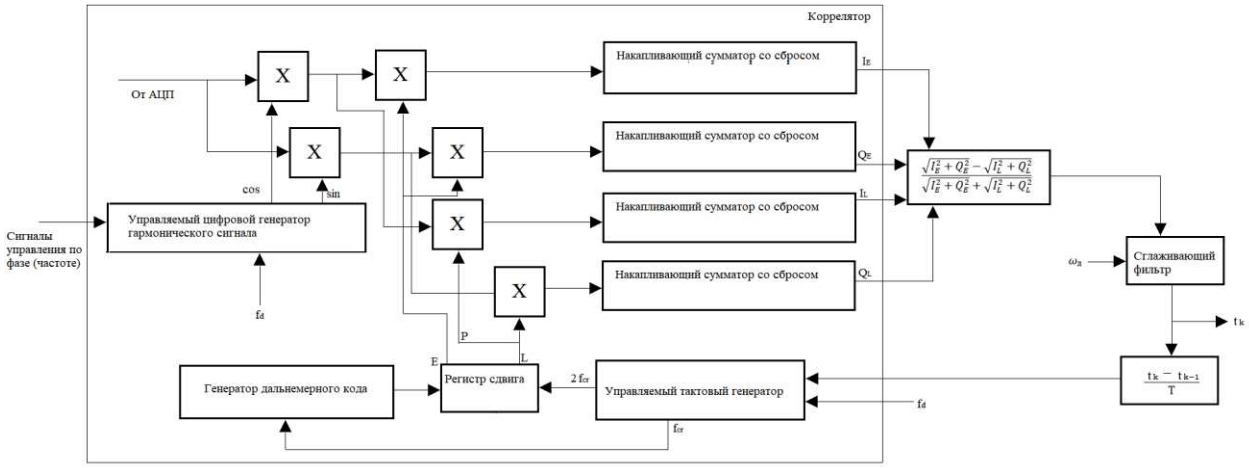

Рисунок 4 – Схема слежения за задержкой

В схеме, приведенной на рисунке 4, управление положением опорного сигнала осуществляется приращением оценки задержки сигнала на соседних тактах работы схемы слежения за задержкой. В качестве сглаживающего фильтра в схемах данного типа часто используют фильтр второго порядка.

Следящая система за задержкой сигнала описывается дискретными уравнениями, аналогичными (3.1), (3.2):

$$x'_{t,k} = \tilde{x}_{t,k} + K_t u_{дt,k}, \tag{2.7}$$

$$\tilde{x}_{t,k} = \Phi_t x'_{t,k-1}. \tag{2.8}$$

При построении комплексных систем слежения за задержкой и доплеровской частотой сигнала в сглаживающем фильтре может вводится оценка доплеровской частоты $\tilde{\omega}_д$, формируемая в схеме слежения за задержкой или в схеме слежения за частотой сигнала.

2.4 Алгоритм работы и схема слежения за частотой сигнала

Система слежения за частотой сигнала, которую часто называют схемой

частотной автоподстройки, используется на промежуточном этапе при переходе из режима поиска сигнала по частоте к режиму непрерывного слежения по фазе и в некогерентном режиме работы навигационной аппаратуры объекта. Данная схема включает частотный дискриминатор, сглаживающий фильтр и перестраиваемый генератор. Частотный дискриминатор можно сформировать из синфазной и квадратурной составляющих I_p, Q_p , сформированных для двух моментов времени t_k и t_{k-1}. Алгоритмы работы частотных дискриминаторов и их краткая характеристика представлены в таблице 3.

Таблица 3 – дискриминаторы схемы слежения за частотой и их характеристики

Алгоритм работы (дискриминирующая функция)	Общие свойства
$I_p(k)Q_p(k-1)-I_p(k-1)Q_p(k)$	Близок к оптимальному при малых соотношениях сигнала к шуму. Крутизна дискриминационной характеристики зависит от квадрата амплитуды A^2. Имеет минимальные вычислительные затраты.
$(I_p(k)Q_p(k-1)-I_p(k-1)Q_p(k))*$ $*\mathrm{sign}(I_p(k)Q_p(k-1)+I_p(k-1)Q_p(k))$	Близок к оптимальному при большом соотношении сигнала к шуму. Крутизна дискриминационной характеристики зависит от квадрата амплитуды A^2. Средние затраты на вычисления
$\frac{I_p(k)Q_p(k-1)-I_p(k-1)Q_p(k)}{I_p(k)Q_p(k-1)+I_p(k-1)Q_p(k)}$	Главный угол арктангенса. Оптимален в смысле максимума функции правдоподобия [18]. Используется при произвольных соотношениях сигнала к шуму. Крутизна дискриминационной характеристики не зависит от амплитуды A. Имеет самые большие затраты на вычисление.

Ширина апертуры дискриминационной характеристики для большинства типов частотных дискриминаторов обратно пропорциональна времени накопления T в корреляторе. Так, для первого из приведенных выше типов Δf = 1/(2T) и для получения ширины апертуры частотного дискриминатора $\Delta f_{ЧД}$ = 500 Гц необходимо выбирать время накопления при формировании квадратурных составляющих T = –1 мс.

Схема слежения за частотой с одним из типов частотных дискриминаторов приведена на рисунке 5.

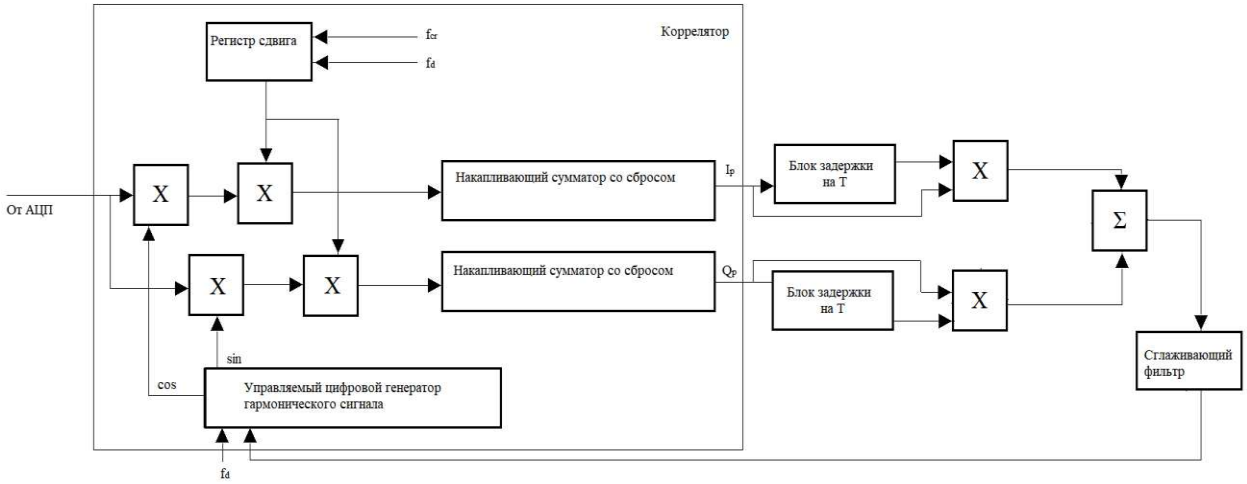

Рисунок 5 – Схема слежения за частотой

Дискретные уравнения, описывающие ССЧ в пространстве состояний, имеют вид:

$$x'_{\omega,k} = \tilde{x}_{\omega,k} + K_{\omega} u_{д\omega,k}, \tag{2.9}$$

$$\tilde{x}_{\omega,k} = \Phi_{\omega} x'_{\omega,k-1}. \tag{2.10}$$

В данной схеме в качестве сглаживающего фильтра обычно используют фильтр второго порядка. В установившемся режиме схема слежения за

частотой обеспечивает ошибку измерения доплеровского смещения частоты менее 50 Гц, что позволяет системе, следящей за фазой захватить сигнал и перейти на устойчивое слежение за фазой сигнала.

3 Проектирование схемы слежения

3.1 Принципы построения аппаратуры потребителей

Навигационная аппаратура потребителей предназначена для определения пространственных координат и составляющих вектора скорости объекта, текущего времени, а также иных навигационных параметров путём приёма и обработки радиосигнала с навигационного спутника.

Современная аппаратура потребителей представляет собой аналого-цифровую систему. Общая схема данной аппаратуры представлена на рисунке 6.

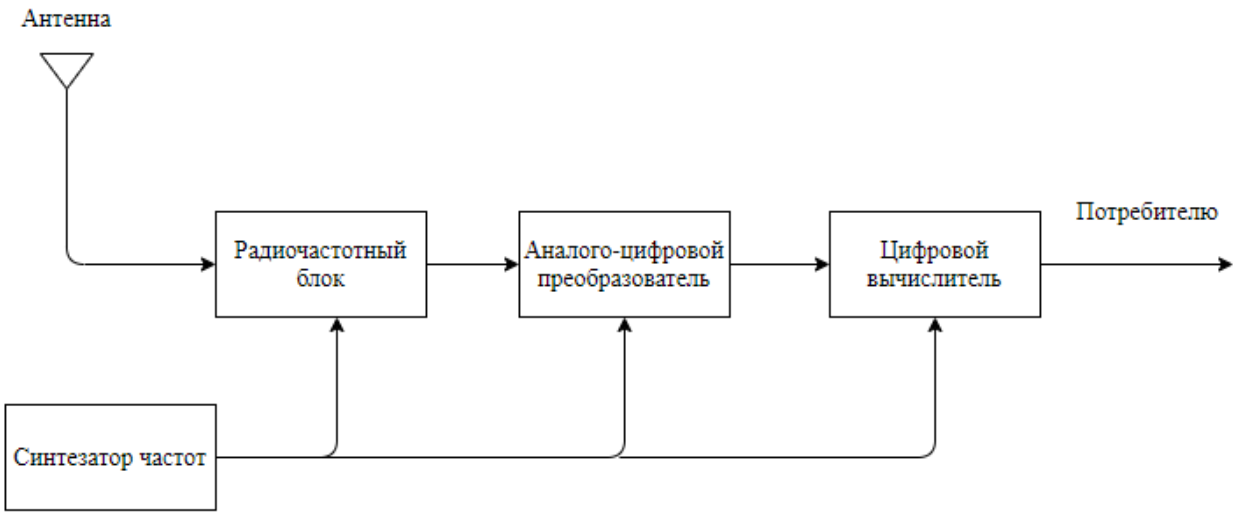

Рисунок 6 – Обобщенная схема навигационной аппаратуры потребителя

Антенна преобразует электромагнитные волны в электрический сигнал. Который передаётся в радиочастотный блок, то есть радиоприёмник, где сигнал усиливается и фильтруется от помех. Далее аналого-цифровой преобразователь (АЦП) трансформирует аналоговый сигнал с радиоприёмника в цифровой и отправляет получившийся сигнал в цифровой вычислитель [4].

Синтезатор частот в свою очередь формирует набор гармонических колебаний, необходимых для работы радиочастотного блока, шкалу времени аппаратуры потребителя и тактовые сигналы синхронизирующие работы АЦП и цифрового вычислителя.

Цифровой вычислитель часто представляют в виде сигнального процессора, который выполняет задачи первичной обработки сигналов:

– формирование опорных сигналов дальномерного кода и управляемых опорных генераторов;

– поиск сигналов по задержке и частоте;

– корреляционную обработку сигналов;

– слежение за дальномерным кодом, частотой сигналов и формирование оценок псевдодальности;

– оценка отношения сигнала к шуму и др.

Реализация цифрового вычислителя представляет жесткую аппаратную часть (многоканальный коррелятор) и программируемый вычислитель. Многоканальный коррелятор как правило представляет собой отдельную микросхему, в которой реализованы все необходимые для работы навигационной аппаратуры потребителя корреляторы (несмещенные, опережающие и запаздывающие), генераторы дальномерных кодов, управляемые опорные генераторы и схемы управления режимами работы коррелятора [17].

В последние годы интенсивно развивается направление, основанное на полностью программной реализации цифрового вычислителя. При этом он может выполняться на программируемых процессорах общего назначения (например, на персональных ЭВМ) или на сигнальных процессорах (Digital Signal Processor (DSP)). Основным достоинством такого подхода является большая гибкость при проектировании новых типов аппаратуры. А недостаток заключается в том, что скорость выполнения расчётов в процессорах общего назначения недостаточно велика что ухудшает параметры точности [11].

3.2 Выбор инструментальных средств проектирования

Для выполнения задачи слежения в реальном времени необходимо обрабатывать большое количество информации. Это требует применения устройств, способных работать на скорости АЦП, то есть на частоте более 100 МГц. Исходя из этого, при разработке подобных устройств применяются программируемые логические интегральные схемы и языки описания аппаратуры. В связи с чем при выполнении поставленной задачи использован язык описания интегральных схем VHDL (англ. VHSIC (Very high speed integrated circuits) Hardware Description Language), который является базовым для разработки аппаратуры современных вычислительных систем [2].

Язык VHDL является фактически международным стандартом в области автоматизации проектирования цифровых систем, это входной язык многих современных систем автоматизированного проектирования как заказных, так и программируемых логических схем, а также программируемых пользователями вентильных матриц. VHDL предназначен, в первую очередь, для спецификации – точного описания проектируемых систем и их моделирования на начальных этапах проектирования: алгоритмическом и логическом. С помощью VHDL можно моделировать электронные схемы с учётом реальных временных задержек. Последние несколько лет довольно успешно разрабатываются системы синтеза схем по спецификациям на данном языке. Так используя систему автоматизированного проектирования Foundation Series 3.1i, можно провести моделирование исходного VHDL-описания схемы, после чего, синтезировав её, получить файл конфигурации микросхемы типа FPGA. Большинство крупнейших фирм производителей программного обеспечения для систем автоматизированного проектирования использую VHDL как основной язык для исходного описания своих проектов.

В целом VHDL представляет собой совокупность лингвистических средств, использование которых позволяет описывать поведение цифровых устройств. Он позволяет описывать параллельные асинхронные процессы

регулярных структур при этом имеет все признаки высокоуровневого языка программирования. В настоящий момент язык развивается, ему посвящаются международные конференции, выходят научные публикации, в которых изучаются проблемы использования на языке VHDL. Он стал языком разработки международных проектов, в том числе осуществляемых с помощью всемирной компьютерной сети Internet. Владение данным языком необходимо для эффективной работы по созданию самой разнообразной электронной аппаратуры на современной элементной базе сверхбольших интегральных схем.

В качестве среды моделирования в представленной работе выбрана система QuestaSim (рисунки 7, 8) компании Mentor Graphics Corporation, представляющая собой комплексную платформу для верификации и отладки сложных проектов, выполненных на ПЛИС и систем, созданных на кристалле.

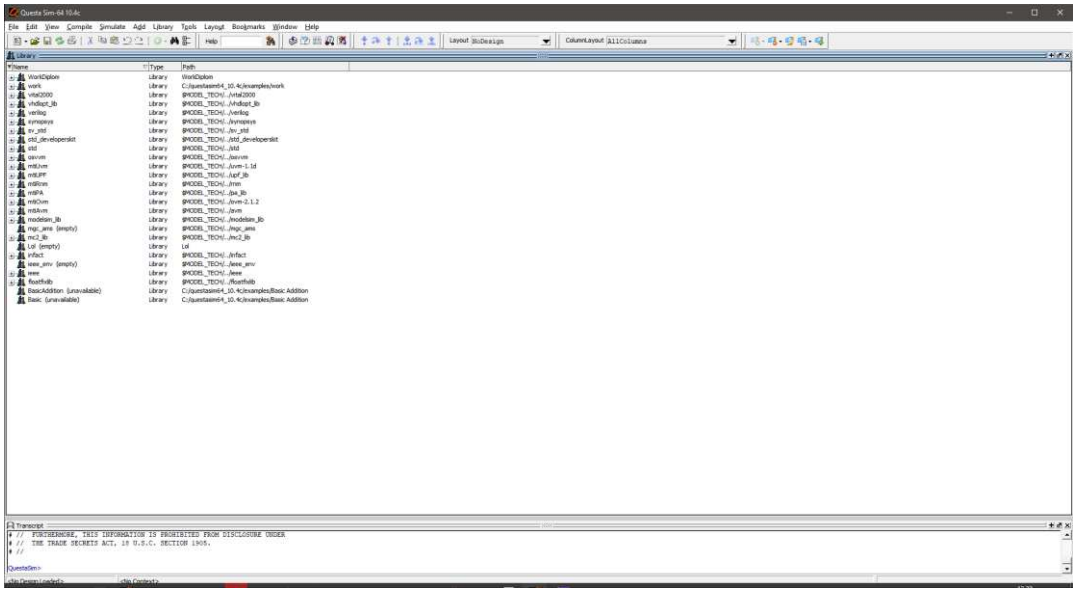

Рисунок 7 – Главное окно программы QuestaSim

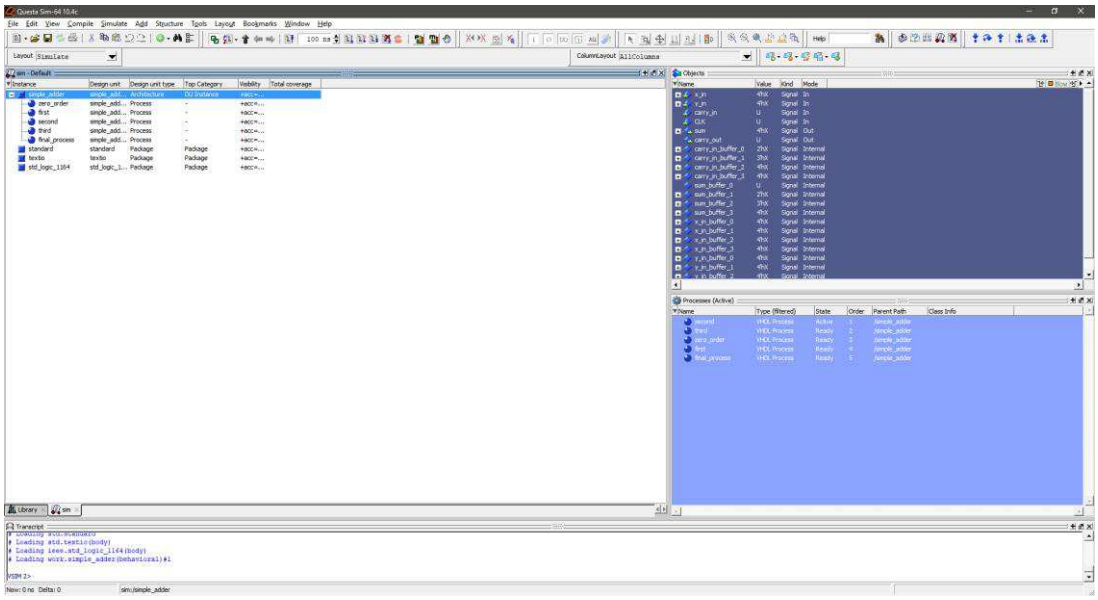

Рисунок 8 – Окно моделирования QuestaSim

Для визуального моделирования отдельных процессов используется программный продукт MATLAB (рисунок 9) от компании MathWorks.

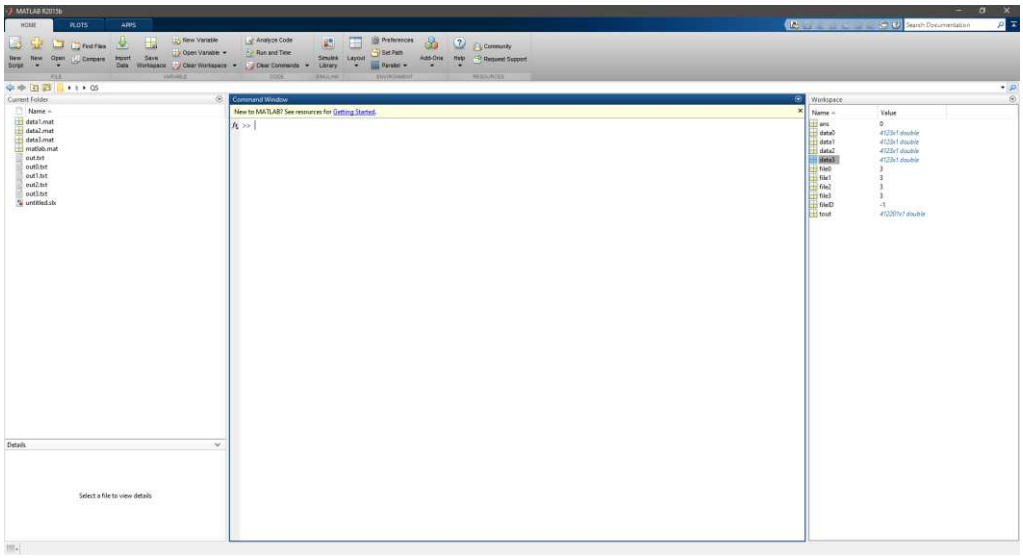

Рисунок 9 – Главное окно программы Matlab

Любой синтезируемый вычислительный алгоритм, описанный на языке VHDL, состоит из суммирования и сдвига регистра. Так как существует три основные архитектуры сумматоров, в далее рассмотрим достоинства и недостатки каждой из них.

3.3 Сумматоры

Однобитовые полусумматоры и полные сумматоры – это универсальные «строительные блоки», которые используются в синтезе многих различных арифметических схем. Полусумматор (HA) получает два входных бита x и y, создавая сумму бит и бит переноса:

$$s = x \text{ XOR } y = x \text{ AND (NOT } y\text{) OR (NOT } x\text{) AND } y, \tag{3.1}$$

$$c = x \text{ AND } y. \tag{3.2}$$

На рисунке 10 изображены три из многих возможных реализаций полусумматоров. Сам полусумматор может рассматриваться как однобитовый двоичный сумматор, который производит сумму его однобитовых входов, а именно: $x + y = (c_{out}s)_{two}$, где знак плюса означает арифметическую сумму [6].

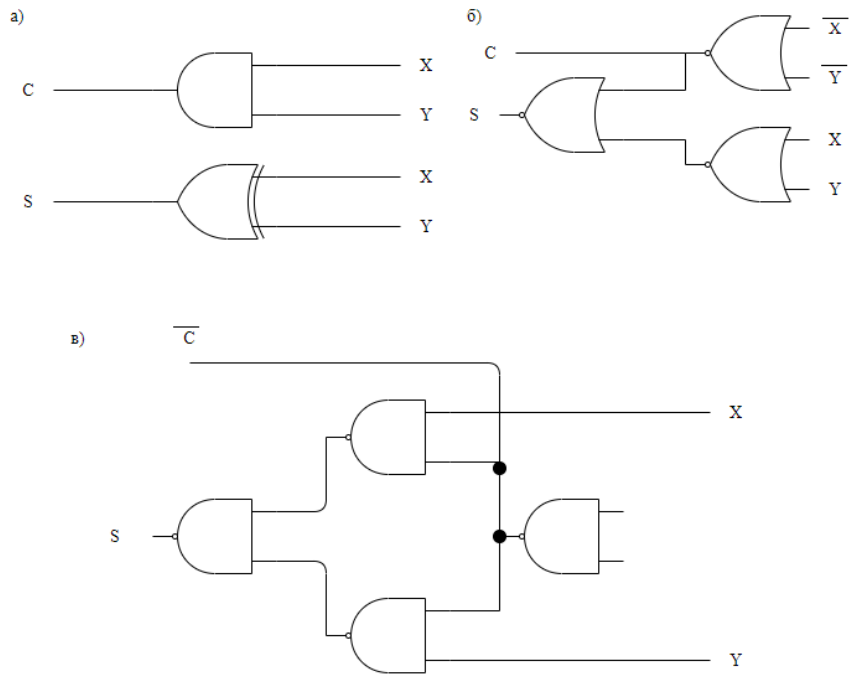

Рисунок 10 – Три реализации полусумматора (а- AND/XOR полусумматор, б-NOR-gate полусумматор, NAND-gate полусумматор с дополненым переносом)

Однако для большинства расчётов используется однобитовый полный сумматор (FA), который в качестве входных значений имеет слагаемые биты x,y и входной бит переноса c_{in}. На выходе также имеется двоичная сумма трёх входных значений и бит переноса в старший разряд c_{out}, которые можно рассчитать используя формулы:

$$s = x \oplus y \oplus c_{in} = xyc_{in} + \overline{xy}c_{in} + \overline{x}y\overline{c}_{in} + x\overline{yc}_{in}, \tag{3.4}$$

$$c_{out} = xy + xc_{in} + yc_{in}. \tag{3.5}$$

Полный сумматор (FA) может быть реализован с использованием двух полусумматоров и логического элемента ИЛИ, как показано на рисунке 11, а. Оператор ИЛИ можно заменить на NAND, если оба полусумматора реализованы по схеме, представленной на рисунке 10, в. В качестве альтернативы можно реализовать полный сумматор как двухуровневую схему AND-OR / NAND-NAND согласно логическим уравнениям (3.4) и (3.5) (рисунок 11, б).

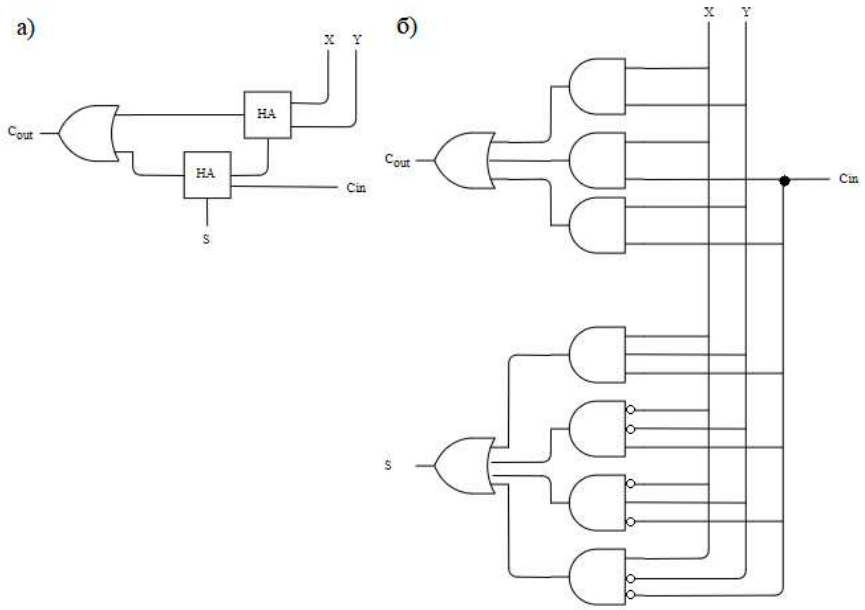

Рисунок 11 – Вомзожные реализцаии полного сумматора (а-основан на полусумматорах, б- двухуровневая схема AND-OR / NAND-NAND)

Из-за важности полного сумматора как арифметического строительного блока, многие оптимизированные схемы FA существуют для различных технологи. На рисунке 12 показан полный сумматор, состоящий из семи инверторов и двух мультиплексоров 4-к-1 (Mux), который подходит для реализации передачи данных CMOS.

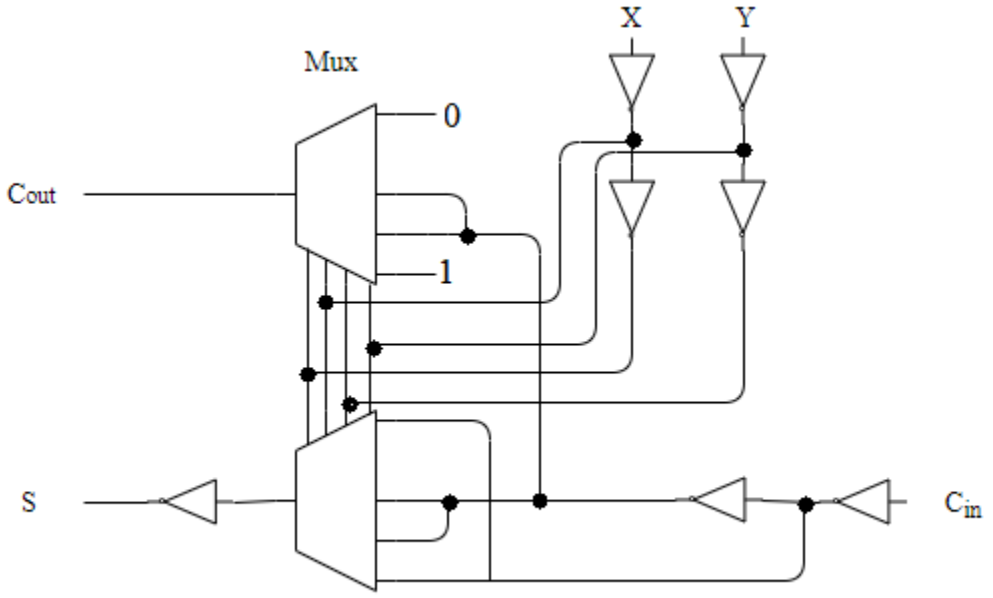

Рисунок 12 – Полный сумматор для передачи данных CMOS

Полные и полусумматоры могут использоваться для реализации множества арифметических функций. Например, последовательный сумматор может быть построен из полного сумматора и регистра переноса [22].

3.3.1 Сумматор с сохранением переноса

Сумматор с сохранением переноса является самой простой в реализации архитектурой. Она основана на последовательном суммировании однобитных чисел с распространением бита переноса на каждый следующий этап. Сумматор с сохранением переноса основан на полном сумматоре (рисунок 13).

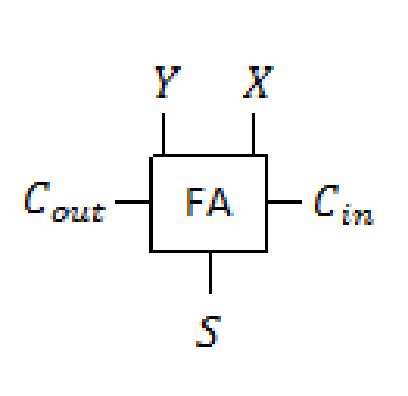

Рисунок 13 – Полный сумматор

Однако не смотря на простоту реализации данного сумматора. Сфера его использования крайне невелика, так как скорость расчёта напрямую зависит от разрядности слагаемых. Так к примеру, для суммирования четырёхбитных чисел потребуется минимум 4 такта времени, а для шестнадцати-битного числа соответственно 16. Схема и результат моделирования четырёхбитного сумматора представлены на рисунках 14 и 15 соответственно.

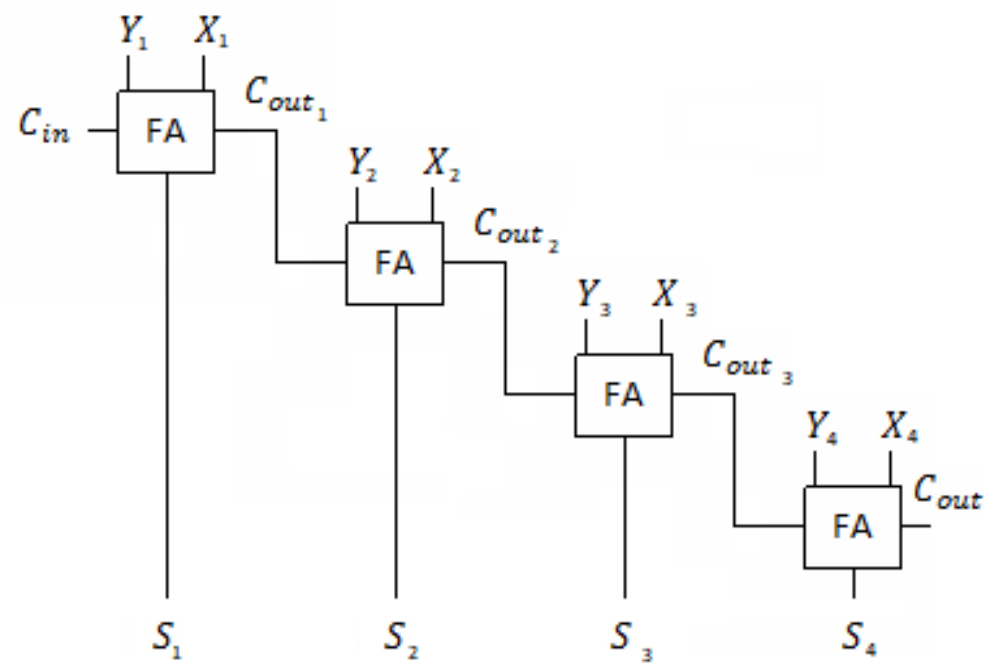

Рисунок 14 – Схема реализации 4-битного сумматора с сохранением переноса

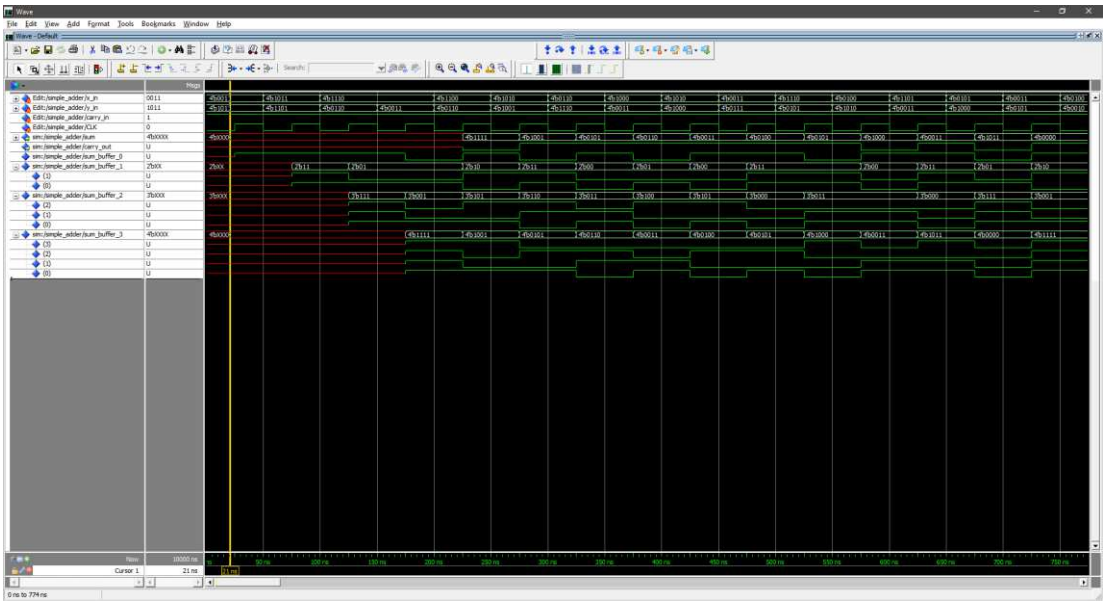

Рисунок 15 – Моделирование конвейерного сумматора с сохранением переноса в QuestaSim

3.3.2 Сумматор с предварительным расчётом переноса

Данная архитектура сумматора является наиболее востребованной в виду того, что распространение переноса рассчитывается по схеме Манчестера. В этом случае очень важными, являются понятия о генерации и распространения переноса. Так как они рассчитываются сразу для группы бит, что позволяет достаточно сильно увеличить скорость вычислений (пропорционально разрядности слагаемых). Предварительный расчёт осуществляется по формулам [3]:

$$\begin{cases} G_i = a_i * b_i; \\ P_i = a_i \oplus b_i; \\ C_1 = G_0 + P_0 * C_0; \\ C_2 = G_1 + G_0 * P_1 + C_0 * P_0 * P_1; \\ C_1 = G_2 + G_1 * P_2 + G_0 * P_1 * P_2 + C_0 * P_0 * P_1 * P_2; \\ C_1 = G_3 + G_2 * P_3 + G_1 * P_2 * P_3 + G_0 * P_1 * P_2 * P_3 + C_0 * P_0 * P_1 * P_2 * P_3, \end{cases} \quad (4.2)$$

где знак «*» соответствует логическому И;

знак «+» соответствует логическому ИЛИ;

знак «⊕» соответствует ИСКЛЮЧАЮЩЕМУ ИЛИ.

Схема сумматора с предварительным расчётом переноса для 4 бит представлена на рисунке 16.

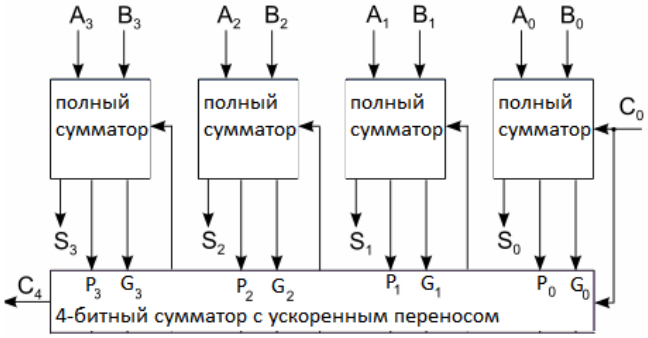

Рисунок 16 – Схема реализации 4-битного сумматора с предварительным расчётом переноса

Также сумматор возможно ускорить, конвейеризовав его работу. На рисунке 17 представлена схема конвейеризации сумматора с предварительным расчётом переноса, а на рисунке 18 – результат моделирования такого сумматора [23].

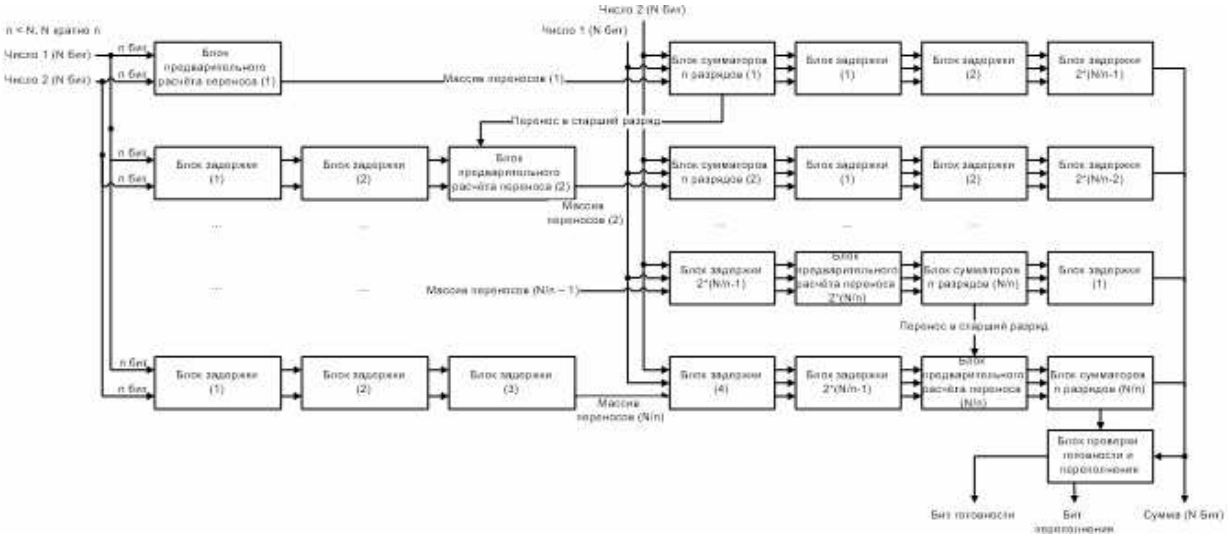

Рисунок 17 – Конвейерный сумматор с предварительным расчётом переноса

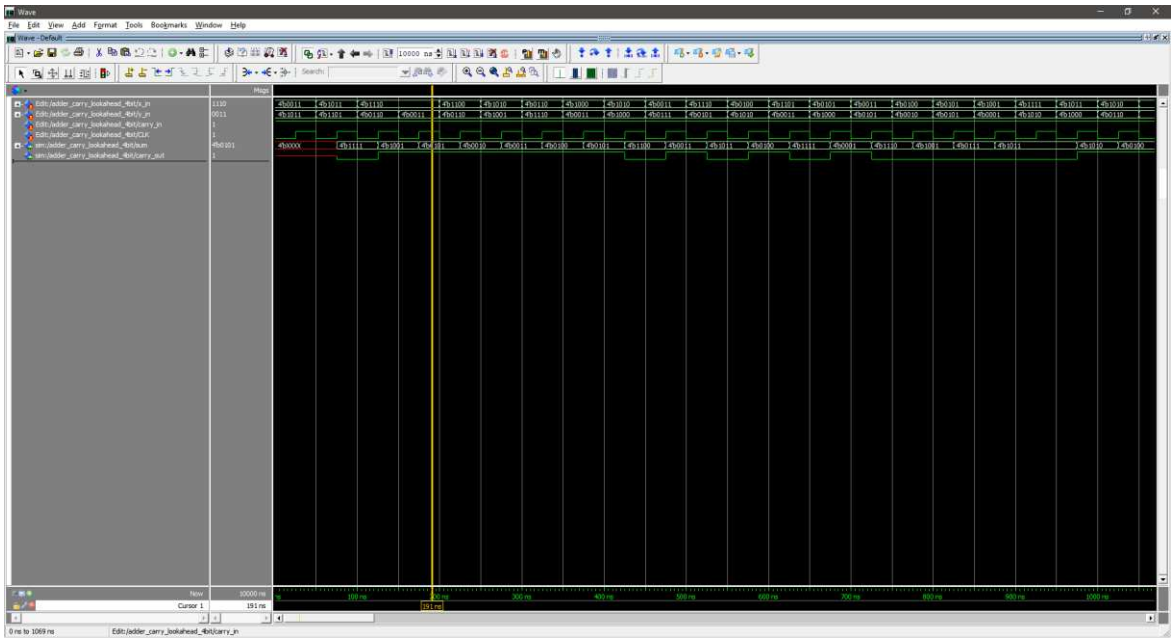

Рисунок 18 – Моделирование конвейерного сумматора с предварительным расчётом переноса в QuestaSim

Соотнеся результаты, полученные на рисунках 15 и 18 можно заметить, что скорость существенно увеличилась. Это произошло из-за того, что биты переноса и распространения вычисляются параллельно сразу для группы из четырёх бит. Что в свою очередь позволяет сделать расчёт суммы для 4-битных чисел всего за 2 такта времени. Но за такую скорость необходимо будет платить так как данная архитектура требует большие аппаратные затраты.

3.3.3 Сумматор с выбором переноса

Данная архитектура подразумевает параллельный расчёт для бит старших разрядов при входных 0 и 1 и выбора результата на основании расчёта текущего разряда.

Такая архитектура показывает себя лучшим образом, если использовать её вместе с другими архитектурами. Так в данном проектном решении для того, чтобы добиться большего быстродействия, будет использоваться

гибридная архитектура сумматора, состоящая из сумматоров с предварительным расчётом и выбором переноса.

Созданный таким методов сумматор на 8 бит будет считать параллельно две пары по четыре бита. Стоит также учитывать, что аппаратные затраты вырастут, так как вторая четвёрка бит рассчитывается сразу для двух значений бита переноса, то есть для 0 и 1. Результат моделирования такой архитектуры сумматора показан на рисунке 19.

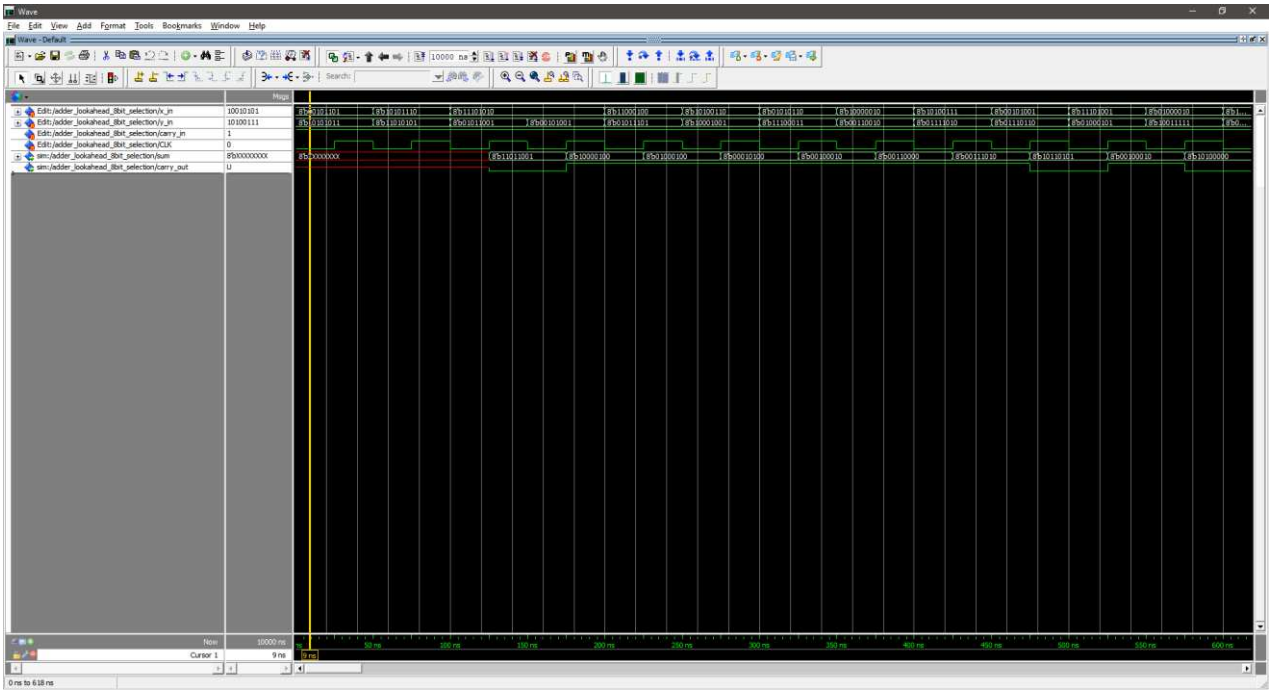

Рисунок 19 – Моделирование гибридного конвейерного сумматора с предварительным расчётом переноса и выбором переноса в QuestaSim

Итог такого суммирования появится на 3 такте времени, хотя при использовании только сумматора с предварительным расчётом переноса потребовалось бы 4 такта. Последующая разница между получением результатов будет только увеличиваться. Так для расчёта 16 бит данной архитектуре потребуется также 3 такта (рисунок 20), когда архитектуре с предварительным расчётом переноса для расчёта требуется 8 тактов.

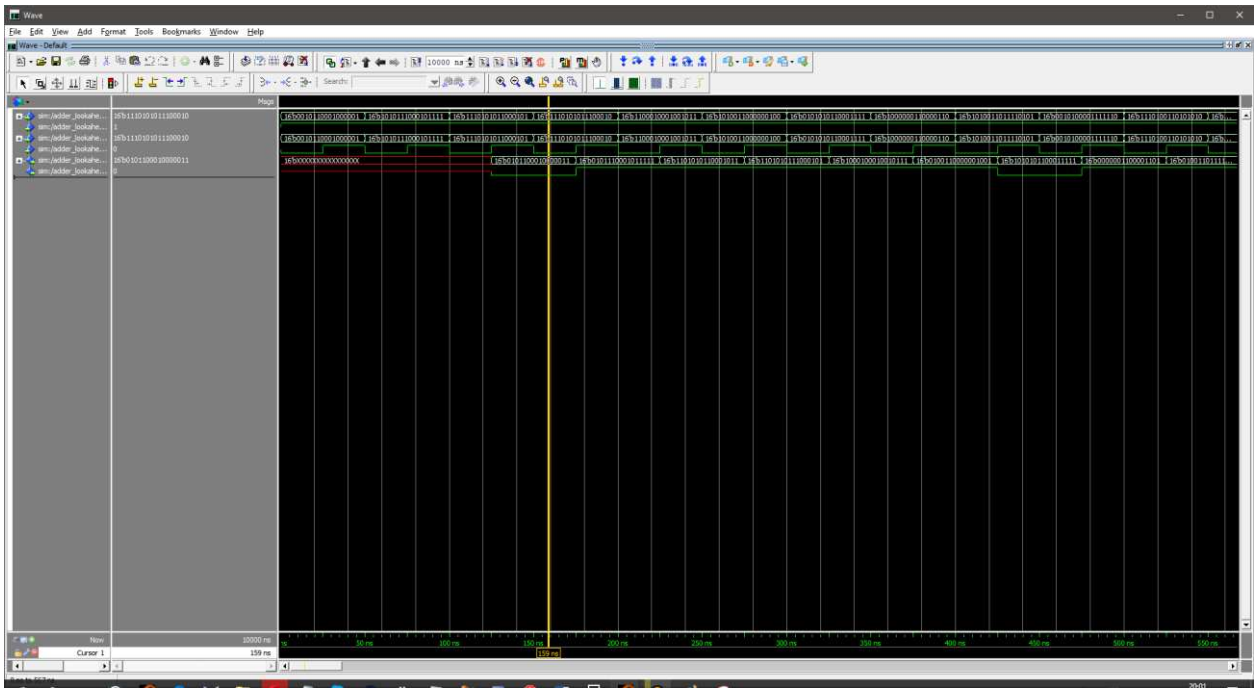

Рисунок 20 – Моделирование гибридного конвейерного сумматора на 16 бит с предварительным расчётом переноса и выбором переноса в QuestaSim

Также можно соединить архитектуру выбора переноса с последовательным переносом. Получившиеся сумматор будет быстрей, но для его реализации на ПЛИС не хватит ресурсов, поэтому на данной ветке развития не будем заострять внимание.

3.4 Коррелятор

Для построения схемы слежения за навигационным сигналом необходим коррелятор, который позволить демодулировать навигационные данные. Сам коррелятор состоит из декодера и сумматора работающего по принципу суммирования «пирамидой» (рисунок 21).

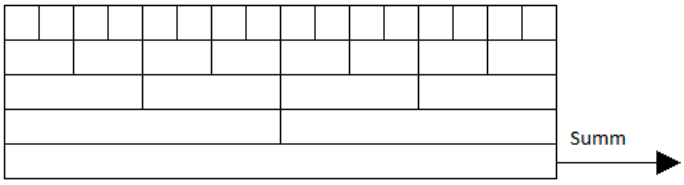

Рисунок 21 – Принцип суммирования «пирамидой»

Для его построения воспользуемся 30-битными сумматорами, имеющими гибридную архитектуру, составленную из сумматоров с предварительным расчётом и выбором переноса. Данная архитектура была выбрана ввиду её быстродействия. Также данный проект поддерживает замену данной архитектуры на любую другую без необходимости иных изменений в структуре или внутри кода программы.

Изначальная ПСП последовательность для ГЛОНАСС составляет 511 значений (чипов). Для увеличения точности каждый чип разделим на 8 точек, то есть продублируем каждое значение 8 раз, итого в целом необходимо просуммировать 4088 значений 1 и −1 что соответствует 12 уровням суммирования. Для того чтобы убедиться, что сумматор считает правильно подадим на него 4088 значений равных 1 и ещё 4088 значений −1 (рисунок 22).

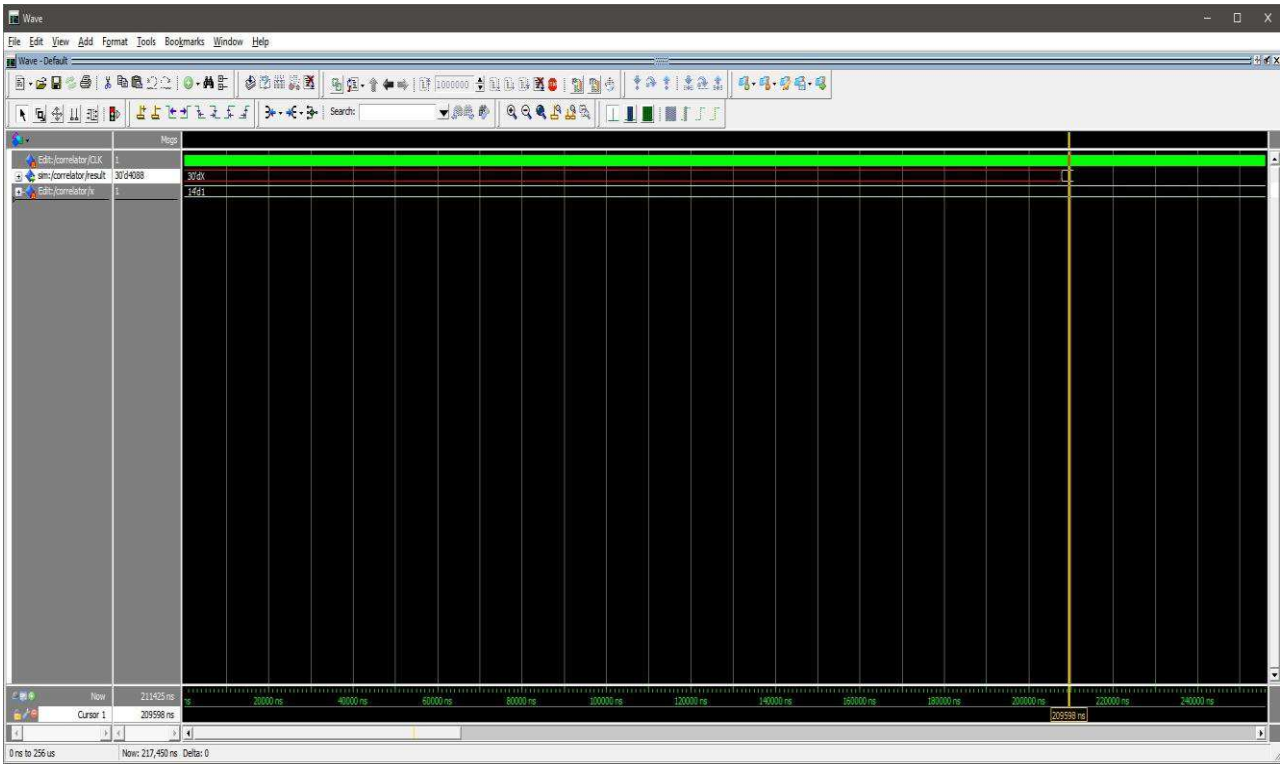

Рисунок 22 – Проверка работоспособности сумматора коррелятора.

Код программы коррелятора представлен в приложении А.

Помимо коррелятора также потребуется генератор м-последовательности.

3.5 Генератор М-последовательности

М-последовательность или последовательность максимальной длины – псевдослучайная двоичная последовательность, порожденная регистром сдвига с линейной обратной связью и имеющая максимальный период. М-последовательности применяются в широкополосных системах связи. Для построения данной последовательности необходимо воспользоваться формулами:

$$\begin{cases} x_0 = x_4 \text{ XOR } x_8; \\ x_{out} = x_7. \end{cases} \tag{4.3}$$

Также следует учесть, что данная последовательность постоянно сдвигается вправо.

Код генератора М-последовательности представлен в приложении Б.

3.6 Схема слежения за задержкой навигационного сигнала с помощью CORDIC-процессора

В целом CORDIC представляет собой итерационный метод сведения прямых вычислений сложных функций к выполнению простых операций сложения и сдвига. Данный метод весьма полезен при вычислении функций на приборах и устройствах с ограниченными ресурсами, например, микроконтроллеры и программируемые логические интегральные схемы. Также в виду того что большинство шагов однотипные, их можно реализовать конвейерной архитектурой при аппаратной реализации.

В данном конкретном случае для слежения за задержкой навигационного сигнала, а именно кодовой последовательности воспользуемся условиями:

$$\begin{cases} \Delta S > 0 \Rightarrow S_{-4} > S_{+4} \Rightarrow \Delta control = ror - X; \\ \Delta S < 0 \Rightarrow S_{-4} < S_{+4} \Rightarrow \Delta control = ror + X; \\ \Delta S = 0 \Rightarrow S_{-4} = S_{+4} \Rightarrow \Delta control = 0. \end{cases} \quad (4.4)$$

Данные условия в отличии от стандартного представления CORDIC, то есть через умножитель, позволят минимизировать затраты ресурсов при этом сохранив скорость расчёта.

Проектное решение на основе CORDIC процессора подразумевает слежение за задержкой кодовой последовательности, так как это позволит минимизировать влияние уровня сигнала, поступающего со спутника.

Для составления схемы слежения за задержкой кодовой последовательности навигационного сигнала необходимо в правильном порядке соединить все составляющее согласно схеме:

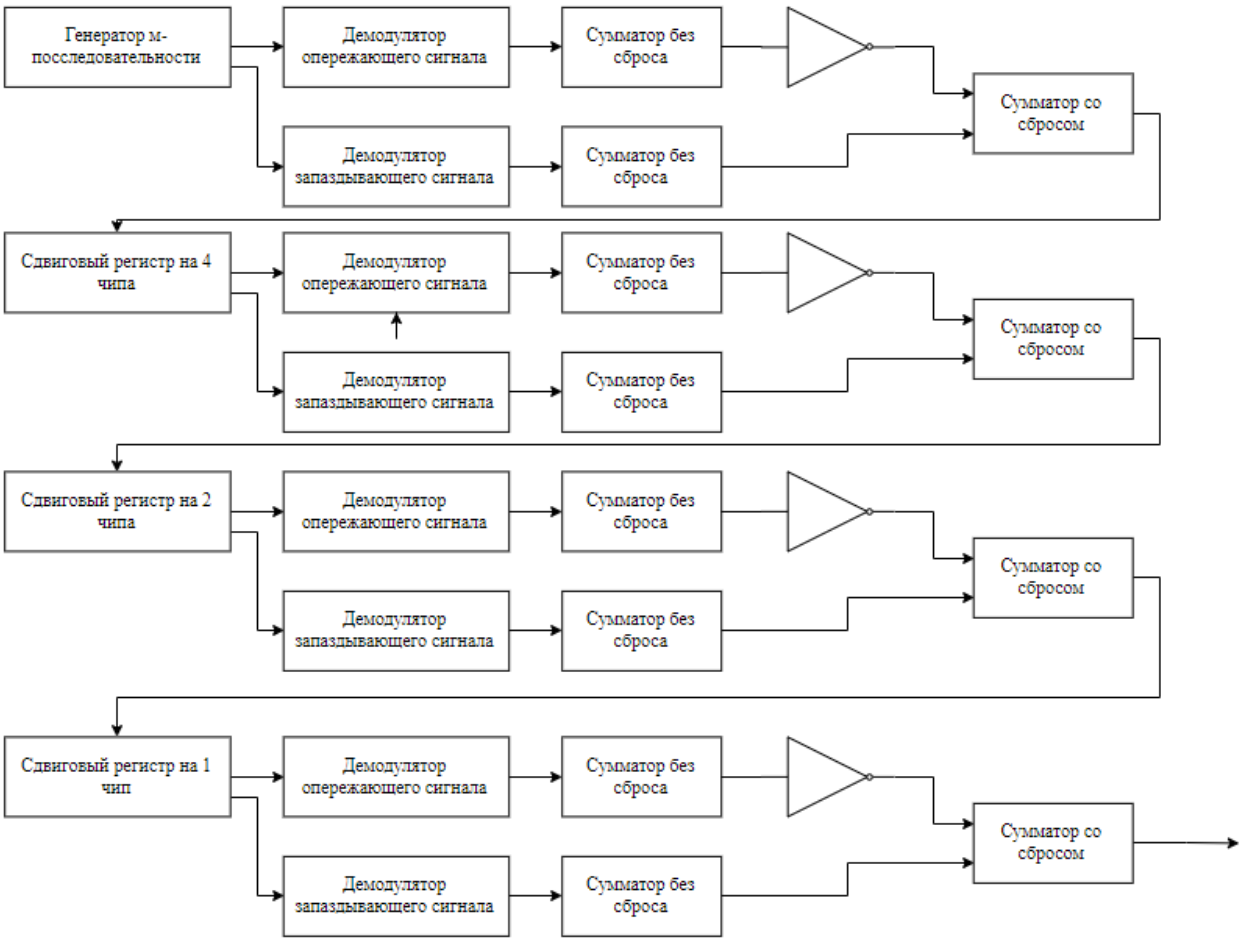

Рисунок 23 – Схема слежения на основе CORDIC-процессора

В данной схеме демодулятор декодирует основную кодовую последовательность относительно сдвинутых, так что если в сдвинутой последовательности стоит 0, то в соответствующем бите не сдвинутой последовательности остаётся тоже самое значение 1 или –1, иначе значения инвертируются. Такая структура основана на процессоре CORDIC только для слежения за кодовой последовательностью [5, 7].

Ввиду того что предложенная в данном проекте схема описана на уровне регистров она может быть реализована на кристалле. Также данное проектное решение позволяет не привязываться к кому-либо производителю, схема может быть реализована как на заказных, так и на устройствах крупной серии.

Основной код программы представлен в приложении В.

Результат работы схемы слежения представлен на рисунке 24, где изначально кодовая последовательность пришла сдвинутой на 2 точки относительно своего истинного значения.

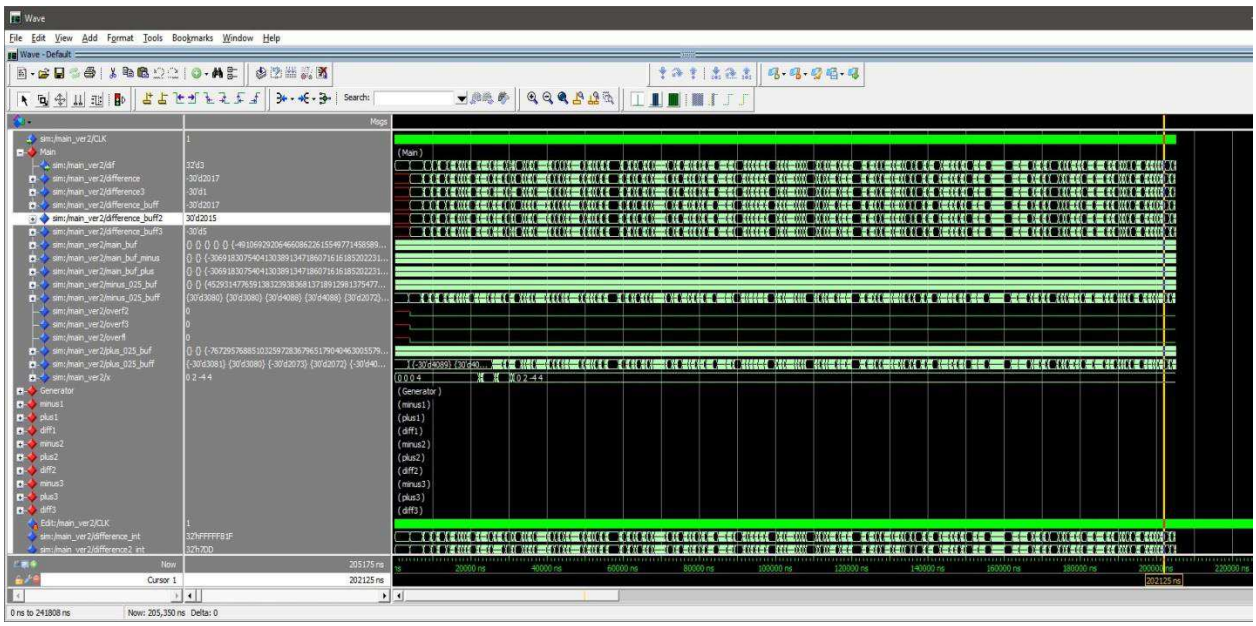

Рисунок 24 – Пример работы схемы слежения

Для большей наглядности сохраним значения сдвиговых регистров в файл и продемонстрируем их на графике MATLAB (рисунок 25).

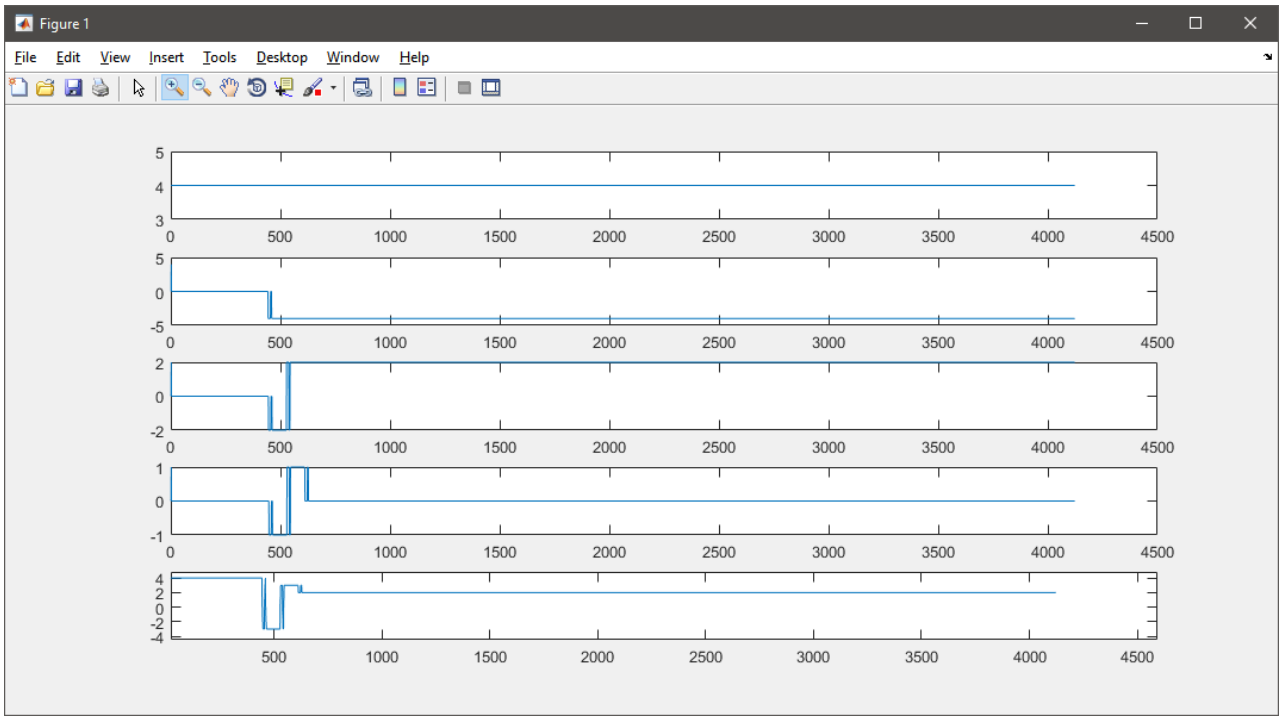

Рисунок 25 – Пример работы схемы слежения

На верхнем графике показано начальное смещение которое является при этом и максимальным значением сдвига. На втором, третьем и четвёртом графиках изображены значения первого, второго и третьего сдвигового регистра. Финальный результат работы схемы слежения продемонстрирован на нижнем графике, на нём можно увидеть, что сигнал синхронизировался, придя в своё истинное значение +2.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы реализована схема слежения за задержкой навигационного сигнала с помощью CORDIC-процессора. При этом выполнен анализ основных известных и обоснование применяемых методов и алгоритмов навигационных определений местоположения, спроектированы различные архитектуры VHDL-описаний технической реализации схемы слежения, проведена их апробация. Моделирование разработанной схемы слежения проведено с использованием программных продуктов MATLAB и ModelSim. Результаты моделирования показали работоспособность выбранных проектных решений и схемы слежения в целом.

Разработанная в ходе выполнения выпускной квалификационной работы схема является универсальной в части ее применения в различных областях: автоматизация позиционирования и управление подвижными объектами, контроль технического состояния и параметров сложных систем автоматизации и т. д. Предложенная схема слежения может быть реализована как на программируемых логических интегральных схемах массового производства, так и на заказных интегральных схемах, специально разработанных для решения навигационных задач. Предлагаемое проектное решение позволит минимизировать затраты по ресурсам или по времени в зависимости от условий задачи путём переключения разных типов архитектур сумматоров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 СТО 4.2 07 2014. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – взамен СТО 4.2 07 2012; дата введ. 03.01.2014. Красноярск, 2013. – 60с.

2 Бибило, П. Н. Основы языка VHDL / П. Н Бибило. – Москва:СОЛОН-Р, 2002. – 224с.

3 Виницкий, А. С. Автономные радиосистемы / А. С. Виницкий. – Москва: Радио и связь. 1986. – 248с.

4 Гришин, Ю. П. Радиотехнические системы / Ю. П. Гришин. – Москва: Высшая школа, 1990. – 496с.

5 Дулевич, В. Е. Теоретические основы радиолокации / В. Е. Дулевич. – Москва: Сов. радио, 1978. – 732с.

6 Жодзишский, М. И. Цифровые радиоприемные системы: Справочник / М. И. Жодзишский. – Москва: Радио и связь, 1990. – 208с.

7 Корн, Г.М. Справочник по математике / Г. М. Корн. – Москва: Наука, 1968. – 832с.

8 Куликов, Е.И. Оценка параметров сигнала на фоне помех / Е. И. Куликов. – Москва: Сов. радио, 1978. – 296с.

9 Первачев, С.В. Адаптивная фильтрация сообщений / С. В. Первачев. – Москва: Радио и связь. 1991. – 160с.

10 Первачев, С.В. Радиоавтоматика / С. В. Первачев. – Москва: Радио и связь, 1982. – 296с.

11 Перов, А.И. Анализ помехоустойчивости системы ФАП приемника сигналов спутниковых радионавигационных систем // Радиотехнические тетради. № 24, 2002. – 124с.

12 Перов, А.И. Статическая теория радиотехнических систем / А. И. Перов – Москва:радиотехника,2003. – 400с.

13 Пестряков, В. Б. Радиотехнические системы / В. Б. Пестряков. –

Москва: Радио и связь, 1985. – 468с.

14 Седж, Э. П. Теория оценивания и ее применение в связи и управлении / Э. П. Седж. – Москва: Связь, 1976. 496с.

15 Сосулин, Ю.Г. Теория обнаружения и оценивания стохастических сигналов / Ю. Г. Сосулин. – Москва: Сов. радио, 1978. – 300с.

16 Харисов, В.Н. Обоснование модели динамики при синтезе схем слежения для приемников СРНС // Радиотехника. Радиосистемы. 2004. – 155с.

17 Харисов, В.И. Оптимальная фильтрация координат подвижного объекта // Известия АН СССР. Радиотехника и электроника, 1984, т. 29, № 10. – 11с.

18 Харисов, В.Н. Статический анализ и синтез радио технических устройств и систем / В. Н. Харисов. –Москва: Радио и связь 1991. – 304с.

19 Шкирятов, В. В. Радионавигационные системы и устройства / В. В. Шкирятов. – Москва: Радио и связь, 1984. – 161с.

20 Ярлыков, М.С. Авиационные радионавигационные устройства и системы / М. С. Ярлыков. – Москва: ВВИА им. Н.Е. Жуковского, 1980. – 384с.

21 Ярлыков, М.С. Статистическая теория радионавигации / М. С. Ярлыков. – Москва: Радио и связь. 1985. – 345с.

22 Behrooz, P. Computer Arithmetic. Algorithms and hardware Designs / P. Behrooz. – New York Oxford: Oxford University Press. 2000. – 510с.

23 Filippov, V. The First Dual-Depth Dual-Frequency Choke Ring // V. Filippov. Nashville. Proceedings of the 11th International Technical Meeting of the Satellite Division of The Institute of Navigation. 1998. – С. 1035-1040.

24 Zhodzishsky, M. Real-Time Kinematic (RTK) Processing for Dual-Frequency GPS/GLONASS // M Zhodzishsky. Nashville. Proceedings of the 11th International Technical Meeting of the Satellite Division of The Institute of Navigation. 1998. – С. 1325-1331.

ПРИЛОЖЕНИЕ А

Программа декодирования и суммирования.

```
-- Decoder + Adder 4088bit
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;

library work;
use work.ALL;

entity correlator_ver2 is
	port( plus_minus025: in signed (4087 downto 0);
	main: in signed (4087 downto 0);
	x_in: in integer;
	CLK: in std_logic;
	result : out std_logic_vector (29 downto 0);
	pl_min025:out signed (4087 downto 0);
	main_out:out signed (4087 downto 0));
end correlator_ver2;


architecture behavioral of correlator_ver2 is

component Adder_lookahead_selection_30bit
port
(x_in : in std_logic_vector (29 downto 0);
	y_in : in std_logic_vector (29 downto 0);
	carry_in : in std_logic;
	CLK : in std_logic;
	sum : out std_logic_vector (29 downto 0);
	carry_out: out std_logic);
end component;

component Adder_lookahead_selection_14bit
port(x_in : in std_logic_vector (13 downto 0);
	y_in : in std_logic_vector (13 downto 0);
	carry_in : in std_logic;
	CLK : in std_logic;
	sum : out std_logic_vector (13 downto 0);
	carry_out: out std_logic);
end component;
```

```
type buff is array (NATURAL range<>) of std_logic_vector (13 downto 0);
type buffe is array (NATURAL range<>) of std_logic_vector (29 downto 0);
signal y: buffe (4087 downto 0);--:= (others =>(others => '0'));
signal res1_buf: std_logic_vector (29 downto 0):= (others =>'0');
signal sum_buff: buffe (2043 downto 0):= (others =>(others => '0'));
signal sum_buff_1: buffe (1021 downto 0):= (others =>(others => '0'));
signal sum_buff_2: buffe (510 downto 0):= (others =>(others => '0'));
signal sum_buff_3: buffe (255 downto 0):= (others =>(others => '0'));
signal sum_buff_4: buffe (127 downto 0):= (others =>(others => '0'));
signal sum_buff_5: buffe (63 downto 0):= (others =>(others => '0'));
signal sum_buff_6: buffe (31 downto 0):= (others =>(others => '0'));
signal sum_buff_7: buffe (15 downto 0):= (others =>(others => '0'));
signal sum_buff_8: buffe (7 downto 0):= (others =>(others => '0'));
signal sum_buff_9: buffe (3 downto 0):= (others =>(others => '0'));
signal sum_buff_10: buffe (1 downto 0):= (others =>(others => '0'));
signal sum_buff_11: std_logic_vector (29 downto 0) := (others => '0');

signal buff_sum: buffe (3 downto 0);

signal overflow: std_logic_vector  (1021 downto 0):= (others => '0');
signal overflow_0: std_logic_vector  (2043 downto 0):= (others => '0');
signal overflow_1: std_logic_vector  (510 downto 0):= (others => '0');
signal overflow_2: std_logic_vector  (255 downto 0):= (others => '0');
signal overflow_3: std_logic_vector  (127 downto 0):= (others => '0');
signal overflow_4: std_logic_vector  (63 downto 0):= (others => '0');
signal overflow_5: std_logic_vector  (31 downto 0):= (others => '0');
signal overflow_6: std_logic_vector (15 downto 0):= (others => '0');
signal overflow_7: std_logic_vector (7 downto 0):= (others => '0');
signal overflow_8: std_logic_vector (3 downto 0):= (others => '0');
signal overflow_9: std_logic_vector (1 downto 0):= (others => '0');
signal overflow_10: std_logic := '0';
signal buff_main: signed (4087 downto 0);
signal buff_plus_minus025: signed (4087 downto 0);

signal lish:std_logic_vector (5 downto 0);

begin

buf_1:process(CLK)
begin
if(rising_edge(CLK)) then
        buff_main <= main ror x_in;
end if;
```

```
end process buf_1;

buf_2:process(CLK)
begin
if(rising_edge(CLK)) then
	buff_plus_minus025 <= plus_minus025;
end if;
end process buf_2;

b1: for i in 4087 downto 0 generate
decoder_1:process(CLK)
begin
if(rising_edge(CLK)) then
--	for i in 4087 downto 0 loop
	if (buff_main(i) = '0') then
		if (buff_plus_minus025(i) = '0') then
		y(i) <= "00000000000000000000000000000001";
		else
		y(i) <= "11111111111111111111111111111111";
		end if;
	else
		if (buff_plus_minus025(i) = '0') then
		y(i) <= "11111111111111111111111111111111";
		else
		y(i) <= "00000000000000000000000000000001";
		end if;
	end if;
--	end loop;
end if;
end process decoder_1;
end generate b1;


pr0:for i in 2043 downto 0 generate
	add_2:Adder_lookahead_selection_30bit
	port map (x_in => y(2*i),
		  y_in => y(2*i+1),
		  carry_in => '0',
		  CLK => CLK,
		  sum => sum_buff(i)(29 downto 0),
		  carry_out => overflow_0(i));
end generate pr0;
```

```
pr1:for i in 1021 downto 0 generate
        add_3:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff(2*i),
--                x_in(29 downto 15) => (others => sum_buff(2*i)(14)),
                  y_in => sum_buff(2*i+1),
--                y_in(29 downto 15) => (others => sum_buff(2*i)(14)),
                  carry_in => '0',
                  CLK => CLK,
                  sum => sum_buff_1(i),
                  carry_out => overflow(i));
end generate pr1;



pr2:for i in 510 downto 0 generate
        add_4:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_1(2*i),
                  y_in => sum_buff_1(2*i+1),
                  carry_in => '0',
                  CLK => CLK,
                  sum => sum_buff_2(i),
                  carry_out => overflow_1(i));
end generate pr2;

p:process(CLK)
begin
if (rising_edge(CLK)) then
        buff_sum(0) <= sum_buff_2(510);
end if;
end process p;


pr:for i in 3 downto 1 generate
process(CLK)
begin
if (rising_edge(CLK)) then
        buff_sum(i) <= buff_sum(i-1);
end if;
end process;
end generate pr;


pr3:for i in 254 downto 0 generate
```

```
        add_5:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_2(2*i),
                  y_in => sum_buff_2(2*i+1),
                  carry_in => '0',
                  CLK => CLK,
                  sum => sum_buff_3(i),
                  carry_out => overflow_2(i));
end generate pr3;

pr4:for i in 126 downto 0 generate
        add_6:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_3(2*i),
                  y_in => sum_buff_3(2*i+1),
                  carry_in => '0',
                  CLK => CLK,
                  sum => sum_buff_4(i),
                  carry_out => overflow_3(i));
end generate pr4;

add_6_1:Adder_lookahead_selection_30bit
port map (x_in => buff_sum(3),
          y_in => sum_buff_3(254),
          carry_in => '0',
          CLK => CLK,
          sum => sum_buff_4(127),
          carry_out => overflow_3(127));



pr5:for i in 63 downto 0 generate
        add_7:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_4(2*i),
                  y_in => sum_buff_4(2*i+1),
                  carry_in => '0',
                  CLK => CLK,
                  sum => sum_buff_5(i),
                  carry_out => overflow_4(i));
end generate pr5;

pr6:for i in 31 downto 0 generate
        add_8:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_5(2*i),
                  y_in => sum_buff_5(2*i+1),
```

```
                carry_in => '0',
                CLK => CLK,
                sum => sum_buff_6(i),
                carry_out => overflow_5(i));
end generate pr6;

pr7:for i in 15 downto 0 generate
        add_9:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_6(2*i),
                y_in => sum_buff_6(2*i+1),
                carry_in => '0',
                CLK => CLK,
                sum => sum_buff_7(i),
                carry_out => overflow_6(i));
end generate pr7;

pr8:for i in 7 downto 0 generate
        add_10:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_7(2*i),
                y_in => sum_buff_7(2*i+1),
                carry_in => '0',
                CLK => CLK,
                sum => sum_buff_8(i),
                carry_out => overflow_7(i));
end generate pr8;

pr9:for i in 3 downto 0 generate
        add_11:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_8(2*i),
                y_in => sum_buff_8(2*i+1),
                carry_in => '0',
                CLK => CLK,
                sum => sum_buff_9(i),
                carry_out => overflow_8(i));
end generate pr9;

pr10:for i in 1 downto 0 generate
        add_12:Adder_lookahead_selection_30bit
        port map (x_in => sum_buff_9(2*i),
                y_in => sum_buff_9(2*i+1),
                carry_in => '0',
                CLK => CLK,
                sum => sum_buff_10(i),
```

```
                    carry_out => overflow_9(i));
end generate pr10;

add_13:Adder_lookahead_selection_30bit
port map (x_in => sum_buff_10(0),
          y_in => sum_buff_10(1),
          carry_in => '0',
          CLK => CLK,
          sum => sum_buff_11,
          carry_out => overflow_10);

result_1:process(CLK)
begin
if (rising_edge(CLK)) then
result <= sum_buff_11;


pl_min025<=buff_plus_minus025;
main_out<=buff_main;
--      result(25 downto 0) <= sum_buff_11(25 downto 0);
--      for i in 29 downto 26 loop
--              result(i) <= sum_buff_11(25);
--      end loop;
end if;
end process result_1;

end behavioral;
```

ПРИЛОЖЕНИЕ Б

Код генератора ПСП-последовательности.

```
--Generator
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;

entity Generator_ver2 is
Port (CLK : in std_logic;
x_in: in integer;
--		out_generator_minus_025: out std_logic_vector (29 downto 0);
--		out_generator_plus_025: out std_logic_vector (29 downto 0)
out_gen_minus_025: out signed (4087 downto 0);
out_gen_plus_025: out signed (4087 downto 0);
main: out signed (4087 downto 0));
end Generator_ver2;

architecture Behavioral of Generator_ver2 is

type buf is array (NATURAL range<>) of std_logic_vector (29 downto 0);
signal glonas: signed (8 downto 0) := "111111111";
signal out_glonas: signed (4087 downto 0);--:=(others => '0');
signal buff_glonas: signed (4087 downto 0);
signal buff_minus_generation: signed (4087 downto 0);
signal buff_plus_generation: signed (4087 downto 0);
signal x:std_logic_vector(2 downto 0) := (others => '0');
signal out1:std_logic;
--signal x_in: integer := 0;

signal buf_out_generator_minus_025: buf (4087 downto 0);
signal buf_out_generator_plus_025: buf (4087 downto 0);

begin

process(CLK)
begin
if(falling_edge(CLK)) then
	if(x="000")then
		x<= "001";
		out1<='0';
	elsif (x="001") then
```

```
                x<="010";
                out1<='0';
        elsif (x="010") then
                x<= "011";
                out1<='0';
        elsif (x="011") then
                x<= "100";
                out1<='0';
        elsif (x="100") then
                x<= "101";
                out1<='0';
        elsif (x="101") then
                x<= "110";
                out1<='0';
        elsif (x="110") then
                x<= "111";
                out1<='0';
        else
                x<="000";
                out1<='1';
        end if;

end if;
end process;




--Create code
glon: process(CLK)
begin
if(rising_edge(CLK)) then
if(out1 = '1') then
        for k in 8 downto 1 loop
        glonas(k) <= glonas(k-1);
        end loop;
glonas(0) <= glonas(4) XOR glonas(8);

--if(out1 = '1') then
--glonas(0) <= glonas(4) XOR glonas(8);
--out_glonas(0)<= glonas(7);
--else
--glonas(0) <= glonas(8);
```

```
end if;
out_glonas(0)<= glonas(7);
end if;
end process glon;

b: for i in 4087 downto 1 generate
generate_glonas:process(CLK)
 begin
if(rising_edge(CLK)) then
        out_glonas(i) <= out_glonas(i-1);
end if;
end process generate_glonas;
end generate b;

process(CLK)
begin
if(rising_edge(CLK)) then
buff_minus_generation <= out_glonas;
end if;
end process;

--Plus and Minus
buf_0:process(CLK)
begin
if(rising_edge(CLK)) then
--      buff_minus_generation <= out_glonas rol 2;
        buff_plus_generation <= out_glonas ror 4;
end if;
end process buf_0;

buf_1:process(CLK)
begin
if(rising_edge(CLK)) then
--      buff_minus_generation <= out_glonas rol 2;
        buff_glonas <= out_glonas ror x_in;
end if;
end process buf_1;



d:process(CLK)
begin
```

```
if(rising_edge(CLK)) then
out_gen_minus_025<= buff_minus_generation;
out_gen_plus_025<= buff_plus_generation;
main<= buff_glonas;
end if;
end process d;


end Behavioral;
```

ПРИЛОЖЕНИЕ В

Главная сводящая программа

```
--Main
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;
library std;
use std.textio.all;

entity main_ver2 is
Port (CLK : in std_logic;
--x_out: out integer;
dif:out integer);
--              out_minus_025: out std_logic_vector (29 downto 0);
--              out_plus_025: out std_logic_vector (29 downto 0));
end main_ver2;

architecture Behavioral of main_ver2 is

component correlator_ver2 is
port( plus_minus025: in signed (4087 downto 0);
        main: in signed (4087 downto 0);
        x_in: in integer;
        CLK: in std_logic;
        result : out std_logic_vector (29 downto 0);
        pl_min025:out signed (4087 downto 0);
        main_out:out signed (4087 downto 0));
end component;

component Generator_ver2 is
port (CLK : in std_logic;
x_in: in integer;
out_gen_minus_025: out signed (4087 downto 0);
out_gen_plus_025: out signed (4087 downto 0);
main: out signed (4087 downto 0));
end component;

component Adder_lookahead_selection_30bit
port
(x_in : in std_logic_vector (29 downto 0);
        y_in : in std_logic_vector (29 downto 0);
```

```
        carry_in : in std_logic;
        CLK : in std_logic;
        sum : out std_logic_vector (29 downto 0);
        carry_out: out std_logic);
end component;


type buff is array (NATURAL range<>) of std_logic_vector (29 downto 0);
type buf is array (NATURAL range<>) of signed (4087 downto 0);
type bu is array (NATURAL range<>) of integer;
signal minus_025 :signed (29 downto 0);
signal plus_025 :signed (29 downto 0);

signal plus_025_buff: buff(5 downto 0);
signal minus_025_buff: buff(5 downto 0);

signal plus_025_buf :buf (5 downto 0);
signal minus_025_buf :buf (5 downto 0);
signal main_buf: buf (5 downto 0);

signal main_buf_minus:buf (5 downto 0);
signal main_buf_plus:buf (5 downto 0);

signal difference_buff: std_logic_vector (29 downto 0);
signal difference: std_logic_vector (29 downto 0);
signal difference_buff2: std_logic_vector (29 downto 0);
signal difference2: std_logic_vector (29 downto 0);
signal difference_buff3: std_logic_vector (29 downto 0);
signal difference3: std_logic_vector (29 downto 0);

signal difference_int: integer;
signal difference2_int: integer;
signal difference3_int: integer;

signal x: bu (3 downto 0);
signal counter: integer := 100;
signal counter2: integer := 200;
signal counter3: integer := 300;
signal overfl: std_logic;
signal overf2: std_logic;
signal overf3: std_logic;
signal start: std_logic := '0';
--x(0):=2;
```

```
begin

process(CLK)
begin
if (rising_edge(CLK)) then
        if (start = '0') then
        x(0) <= 4;
        start <= '1';
end if;
end if;
end process;



gen:Generator_ver2
port map (CLK => CLK,
x_in => x(0),
out_gen_minus_025 => minus_025_buf(0)(4087 downto 0),
out_gen_plus_025 => plus_025_buf(0)(4087 downto 0),
main => main_buf(0));

minus: correlator_ver2
port map(plus_minus025 =>minus_025_buf(0),
        main=> main_buf(0),
        x_in => 0,
        CLK => CLK,
        result => minus_025_buff(0),
        pl_min025 =>minus_025_buf(1),
        main_out => main_buf_minus(1));

plus: correlator_ver2
port map(plus_minus025 =>plus_025_buf(0),
        main=> main_buf(0),
        x_in => 0,
        CLK => CLK,
        result => plus_025_buff(0),
        pl_min025 =>plus_025_buf(1),
        main_out => main_buf_plus(1));


process(CLK)
begin
```

```
if(rising_edge(CLK)) then
        minus_025_buff(1) <= minus_025_buff(0);
end if;
end process;

process(CLK)
begin
if (rising_edge(CLK)) then
        for i in 29 downto 0 loop
        if (plus_025_buff(0)(i) = '0') then
        plus_025_buff(1)(i)<='1';
        elsif(plus_025_buff(0)(i) = '1') then
        plus_025_buff(1)(i)<='0';
        end if;
        end loop;
end if;
end process;

diff: Adder_lookahead_selection_30bit
port map(x_in =>plus_025_buff(1),
        y_in =>minus_025_buff(1),
        carry_in => '0',
        CLK =>CLK,
        sum => difference_buff,
        carry_out=>overfl);

process(CLK)
begin
if(rising_edge(CLK)) then
        difference <= difference_buff;
        difference_int <= to_integer(signed(difference_buff));
end if;
end process;


process(CLK)
begin
if(rising_edge(CLK)) then
if (difference_int < 128 and difference_int > -128) then
x(1)<= 0;
        elsif (difference(29) = '1') then
                x(1)<=-4;
        else
```

```
                x(1)<=+4;
        end if;
end if;
--end if;
end process;

minus2: correlator_ver2
port map(plus_minus025 =>minus_025_buf(1),
        main=> main_buf_minus(1),
        x_in => x(1),
        CLK => CLK,
        result => minus_025_buff(2),
        pl_min025 =>minus_025_buf(2),
        main_out => main_buf_minus(2));

plus2: correlator_ver2
port map(plus_minus025 =>plus_025_buf(1),
        main=> main_buf_plus(1),
        x_in => x(1),
        CLK => CLK,
        result => plus_025_buff(2),
        pl_min025 =>plus_025_buf(2),
        main_out => main_buf_plus(2));


process(CLK)
begin
if(rising_edge(CLK)) then
        minus_025_buff(3) <= minus_025_buff(2);
end if;
end process;

process(CLK)
begin
if (rising_edge(CLK)) then
        for i in 29 downto 0 loop
        if (plus_025_buff(2)(i) = '0') then
        plus_025_buff(3)(i)<='1';
        elsif(plus_025_buff(2)(i) = '1') then
        plus_025_buff(3)(i)<='0';
        end if;
        end loop;
end if;
```

```
end process;

diff2: Adder_lookahead_selection_30bit
port map(x_in =>plus_025_buff(3),
        y_in =>minus_025_buff(3),
        carry_in => '0',
        CLK =>CLK,
        sum => difference_buff2,
        carry_out=>overf2);

process(CLK)
begin
if(rising_edge(CLK)) then
        difference2 <= difference_buff2;
        difference2_int <= to_integer(signed(difference_buff2));
end if;
end process;


process(CLK)
begin
if(rising_edge(CLK)) then
if (difference2_int < 128 and difference2_int > -128) then
x(2)<= 0;
        elsif (difference2(29) = '1') then
                x(2)<=-2;
        else
                x(2)<=+2;
        end if;
end if;
--end if;
end process;

minus3: correlator_ver2
port map(plus_minus025 =>minus_025_buf(2),
        main=> main_buf_minus(2),
        x_in => x(2),
        CLK => CLK,
        result => minus_025_buff(4),
        pl_min025 =>minus_025_buf(3),
        main_out => main_buf_minus(3));

plus3: correlator_ver2
```

```
port map(plus_minus025 =>plus_025_buf(2),
        main=> main_buf_plus(2),
        x_in => x(2),
        CLK => CLK,
        result => plus_025_buff(4),
        pl_min025 =>plus_025_buf(3),
        main_out => main_buf_plus(3));


process(CLK)
begin
if(rising_edge(CLK)) then
        minus_025_buff(5) <= minus_025_buff(4);
end if;
end process;

process(CLK)
begin
if (rising_edge(CLK)) then
        for i in 29 downto 0 loop
        if (plus_025_buff(4)(i) = '0') then
        plus_025_buff(5)(i)<='1';
        elsif(plus_025_buff(4)(i) = '1') then
        plus_025_buff(5)(i)<='0';
        end if;
        end loop;
end if;
end process;

diff3: Adder_lookahead_selection_30bit
port map(x_in =>plus_025_buff(5),
        y_in =>minus_025_buff(5),
        carry_in => '0',
        CLK =>CLK,
        sum => difference_buff3,
        carry_out=>overf3);

process(CLK)
begin
if(rising_edge(CLK)) then
        difference3 <= difference_buff3;
        difference3_int <= to_integer(signed(difference_buff3));
end if;
```

```
end process;


process(CLK)
begin
if(rising_edge(CLK)) then
if (difference3_int < 128 and difference3_int > -128) then
x(3)<= 0;
        elsif (difference3(29) = '1') then
                x(3)<=-1;
        else
                x(3)<=+1;
        end if;
end if;
--end if;
end process;

process(CLK)
begin
if(rising_edge(CLK)) then
--      x_out <= x;
        dif<=to_integer(signed(difference3));
end if;
end process;


write0_0:process(CLK)
file out_file : text open WRITE_MODE is "I:\QS\out.txt";
variable fline : line;
begin
if (rising_edge(CLK)) then
WRITE( fline, x(0));
WRITE( fline, x(1));
WRITE( fline, x(2));
WRITE( fline, x(3));
writeline(out_file, fline);
end if;
end process write0_0;


write0:process(CLK)
file out_file : text open WRITE_MODE is "I:\QS\out0.txt";
variable fline : line;
```

```
begin
if (rising_edge(CLK)) then
WRITE( fline, x(0));
writeline(out_file, fline);
end if;
end process write0;


write1:process(CLK)
file out_file : text open WRITE_MODE is "I:\QS\out1.txt";
variable fline : line;
begin
if (rising_edge(CLK)) then
WRITE( fline, x(1));
writeline(out_file, fline);
end if;
end process write1;


write2:process(CLK)
file out_file : text open WRITE_MODE is "I:\QS\out2.txt";
variable fline : line;
begin
if (rising_edge(CLK)) then
WRITE( fline, x(2));
writeline(out_file, fline);
end if;
end process write2;


write3:process(CLK)
file out_file : text open WRITE_MODE is "I:\QS\out3.txt";
variable fline : line;
begin
if (rising_edge(CLK)) then
WRITE( fline, x(3));
writeline(out_file, fline);
end if;
end process write3;


end Behavioral;
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра «Систем автоматики, автоматизированного управления и проектирования»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ С.В. Ченцов
«18» 06. 2018 г.

БАКАЛАВРСКАЯ РАБОТА

15.03.04 – Автоматизация технологических процессов и производств

РАЗРАБОТКА СХЕМЫ СЛЕЖЕНИЯ ЗА ЗАДЕРЖКАМИ НАВИГАЦИОННОГО СИГНАЛА С ПОМОЩЬЮ CORDIC-ПРОЦЕССОРА

Руководитель	_____	__.06. 2018 г.	зав. кафедрой, канд. техн. наук Д.В. Капулин
Выпускник	_____	14.06. 2018 г.	Н.Н. Овчинников
Нормоконтролер	_____	14.06. 2018 г.	Т.А. Грудинова

Красноярск 2018