|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**факультет “Информатика и системы управления”**

**Лабораторная работа №4**

**“Проектирование цифрового синтезатора”**

**по курсу**

**“Проектирование цифровых устройств информационных и телекоммуникационных систем ”**

Продолжительность работы: 4 часа

Самостоятельная подготовка: 4 часа

Составитель: Федоров С.В.

## Москва

**2021**

**Содержание**

Цель работы 2

Введение 2

1. Разработка модулей 3

1.1. Накопитель фазы 3

1.2. Просмотровая таблица функции 3

1.3. Дельта-сигма модулятор 4

2. Реализация синтезатора. 6

Содержание отчета 11

Контрольные вопросы 11

Самостоятельная подготовка 11

Литература 11

# Цель работы

Осуществить полный цикл проектирования цифрового устройства на ПЛИС на языке SystemVerilog. Реализовать генератор периодических функций на основе метода прямого цифрового синтеза на микросхеме семейства Cyclone IVE.

**Требования к аппаратному обеспечению:**

**Для выполнения лабораторной работы требуется, чтобы к компьютеру была подключена отладочная плата Terasic DE2-115. В данной лабораторной работе в качестве базового используется проект системы на кристалле, разработанный в л.р.№2.**

Введение

Прямой цифровой синтез - это метод генерации сигнала заданной формы путём вычисления значений сигнала в последовательных дискретных отсчётах времени и преобразовании их в аналоговый сигнал с помощью ЦАП. Структура системы приведена на рис 1.

Рисунок 1. Схема цифрового синтезатора

Накопитель фазы

ПЗУ

Sin

Дельта-сигма модулятор

к осциллографу

R

C

CLK

CLK

CLK

инкремент фазы

Инкремент фазы поступает на вход накопителя фазы и определяет скорость изменения фазы формируемого сигнала. Накопитель фазы выдаёт адрес текущего отсчёта на просмотровую таблицу синусоиды. Значение с выхода просмотровой таблицы поступает на дельта-сигма модулятор, который работает как ЦАП. Двухуровневый (1 битный) выходной сигнал модулятора подается на выход ПЛИС и сглаживается простым RC фильтром НЧ, соединенным с выходом ПЛИС. При изменении приращения фазы изменяется частота формируемого на выходе сигнала.

1. Разработка модулей

1.1. Накопитель фазы

**Задание**

Разработать параметризируемый накопитель фазы.

Интерфейс модуля

Входы:

phinc[7:0] – величина приращения фазы за один период тактового сигнала;

clk – тактовый сигнал;

clr\_n – вход асинхронного сброса;

Выходы:

phase[7:0] – 8 старших значащих битов выхода накопителя;

Параметры:

WIDTH – разрядность накопителя фазы (значение по умолчанию – 14).

Принцип действия

Накопитель фазы - это классический аккумулятор, который сохраняет накопленную сумму в регистре и использует её в качестве одного из операндов сумматора на каждом такте. Второй операнд поступает с входа и определяет величину приращения фазы. Разрядность аккумулятора должна быть параметризирована. Входные значения складываются с младшими битами регистра аккумулятора, а на выход поступают старшие 8 разрядов аккумулятора, поэтому при разрядности аккумулятора M и значении N на входе приращения фазы выход фазы будет увеличиваться на единицу один раз в 2(M-8)/N тактов.

1.2. Просмотровая таблица функции

**Задание**

Реализовать просмотровую таблицу синусоиды в диапазоне от 0 до 2, содержащую 256 отсчётов. Выход должен быть представлен 8 разрядным знаковым значением синусоиды (в дополнительном коде), соответствующим номеру отсчёта на входе. Объединить и отладить модули накопителя фазы и просмотровой таблицы периодической функции (реализовать тестбенч и провести моделирование в пакете ModelSim).

Интерфейс модуля

Определяется мастером настройки мегафункций

Принцип действия

Просмотровая таблица может быть реализована на ПЗУ, шина адреса которого управляется входным значением фазы сигнала, а ячейки содержат соответствующие значения сигнала. Для решения данной задачи требуется 256\*8 ячеек памяти. Для реализации используйте модуль **ROM: 1-PORT** в MegaWizard Plug-In Manager. Файл инициализации памяти **sine256.mif** прилагается в исходных файлах для лабораторной работы.

1.3. Дельта-сигма модулятор

**Задание**

Разработать дельта-сигма модулятор 1го порядка, который осуществляет передискретизацию 8-разрядного входного отсчёта сигнала в однобитную выходную последовательность (2 возможных состояния на выходе). Провести отладку модулятора. Рекомендуется провести моделирование на значениях констант: 0, значение, близкое к максимуму шкалы, значение, близкое к минимуму шкалы.

Интерфейс модуля

Входы: val[7:0] - входные данные модулятора;

clk – тактовый сигнал;

clr\_n – вход асинхронного сброса.

Выход: daco - однобитная выходная последовательность.

**Принцип действия**

На вход поступают 8-разрядные отсчеты xn (рис. 2). Выход yn - двухуровневый дискретизированный с большой частотой (передискретизация).

Q- квантователь. Вычисляет однобитное значение yn и ошибку квантования εn. Уравнения, определяющие работу квантователя, описываются следующим образом:

Формирование выходного значения:

yn=1, y’n=максимум шкалы, если un ближе к максимуму шкалы

yn=0, y’n=минимум шкалы, если un ближе к минимуму шкалы

Формирование ошибки квантования:



εn=y’n-un

Рисунок 2. Схема дельта-сигма модулятора 1ого порядка

Таким образом, мы аппроксимируем значение на входе квантователя однобитным значением, причем 0 соответствует минимуму шкалы, а 1 – максимуму шкалы. Максимум и минимум шкалы определяются границами диапазона разрядной сетки входных данных.

С помощью обратной связи ошибка квантования накапливается в цифровом интеграторе и, таким образом, учитывается на следующих тактах. Значение на выходе элемента задержки на один такт Z-1 (регистра un) определяется как

un=xn-1- εn-1

Таким образом, работа модулятора описывается системой уравнений:

un=xn-1- εn-1

εn=y’n-un

Проведем анализ работы схемы в частотной области. Из системы получим следующее уравнение:

y’n= xn-1 + εn - εn-1

Выполним Z-преобразование:

Y(z)= X(z)z-1+E(z)(1-z-1)

Передаточная функция по ошибке квантования

Hε(z)=Y(z)/E(z)=1-z-1

Для оценки мощности шума квантования перейдем к преобразованию Фурье, заменив z на ejωT, где T – период работы квантователя (период тактового импульса).

Hε(ω)=1- ejωT

Выразив комплексную экспоненту через формулу Эйлера, вычислим квадрат модуля Hε(ω), определяющий мощность шумов квантования в выходном сигнале в частотной области:

|Hε(ω)|2=2(1-соs(ωT))

или

|Hε(f)|2=2(1-соs(2πfT))

На рис.3 голубая линия соответствует мощности шумов квантования, красная – спектру генерируемого сигнала, зеленая – АЧХ внешнего ФНЧ.

Очевидно, что основная мощность шума квантования сосредотачивается в высокочастотной области спектра. Спектр полезного сигнала должен лежать в низкочастотной области.

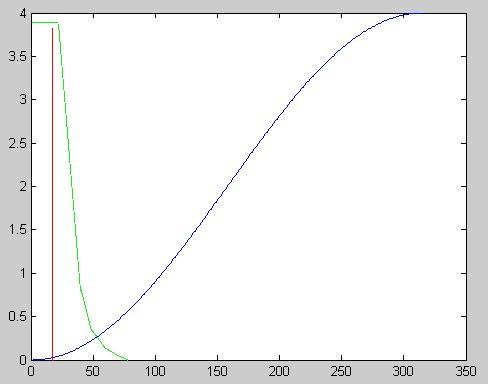
Последующей аналоговой НЧ фильтрацией достигается эффективное разрешение, соответствующее 8-ми разрядному ЦАПу.

Рисунок 3. Спектр шума квантования дельта-сигма модулятора 1ого порядка

2. Реализация синтезатора

**Задание**

Используя разработанные модули, реализовать модуль цифрового синтезатора и интегрировать его состав системы на кристалле из Л.Р.№2.

Интерфейс модуля

Входы:

clk – тактовый сигнал;

clr\_n – вход асинхронного сброса;

wr\_data – сигнал writedata интерфейса Avalon Slave-MM;

wr\_n – сигнал write\_n интерфейса Avalon Slave-MM;

Выходы:

fout – выход модулятора;

Параметры:

PHACC\_WIDTH – разрядность накопителя фазы (значение по умолчанию – 14).

Задание

1. Реализовать модуль цифрового синтезатора с интерфейсом Avalon-MM. Произвести моделирование модуля. Показать преподавателю результаты работы.
2. Интегрировать проект в состав системы на кристалл. Разработать тестовое ПО. Проверить правильность работы схемы с помощью осциллографа и светодиода.

**Порядок выполнения работы**

1. На основе разработанных модулей реализуйте модуль с интерфейсом Avalon Slave-MM, соответствующий схеме, приведенной на рис. 1, как отдельный проект в Quartus. По интерфейсу доступен только один регистр – регистр, в котором хранится значение инкремента фазы, поэтому вход адреса не требуется. Любая запись в модуль приводит к записи в данный регистр. Выходной сигнал модулятора выведите на одноразрядный выход **fout**.
2. Осуществите моделирование работы модуля в пакете ModelSim. Тестбенч **gen\_tb.sv** прилагается в исходных файлах для лабораторной работы. При правильной работе модуля должен быть сформирован один период синуса.
3. Изучите тестбенч, объясните формирование значения переменной phinc\_val и времени ожидания перед директивой $stop.

* Почему именно за это время из памяти выбирается один период синуса?
* Какая разрядность накопителя фазы требуется для формирования синусоидального сигнала в диапазоне частот от единиц до сотен Герц при изменении инкремента фазы от 1 до 255 (период синуса содержит 256 отсчетов, частота тактового импульса- 50МГЦ)?

Покажите результаты моделирования и ответы на вопросы преподавателю. Приведите их в отчете.

Пример диаграмм, полученных при моделировании, приведен на рис. 4. Помимо узлов верхнего уровня тестбенча, в окно выведено значение регистра инкремента фазы для контроля записи в регистр, а также значения синуса на входе модулятора и выхода регистра u. Обратите внимание на характер и диапазон изменения u.

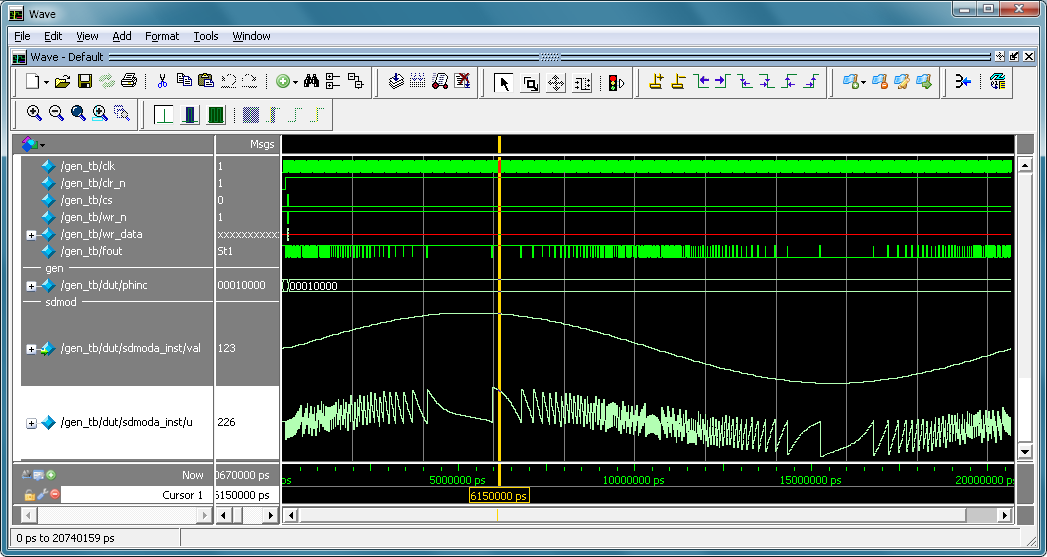


Рисунок 4. Пример результата моделирования синтезатора

1. Аналогично интеграции модуля **dec.sv** в л.р. №2 интегрируйте модуль в состав системы на кристалле, реализованной в л.р. №2. Так как проект модуля синтезатора содержит несколько файлов и файл инициализации памяти, рекомендуется скопировать папку с исходными файлами синтезатора в папку HDL проекта л.р. №2, а тестбенч в папку Testbench. Таким образом можно при необходимости изменять и моделировать синтезатор как отдельный проект и использовать его как модуль в системе на кристалле.

При создании компонента в **Platform Designer** добавьте в список файлов для синтеза все исходные файлы проекта, включая файл, реализующий ПЗУ, созданный мастером (рис.5) и файл mif с отсчетами синуса (на рисунке не показан, нужно добавить его и указать тип MIF Data в столбце Type).

Далее в столбце Attributes установите атрибут Top-level File для файла верхнего уровня и нажмите Analyze Synthesis Files. Если в списке Top-Level Module не появился автоматически модуль верхнего уровня, укажите его. Далее настройте интерфейсы, типы сигналов, имя модуля и сохраните компонент аналогично л.р. №2.

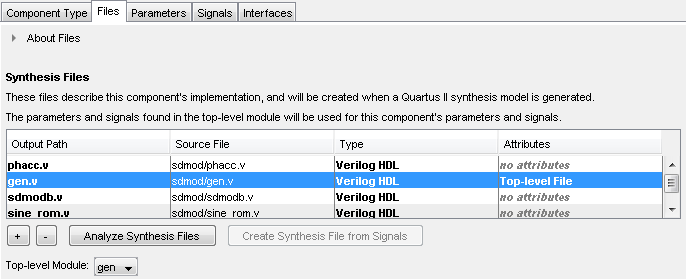


Рисунок 5. Добавление нескольких файлов для сложного модуля

1. Добавьте в систему созданный компонент. Настройте параметр, определяющий разрядность накопителя фазы, таким образом¸ чтобы при тактировании модулятора частотой 50 МГц, минимальная выходная частота формируемого синусоидального сигнала лежала в диапазоне от 0.1 до 1 Гц. Подключите модуль к тактовому импульсу **clk** и шине данных процессора **data\_master**. Экспортируйте выход модулятора.
2. Отключите компонент семафора, сняв переключатель в столбце **Use**. Обновите карту памяти и сеть сигналов сброса.
3. Сохраните проект **Platform Designer**. На странице **Generation** нажмите **Generate**. По завершении генерации закройте **Platform Designer**.
4. Так как проект будет запускаться на отладочной плате, надо сделать корректные назначения портов ввода-вывода и их стандартов. Кроме того, на плате отсутствует сигнал сброса, вход которого экспортирован в системе на кристалле. Для удобства создадим настройки с помощью мастера генерации шаблона проекта системы на кристалле, который использовался в курсе ПОВС (при использовании ОС Linux шаблон уже создан и находится в папке **CodeGenerated\DE2\_115\Lab4**, переходите к п. 9). Запустите программу **DE2\_115\_SystemBuilder.exe** (есть в материалах лабораторной работы). Включите только **Clock** и **Led** и сгенерируйте проект шаблона под именем **Lab4** (рис. 6).
5. Скопируйте файлы **Lab4.v** и **Lab4.sdc** из папки, в которой был сохранен проект шаблона, в папку **Top** лабораторной работы. Добавьте их в состав проекта. Перейдите на закладку **Files** в навигаторе проекта и установите файл **Lab4.v** верхним уровнем проекта.
6. В случае, если система на кристалле не является верхним уровнем при синтезе, задействование сгенерированных в **Platform Designer** файлов следует осуществлять иначе, чем это было сделано в л.р. №2. Удалите файл **niosII.qsys** из состава проекта и добавьте файл **niosII.qip** из папки **niosII\synthesis** в папке проекта.
7. Для копирования настроек размещения выводов и их стандартов откройте файл настроек Вашего проекта **<имя проекта>.qsf** в папке проекта и **Lab4.qsf** в папке шаблона в текстовом редакторе (например, Notepad++). Скопируйте назначение выводов и их стандартов в группах CLOCK и LED из файла **Lab4.qsf** в файл настроек Вашего проекта. Если в файле настроек Вашего проекта есть другие настройки размещения и стандартов выводов, удалите их. Сохраните файл настроек Вашего проекта.

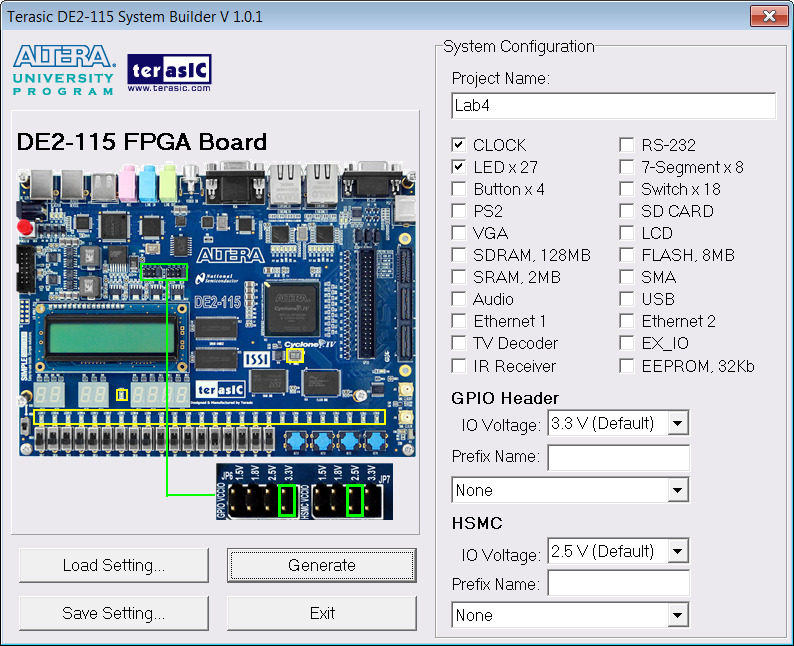


Рисунок 6. Настройка проекта системы на кристалле.

1. Задействуйте систему на кристалле в модуле верхнего уровня, а также определите значения неиспользуемых светодиодов следующим образом (скорректируйте, если имена выводимых сигналов отличаются):

//===========================================

// Structural coding

//===========================================

niosII u0 (

.clk\_clk (CLOCK\_50),

.reset\_reset\_n (1'b1),

.gen\_fout (LEDG[0])

);

assign LEDG[7:1] = 1'b0;

assign LEDR[17:0] = 1'b0;

1. Для возможности вывода сигнала генератора не только на светодиод, но и на осциллограф, добавьте в список портов одноразрядный выход с именем FOUTA и продублируйте выход синтезатора, выведенный на LEDG[0], на выход FOUTA.
2. Сохраните обновленный файл Lab4.v и проведите анализ проекта в в Quartus, вызвав **Processing > Start > Start Analysis & Elaboration**. При наличии ошибок исправьте их.
3. Осуществите назначение добавленного вывода FOUTA на вывод ПЛИС. Назначение выводов может осуществляться в редакторе **Pin Planner** (**Assignments->Pin Planner**). Требуется назначить выход на вывод AE15, и указать стандарт 3.3-V LVTTL (см. пример для другого проекта на рис. 6). Обратите внимание на то, что здесь же можно указать ток выхода, скорость нарастания и другие настройки.

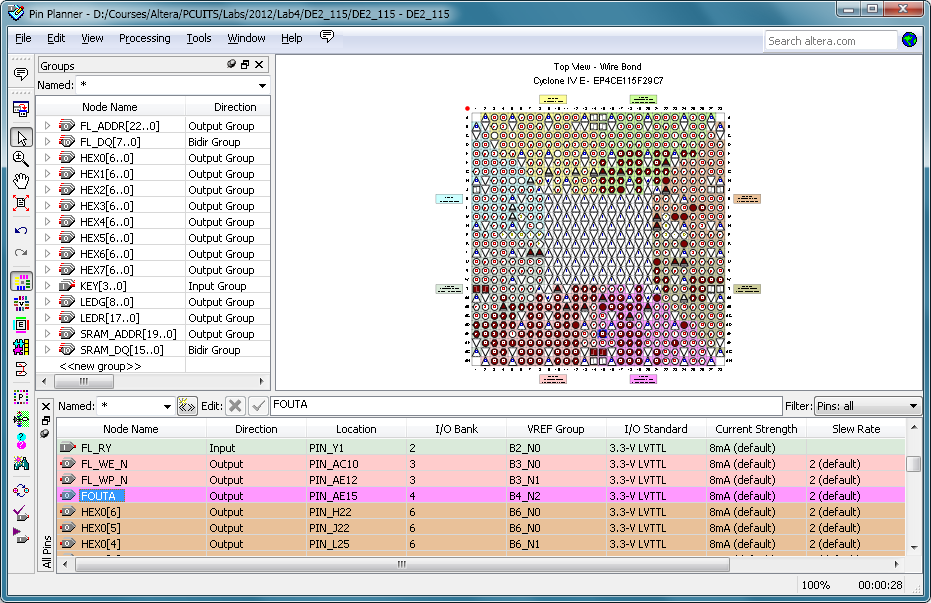


Рисунок 6. Редактор разводки выводов Pin Planner

1. Откомпилируйте проект с введенными назначениями выводов. Откройте программатор **Programmer** САПР Quartus, загрузите конфигурацию системы на кристалле в отладочную плату. По завершении программирования закройте программатор.
2. Разработайте программное обеспечение системы на кристалле, управляющее инкрементом фазы. Выход схемы FOUTA выведен на вывод 10 разъема JP5 для фильтрации внешним RC-фильтром, также сигнал выведен на светодиод LEDG0. Получите у преподавателя RC-фильтр для проверки формирования аналогового сигнала. Проверьте правильность работы разработанного проекта на осциллографе и визуально на светодиоде LEDG0.

Содержание отчета

1. ФИО студента, номер группы, дата выполнения лабораторной работы.
2. Исходные тексты модулей и временные диаграммы, полученные при моделировании модулей отдельно и проекта в целом.
3. Исходный код программы.

### Контрольные вопросы

1. Дельта-сигма модулятор первого порядка.
2. Конфигурация ПЛИС Altera на основе статической памяти.
3. Язык SystemVerilog

### Самостоятельная подготовка

В рамках самостоятельной подготовки требуется реализовать модули раздела 1 и тестбенчи для них.

Для лучшего понимания работы модулятора проведите расчет значений un,xn,y'n и εn для 5 тактов работы дельта-сигма модулятора. Приведите значения для каждого такта и график выходного сигнала yn. Используйте значения: xn=номер варианта\*3-21 для группы 1 и xn=-номер варианта\*4+12 для группы 2. В начальный момент времени регистр z-1 сброшен.

Для проверки корректности работы модулятора рекомендуется осуществить его моделирование отдельно на нескольких постоянных значениях, например, 0, значение, близкое к максимуму шкалы и значение, близкое к минимуму шкалы. Таким образом, можно проверить соответствие поведения модулятора уравнениям, приведенным в п. 1.3.

При выполнении лабораторной работы в классе осуществляется отладка реализованных модулей, объединение модулей, моделирование генератора как компонента системы на кристалле, его интеграция в состав системы на кристалле и тестирование системы в целом.

Для допуска к лабораторной работе требуется представить исходные тексты реализованных модулей и их тестбенчей, а также расчет значений при работе модулятора.

### Литература

1. Федоров С.В. Язык Verilog. Электронное учебное пособие. - М.: МГТУ им. Баумана, 2009. - 70c.
2. Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. -М.: СОЛОН-Пресс, 2003. – 320c.
3. Методические материалы к лабораторной работе.