СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc516240527)

[ОСНОВНАЯ ЧАСТЬ 6](#_Toc516240528)

[Глава 1 Теоретическая часть 6](#_Toc516240529)

[1.1 Аппаратные составляющие вычислительного устройства 6](#_Toc516240530)

[1.2 Адресные пространства, способы адресации, форматы команд 12](#_Toc516240531)

[1.3 Программируемая логическая интегральная схема 15](#_Toc516240532)

[1.4 Программируемые пользователем вентильные матрицы (FPGA) 18](#_Toc516240533)

[1.5 Ускоренное выполнение операции умножения 21](#_Toc516240534)

[Глава 2 Практическая часть 24](#_Toc516240535)

[2.1 Код программы с горизонтальным подходом к реализации 24](#_Toc516240536)

[2.2 Блок схема алгоритма умножения 26](#_Toc516240537)

[2.3 Структурная схема 28](#_Toc516240538)

[2.4 Последовательность микрокоманд 29](#_Toc516240539)

[2.5 Написание микропрограммы 31](#_Toc516240540)

[ЗАКЛЮЧЕНИЕ 37](#_Toc516240541)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 39](#_Toc516240542)

# ВВЕДЕНИЕ

Темой данного дипломного проекта является разработка арифметического устройства и блока устройства управления вычислительного устройства для команды ускоренного умножения чисел с фиксированной запятой методом Лемана разработанные на ПЛИС.

Актуальность разрабатываемой темы дипломного проекта определяется необходимостью наличия навыков по разработке устройств данного типа для студентов направления 09.02.01.

Разрабатываемое вычислительное устройство состоит из арифметико-логического и управляющего устройств.

Устройство управления (УУ) предназначено для выработки управляющих сигналов, необходимых для выполнения любого действия, происходящего в СОД

Классификация УУ:

* + по структурной организации:
    1. Централизованные;
    2. Смешанные – централизованные + местные;
    3. Иерархические (в ВС);
  + по технической организации:
    1. С жесткой логикой работы;
    2. Устройства микропрограммного управления.

Арифметико-логическое устройство (АЛУ). Предназначается для выполнения арифметических и логических операций над данными. В АЛУ выполняются операции с фиксированной точкой; с плавающей точкой; операции двоично – десятичной арифметики, логические операции, обработка алфавитно-цифровой информации.

Арифметико-логическое устройство в зависимости от выполнения функций можно разделить на две части:

* + Микропрограммное устройство (устройство управления), задающее последовательность микрокоманд (команд);
  + Операционное устройство, в котором реализуется заданная последовательность микрокоманд (команд).

АЛУ могут работать либо с числами с фиксированной запятой, либо плавающей.

По способу действия над операндами АЛУ делятся на последовательные и параллельные. В последовательных АЛУ операнды поступают в последовательном коде (побитно), то есть они являются одноразрядными. Недостатком последовательных АЛУ является низкое быстродействие, достоинством - простота реализации. В параллельных АЛУ операнды поступают в параллельном коде в виде 8-и, 16-и, 32-х разрядных слов. Операция производится одновременно над всеми битами слова.

По структуре АЛУ делятся на АЛУ с непосредственными связями и АЛУ с магистральной структурой. В первом типе межрегистровые связи внутри АЛУ осуществляются непосредственно друг с другом; во втором - обмен между регистрами осуществляется через общую шину.

По способу организации работы различают асинхронные и синхронные АЛУ. В асинхронных АЛУ определяется момент окончания текущей операции, после чего может начинаться следующая операция. В синхронных АЛУ на выполнение любых операций отводится фиксированный интервал времени, независимо от того, какая операция выполняется «короткая» (сложение, вычитание) или «длинная» (умножение, деление). Первый тип АЛУ более быстродействующий, чем второй.

По характеру использования элементов и узлов АЛУ делятся на блочные и многофункциональные. В блочных АЛУ для каждого типа операции применяют специализированный блок обработки, например, блок умножения, блок деления, блок сложения - вычитания, блок логических операций, блок десятичной арифметики, блок арифметики с плавающей точкой. В многофункциональных АЛУ для выполнения всех типов операций используется один блок, выполняющий все виды операций. В блочных АЛУ быстродействие выше, поскольку каждый блок может выполнить операции параллельно с операциями в других блоках. Однако затраты оборудования при этом возрастают. В многофункциональных АЛУ используется общее оборудование для выполнения различных типов операций. Параллелизм при выполнении операций в многофункциональных АЛУ невозможен, поэтому их быстродействие невысокое, но затраты оборудования минимальны.

Целью дипломного проекта является разработка микропрограммного устройства управления вычислительного устройства для команды ускоренного умножения чисел с фиксированной запятой методом Лемана и его реализация на ПЛИС.

Для выполнения цели в дипломном проекте поставлены следующие задачи:

* + Изучение ускоренного умножения чисел с фиксированной запятой методом Лемана;
  + Изучение принципа работы вычислительного устройства;
  + Выбор способа организации блока устройства управления;
  + Разработка структурной схемы блока устройства управления и арифметического устройства вычислительного устройства;
  + Разработка программной реализации вычислительного устройства;

Объектом исследования является вычислительное устройство для команды ускоренного умножения чисел с фиксированной запятой методом Лемана.

Для написания программы вычислительного устройства используется язык описания аппаратуры SystemVerilog.

Предметом исследования является вычислительное устройство для команды ускоренного умножения чисел с фиксированной запятой методом Лемана. В процессорах вычислительные устройства используются для выполнения различных команд.

При выполнении проекта использовались следующие методы научного исследования:

* + Теоретический анализ и синтез;
  + Реферирование информационных источников;
  + Анализ научной литературы и документов;
  + Моделирование вычислительного устройства.

# ОСНОВНАЯ ЧАСТЬ

# Глава 1 Теоретическая часть

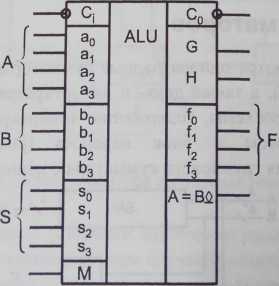
## 1.1 Аппаратные составляющие вычислительного устройства

Вычислительное устройство состоит из арифметико-логического устройства и блока устройства управления (БУУ).

Арифметико-логические устройства, или сокращенно АЛУ (ALU, Arithmetic-Logic Unit), выполняют над словами ряд действий. Основой АЛУ служит сумматор, схема которого дополнена логикой, расширяющей функциональные возможности АЛУ и обеспечивающей его переход с одной операции на другую. [1]

Для наращивания разрядности АЛУ объединяются в вариантах с последовательными или параллельными переносами. Логические возможности АЛУ различных технологий идентичны. В силу самодвойственности выполняемых операций условное обозначение и таблица истинности АЛУ встречаются в двух вариантах, отличающихся взаимно обратными значениями переменных.

АЛУ (см. Рисунок 1.1) имеет входы операндов А и В, входы выбора операций S, вход переноса и вход М (Mode), сигнал которого задает тип выполняемых операций: логические (М = 1) или арифметико-логические (М = 0). Результат операции вырабатывается на выходах F, выходы G и Н дают функции генерации и прозрачности, используемые для организаций параллельных переносов при наращивании размерности АЛУ. Сигнал – выходной перенос, а выход А = В есть выход сравнения на равенство с открытым коллектором.

  
Рисунок 1.1 Условное обозначение АЛУ

Работа какого-либо устройства (операционного блока), в том числе центрального процессора, полностью определяется структурной схемой его работы.

Структурная схема состоит из блоков прямоугольной и ромбовидной формы. Прямоугольные блоки определяют работу узлов операционного блока, ромбовидные – обеспечивают выбор одного из возможных путей реализации структурной схемы. [2]

Действия, указанные в прямоугольном блоке, которые должен выполнить операционный блок, образуют микрокоманду. Каждое из указанных действий в прямоугольном блоке называют микроприказом. Поэтому микрокоманда состоит из одного или нескольких микроприказов. Заметим, что могут быть и пустые микрокоманды, т.е. не задающие никакого действия операционному блоку.

Любой из микроприказов практически может рассматриваться как пересылка информации с целью ее запоминания или преобразования из одного регистра в другой регистр, включая также микроприказы РАП:=РАК, РЧП:=ЧТ(РАП), РС:=PI+P2.

Чтобы обеспечить выполнение требуемых микроприказов не постоянно, а в строго определенное время, необходимо расчленить связи, соединяющие соответствующие регистры, управляющими вентилями.

В этом случае управляющий вентиль имеет два входа (от передающего триггера и управляющий) и один выход, подсоединенный к приемному триггеру (см. Рисунок 1.2).

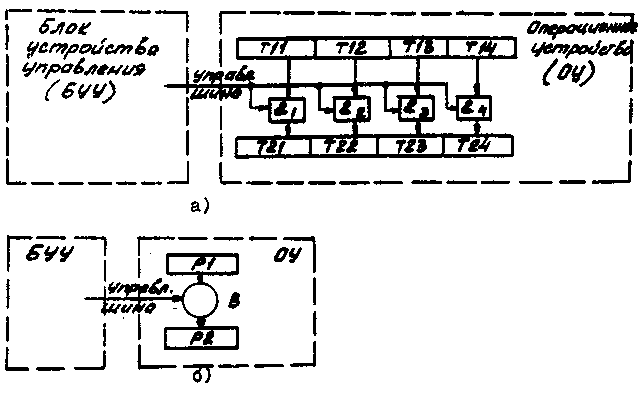


Рисунок 1.2 Упрощенная структурная схема процессора

Тогда во время установления единичного сигнала на управляющем входе управляющего вентиля будет происходить передача данных от передающего к приемному регистру. Таким образом определяется взаимно однозначное соответствие между микроприказами и управляющими вентилями, т.е. каждому микроприказу взаимно-однозначно соответствует номер управляющего вентиля. Поэтому устройство управления, выдавая управляющие единичные сигналы на определенные управляющие вентили и в требуемое время (в соответствии со структурной схемой), обеспечивает выполнение операционным устройством необходимого прямоугольного блока структурной схемы.

Действия, определенные в ромбовидном блоке, сводятся к проверке указанных условий и выборе одного из возможных путей реализации структурной схемы.

Проверку условия и выбор пути выполняет устройство управления. Для этого устройству управления должны быть доступны триггеры, от содержимого которых зависят проверяемые условия. Будем называть множество этих триггеров индикаторами.

Таким образом, при обсуждении работы устройства управления (УУ) требуется вьщелить в операционном блоке управляющие вентили B1, В2,..., BN, к которым подсоединены шины от блока устройства управления, и регистр индикаторов И1, И2,..., ИМ, шины от которых подсоединены к блоку устройства управления (см. Рисунок 1.3).

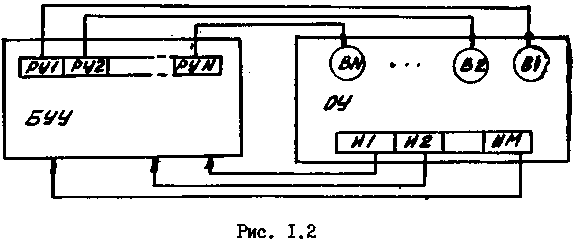


Рисунок 1.3 Структурная схема работы операционного устройства

Можно отметить следующее:

* + Число различных микроприказов определяет число управлявших вентилей, а каждому микроприказу соответствует свой управляющий вентиль;
  + При выполнении операционным блоком микрокоманды индикаторы операционного блока устанавливаются в новые состояния (быть может, совпадающие со старыми);
  + Устройство управления в соответствии с прямоугольными блоками структурной схемы работы операционного устройства формирует и выдает управляющие сигналы на управляющие вентили. Каждый такт работы операционного устройства полностью определяется номерами управляющих вентилей, на которые подаются единичные управляющие сигналы. Состояния индикаторов учитываются устройством управления при необходимости выбора одного из блоков;
  + Логика блока устройства управления определяется структурной схемой работы операционного устройства.

Введем в блоке устройства управления регистр управления (РУ) длиной *N* бит. Установим взаимно-однозначное соответствие между номерами разрядов регистра управления и номерами управляющих вентилей по правилу: i*-*й разряд РУ соответствует i-му вентилю. Тогда работа устройства управления сводится к систематическому формированию двоичных последовательностей.

Любое устройство обработки цифровой информации, включает в себя две основные части:

* + Операционную часть (операционное устройство);
  + Управляющую часть (устройство управления).

Операционная часть состоит из регистров, счётчиков, сумматоров, дешифраторов и связей между ними. Операционная часть функционирует под воздействием управляющих сигналов, которые вырабатывает управляющее устройство. Операционная часть выполняет заданную микропрограмму, состоящую из микрокоманд.

Микрокоманда включает в себя одну или несколько микроопераций. Микрооперация – это элементарная функциональная операция, выполняемая под воздействием одного управляющего сигнала в течение одного такта. Если в течение одного такта выполняется несколько микроопераций под воздействием различных управляющих сигналов, то они объединяются в одну микрокоманду.

Устройство управления(УУ) служит для выработки последовательности управляющих сигналов, под воздействием которых выполняются микрооперации. В зависимости от способа выработки управляющего сигнала различают 2 основных подхода к построению УУ:

* + Микропрограммная реализация УУ (см. Рисунок 1.4);
  + Аппаратная реализация УУ (схемная реализация или УУ с жёсткой логикой).

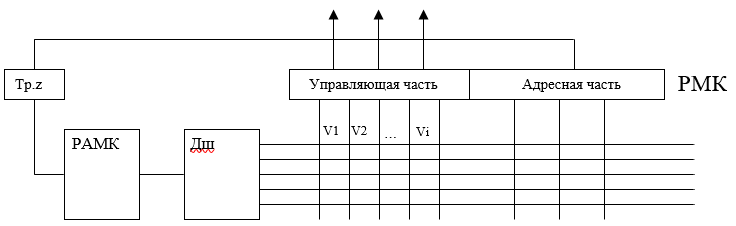


Рисунок 1.4 Микропрограммная реализация устройства управления, предложенная М.В. Уилксом

РАМК – регистр адреса микрокоманд;

РМК – регистр микрокоманд;

Дш – дешифратор;

Тр.z – триггер задержки.

Микропрограмма, состоящая из микрокоманд, записывается в память микрокоманд. Каждая микрокоманда состоит из двух частей: управляющей части, где кодируются управляющие сигналы Vi и адресной части.

В адресной части микрокоманды кодируется адрес ячейки памяти, где хранится следующая выполняемая микрокоманда. В начальный момент времени на РАМК подается адрес ячейки памяти, где хранится первая микрокоманда. По этому адресу из памяти микрокоманда считывается и подается на регистр микрокоманд.

Из управляющей части микрокоманды управляющие сигналы подаются на вентили в операционную часть ЦП, а из адресной части на регистр адреса микрокоманд в следующем такте заводится адрес следующей выполняемой микрокоманды.

Существует два подхода к реализации микропрограммного устройства управления: горизонтальный и вертикальный.

При горизонтальном микропрограммировании для каждого управляющего сигнала в управляющей части выделяется отдельный разряд. Если в некотором такте управляющий сигнал должен быть равен 1, то в соответствующем разряде записывается 1. Таким образом, количество разрядов в управляющей части соответствует числу управляющих сигналов. При такой организации можно совмещать микрооперации во времени.

Достоинство: высокое быстродействие за счёт совмещения микроопераций во времени.

Недостатки: требуется большая ёмкость памяти для хранения микрокоманды, так как реально совместить большое количество микроопераций в одной микрокоманде не удаётся.

Вертикальный подход к реализации микропрограммного устройства управления. В операционной части МК кодируется номер управляющего сигнала, поэтому на выходе регистра микрокоманд в управляющей части ставится дешифратор.

Достоинство: значительно сокращается емкость памяти микрокоманд.

Недостатки: увеличенное время выполнения микрокоманд из-за невозможности совмещения микроопераций в микрокоманде и появления дешифратора на выходе.

Выполнение операции перехода на микропрограммном уровне. При необходимости выполнения команды перехода, на микропрограммном уровне, адрес следующей выполняемой микрокоманды будет состоять из 2-х частей:

* + Основной (базовой) части (адреса микрокоманды), который выбирается (хранится) в адресном поле микрокоманды перехода;
  + Значений признаковых триггеров, которые определяют младшую часть адреса микрокоманды. Значения признаковых триггеров формируются в операционной части центрального процессора. Таким образом, при использовании одного признакового триггера, при выполнении операции перехода на микропрограммном уровне в зависимости от условия, которое содержится на признаковом триггере, будет сформировано два адреса, отличающихся младшими разрядами (см. пример выполнения операции умножения в АЛУ).

## 1.2 Адресные пространства, способы адресации, форматы команд

Совокупность адресов, которые могут быть использованы процессором, образует адресное пространство МПС. Адреса памяти могут занимать все адресное пространство (АП) или его часть, а линейно организованная память независимо от ее технической реализации может быть условно представле­на набором регистров (ячеек), число которых М, а разрядность – N (см. Рисунок 1.5). [3]



Рисунок 1.5 Условное представление линейной организации памяти

Диапазон адресов, к которым может обращаться процессор (т. е. емкость АП) связан с разрядностью шины адреса m соотношением АП = 2m*.* Например, с помощью 16-разрядной шины адреса можно адресовать 216 = 64К объектов, с помощью 20-разрядной 1М объектов и т. д.

АП используется блоками ОЗУ, ПЗУ и ВУ, к которым обращается процессор. Распределение АП между указанными претендентами производится проектировщиком системы, имеющим известную свободу действий, хотя у конкретных процессоров могут быть особенности, заставляющие отдавать определенную область АП для адресации определенных объектов.

Для оценки емкостей АП используются обычно единицы измерения К=210=1024, М=220, Г=230. Адреса в АП обычно выражают в шестнадцатеричной системе счисления. Так, например, диапазон адресов в АП с емкостью 64К будет записан как 0000h...FFFFh.

Адреса внешних устройств могут размещаться в отдельном адресном пространстве или же занимать часть общего. Выбор адресованной ячейки памяти или ВУ осуществляется схемами декодирования адреса. При управлении памятью и ВУ процессор должен вначале сформировать нужный адрес, который затем декодируется аппаратными средствами.

В МПС применяют несколько способов формирования адресов. Для систем с простыми процессорами характерны способы адресации, рассмотренные далее (см. Рисунок 1.6)*.*

При прямой адресации адрес операнда содержится в команде, подлежащей выполнению. Прямая адресация удобна, но удлиняет команды (увеличивает их разрядности), причем существенно, т. к. при значительных емкостях памяти разрядности адресов достаточно велики.

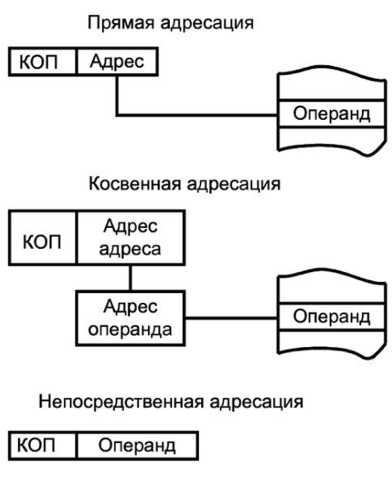


Рисунок 1.6 Способы адресации

При прямой регистровой адресации, когда операнд находится в одном из внутренних регистров процессора, адрес является малоразрядным, поскольку число внутренних регистров мало. В этом случае прямая адресация сочетается с компактностью команд и проявляет все свои достоинства. В частности, для адресации 8 регистров общего назначения достаточны трехразрядные адреса, а для адресации 4-х регистровых пар даже двухразрядные.

При косвенной адресации в команде явно или неявно указывается регистр процессора, содержащий адрес операнда (можно сказать, что в команде указывается или подразумевается короткий адрес адреса). Команда компактна, но для ее выполнения требуется предварительная настройка— загрузка регистра косвенного адреса (индексного регистра). Косвенная адресация особенно удобна при обработке списков, когда настройка производится однократно, а очередной адрес получается модификацией предыдущего (изменением его на единицу). В зависимости от действий, которые производятся над содержимым регистра при формировании адреса, различают несколько видов косвенной адресации. В частности, при простой косвенной адресации обращение к ячейке памяти производится по адресу, находящемуся в индексном регистре, т. е. никаких действий над его содержимым не производится. При относительной косвенной адресации адрес ячейки памяти получается суммированием содержимого индексного регистра и числа, задаваемого в команде.

При непосредственной адресации в команде содержится сам операнд. Использование различных видов адресации сокращает объем и время выполнения программ.

Команда может быть однобайтной, двухбайтной или трехбайтной (см. Рисунок 1.7). Первый байт содержит код операции КОП. сведения о способе адресации, а если команда однобайтная, то и адрес операнда. Размещение адреса в однобайтной команде возможно для операций типа "регистр-регистр" с короткими адресами. Второй байт двухбайтной команды содержит непосредственный операнд либо адрес ВУ, а второй байт трехбайтной команды — младший полуадрес операнда. Третий байт команды содержит старший полуадрес операнда или байт непосредственных данных при загрузке пары регистров.

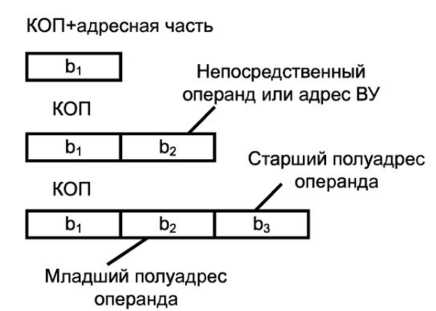


Рисунок 1.7 Форматы команд МП

## 1.3 Программируемая логическая интегральная схема

Программируемая логическая интегральная схема (ПЛИС) – электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования (проектирования). Для программирования используются программатор и IDE (отладочная среда), позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры: SystemVerilog, Verilog, VHDL, AHDL и др. Альтернативой ПЛИС являются:

* + Программируемые логические контроллеры (ПЛК);
  + Базовые матричные кристаллы (БМК), требующие заводского производственного процесса для программирования;
  + ASIC – специализированные заказные большие интегральные схемы, которые выгодно изготавливать крупными сериями;
  + Специализированные компьютеры, процессоры или микроконтроллеры, однако они медленнее ПЛИС из-за программного способа реализации алгоритмов.

Некоторые производители предлагают процессоры, которые можно модифицировать под конкретную задачу, а затем встроить в ПЛИС, таким образом можно уменьшить размер платы и упростить проектирование самой ПЛИС и увеличить быстродействие ПЛИС.

Проектирования на ПЛИС состоит из следующих этапов:

* + Задание электрической схемы или программы на специальных языках описания аппаратуры: SystemVerilog, Verilog, VHDL, AHDL и др.;
  + Логический синтез с помощью специальных программ-синтезаторов. В данных программах на основе модели, записанной на языке описания аппаратуры, создается список электрических соединений;
  + Размещение микросхемы ПЛИС на печатной плате с помощью САПР печатных плати выполнение согласования параметров передачи сигналов ПЛИС с периферийными модулями либо шиной компьютера;
  + Создание загрузочного файла прошивки;
  + Запись файла в память микросхемы ПЛИС.

ПЛИС имеет широкое применение при построении различных по сложности и по возможностям цифровых устройств, например:

* + Устройств с большим количеством портов ввода-вывода;
  + Устройств, выполняющих цифровую обработку сигнала;
  + Цифровой видео- аудиоаппаратуры;
  + Устройств, выполняющих передачу данных на высокой скорости;
  + Устройств, выполняющих криптографические операции, систем защиты информации;
  + Устройств, предназначенных для проектирования и прототипирования интегральных схем специального назначения (ASIC);
  + Устройств, выполняющих роль (коммутаторов) между системами с различной логикой и напряжением питания;
  + Реализаций нейрочипов;
  + Устройств, выполняющих моделирование квантовых вычислений.

В 1970 году компания «TI» разработала программируемые с помощью маски интегральные схемы, основанные на ассоциативном ПЗУ фирмы «IBM». Эта микросхема называлась TMS2000 и программировалась чередованием металлических слоёв в процессе производства интегральной схемы. TMS2000 имела до 17 входов и 18 выходов с 8 JK-триггерами в качестве памяти. Для этих устройств компания «TI» ввела термин PLA (programmable logic array – программируемая логическая матрица).

PAL (programmable array logic) – программируемый массив (матрица) логики. Разница между PLA и PLM состоит в доступности программирования внутренней структуры (матриц).

GAL (generic array logic) – это ПЛИС, имеющие программируемую матрицу «И» и фиксированную матрицу «ИЛИ».

CPLD (complex programmable logic device – сложные программируемые логические устройства) содержат относительно крупные программируемые логические блоки — макроячейки, соединённые с внешними выводами и внутренними шинами. Функциональность CPLD кодируется в энергонезависимой памяти, поэтому нет необходимости их перепрограммировать при включении. Может применяться для расширения числа входов/выходов рядом с большими кристаллами, или для предобработки сигналов (например, контроллер COM-порта, USB, VGA).

FPGA (field-programmable gate array) содержат блоки умножения-суммирования, которые широко применяются при обработке сигналов (DSP – digital signal processing), а также логические элементы (как правило, на базе таблиц перекодировки – таблиц истинности) и их блоки коммутации.

## 1.4 Программируемые пользователем вентильные матрицы (FPGA)

FPGA (Field Programmable Gate Arrays) – программируемые пользователем вентильные матрицы – наиболее обширный класс программируемых схем, обладающих максимальными функциональными возможностями. На их основе созданы системы на программируемом кристалле СнПК (SoPC, Systems on Programmable Chip). [4]

С учетом архитектурных особенностей и областей применения выделим следующие подклассы FPGA и систем на их основе:

* + FPGA невысокой и средней сложности;
  + FPGA высокой сложности и системы на кристалле;
  + Микроконтроллерные программируемые системы.

В разработке FPGA участвуют десятки фирм, ведущие среди них – Xilinx (пионер в создании FPGA), Intel (Altera). Microsemi (Actel), Atmel, Lattice Semiconductor, Cypress Semiconductor и др. Этими фирмами выпускаются семейства FPGA, которые по мере освоения новых технологических процессов (с интервалом в год-два) подвергаются модификациям и образуют серии, состоящие из родственных семейств.

FPGA, как и другие программируемые пользователем микросхемы (ППМС), выпускаются как полностью готовые, т. е. относятся к стандартной продукции, что сопровождается известными преимуществами – массовостью производства и снижением стоимости. Потребитель использует их, не обращаясь к изготовителю (выполняет программирование FPGA самостоятельно).

Благодаря регулярной структуре FPGA реализуются с уровнем интеграции, близким к максимальному. Вместе с тем, поскольку для средств программирования межсоединений требуются значительные затраты дополнительной площади кристалла, по количеству логических элементов, предоставляемых для реализации проекта, FPGA, как и другие ППМС, уступают полузаказным и, тем более, заказным схемам.

Отметим две области применения FPGA:

* + Отработка прототипов блоков и систем при их проектировании, даже если их конечная реализация рассчитана на другие средства;
  + Создание конечной продукции для изделий не слишком большой тиражности быстрыми и эффективными способами.

Вначале развитие ППМС было направлено на замену схем малого и среднего уровней интеграции и на перенос концепции вентильных матриц в область малотиражной аппаратуры, но в дальнейшем в связи с появлением перепрограммируемости стало ясно, что это нечто значительно большее. Это, в частности, видно из приведенных далее примеров.

Построение реконфигурируемых систем. При использовании аппаратуры встречаются ситуации, в которых те или иные блоки работают поочередно. Например, средства помехоустойчивого кодирования и декодирования при передаче и приеме данных. Обе функции (кодирование и декодирование) никогда не выполняются одновременно. Поэтому не обязательно иметь два устройства (кодер и декодер), а можно иметь одну перепрограммируемую схему с двумя разными конфигурациями, хранимыми в ПЗУ и поочередно загружаемыми в ПЛИС (программируемую логическую интегральную схему). В такой реконфигурируемой системе одна и та же аппаратная часть может выполнять различные преобразования после соответствующей перестройки.

Задачи логической эмуляции. При отладке устройств традиционно пользовались как изготовлением прототипа, так и программными моделями. Изготовление прототипа – сложная и дорогостоящая задача, но с его помощью можно вести тестирование с реальными сигналами и на высоких скоростях, наблюдая фактические возможности устройства. Программное моделирование лишено указанных достоинств, но проще и дешевле. Модели легко модифицируются для удаления ошибок в проекте и в них обеспечивается хорошая наблюдаемость процессов в объекте исследования.

Применение ПЛИС в задачах логической эмуляции дает сочетание достоинств обоих классических подходов. Система из микросхем программируемой логики легко создается и модифицируется, но может работать с реальными сигналами и частотами. В то же время программные модели и макетирование не зачеркиваются появлением перепрограммируемых микросхем. Создание программной модели может остаться наиболее быстрым и дешевым вариантом эмуляции, а макетирование отличается наиболее полным отображением свойств объекта. Таким образом, применение ПЛИС хорошо дополняет прежние методы разработки и тестирования схем.

Построение динамически реконфигурируемых систем. От простых переконфигурируемых систем, системы с динамической переконфигурацией (Run-Time Reconfiguration) отличаются тем, что в них требуется быстрая смена настроек. Обычная настройка с введением в микросхему последовательного потока битов или байтов занимает достаточно большое время. В динамически реконфигурируемых системах уже имеется (хранится) набор предварительно загруженных настроек, быстро сменяющих друг друга соответственно требованиям реализуемого алгоритма.

Динамически переконфигурируемая микросхема может иметь практически любое число настроек, которое ограничивается лишь емкостью памяти для их хранения. Устройства с динамической реконфигурацией уже используются практически и дают ожидаемый положительный эффект. Проблемы построения систем на микросхемах ПЛ с динамической переконфигурацией в настоящее время активно исследуются.

FPGA-процессоры. В современной литературе ставится вопрос о построении FPGA-процессоров с иными в сравнении с микропроцессорами свойствами. Алгоритмы работы процессора загружаются в FPGA принципиально подобно загрузке в память микропроцессорной системы выполняемой программы. Но в противоположность микропроцессорной системе, возникает сильно выраженный параллелизм на уровне логических блоков с простейшими операциями (типа воспроизведения функции от данного числа аргументов). Такие FPGA-процессоры могут давать хорошие результаты при параллельной обработке данных, где большое число переменных преобразуется сходным образом.

Эффективность схем с программируемой структурой стимулирует быстрый рост соответствующей отрасли промышленности и объемов их производства, а также научных исследований по развитию их архитектур, схемотехники и алгоритмов решения практических задач.

## 1.5 Ускоренное выполнение операции умножения

Методы ускорения умножения можно условно разделить на логические и аппаратные. Логические методы позволяют сократить время вычисления за счет более эффективных алгоритмов умножения, в частности за счет применения избыточных систем счисления и систем с основанием, которое больше двух. В аппаратных методах упор делается на схемное сокращение времени вычисления и суммирования частичных произведений. На практике оба этих подхода совмещают.

Наиболее известным и распространенным представителем этой группы логических методов ускорения операции умножения является алгоритм Бута. В основе алгоритма Бута лежит соотношение, характерное для последовательности двоичных цифр.

На практике большее распространение получила модификация алгоритма Бута, где количество операций сложения при любом сочетании единиц и нулей в множителе всегда равно n/2. В модифицированном алгоритме производится перекодировка цифр множителя из стандартной двоичной системы в избыточную систему, где каждое число представляет собой коэффициент, на который умножается множимое перед добавлением к СЧП. Одновременно анализируются три разряда множителя (два текущих и старший разряд из предыдущей тройки) и, в зависимости от комбинации 0 и 1 в этих разрядах, выполняется прибавление или вычитание множимого, прибавление или вычитание удвоенного множимого либо никакие действия не производятся.

Еще большее сокращение количества сложений может дать модификация, предложенная Леманом. Здесь, даже при наименее благоприятном сочетании цифр множителя, количество операций сложения не превышает величины n/2, а в среднем же оно составляет n/3. Суть модификации заключается в следующем:

* Если две группы нулей разделены единицей, стоящей в k-й позиции, то вместо вычитания в k-й позиции и сложения в (k + 1) позиции достаточно выполнить только сложение в k-й позиции;
* Если две группы единиц разделены нулем, стоящим в k-й позиции, то вместо сложения в k-й позиции и вычитания в (k + 1) позиции достаточно выполнить только вычитание в k-й позиции.

Действия, выполняемые на i-м шаге умножения, можно описать с помощью логических выражений, указанных в формуле 1.1 и в формуле 1.2.

, (1.1)

, (1.2)

где bi – цифра разряда множителя; di – двоичная переменная, единичное значение которой для соответствующего разряда множителя указывает на необходимость выполнения арифметического действия; si – знак арифметического действия.

При di=1, si=bi+1\*di. Если si=0, множимое прибавляется к СЧП, а при si=1 – оно вычитается из СЧП.

Так как у младшего разряда множителя отсутствует предыдущее значение, то считается что b0=0, а младший разряд записывают в b1.

# Глава 2 Практическая часть

## 2.1 Код программы с горизонтальным подходом к реализации

1 этап. Выбор машинной Команды

МК\_0

V1 – РАП=СЧК

V16 – Pd=0

V20 – PA=0

V23 – CМ=0

V33 – Рсд=0

МК\_1

V2 – СЧК=СЧК+1

V3 – РЧП=Чт(РАП)

МК\_2

V5 – РК=РЧП

V7 – РАП=РК(А2)

2 этап. Формирование исполнительного адреса и выбор операндов.

МК\_3

V3 – РЧП=Чт(РАП)

МК\_4

V6 – РАП=РК(А1)

V9 – Р2=РЧП

МК\_5

V3 – РЧП=Чт(РАП)

V32 – РБУФ=РАП

МК\_6

V8 – Р1=РЧП

V13 – Тзн=Р1[k]**⊕**P2[k]

V30 – РБУФ=РБУФ+1

V34 – СЧЦ=N

3 этап. Непосредственное выполнение операции в АЛУ.

МК\_7

V36 – Р1[k]=0

V37 – Р2[k]=0

И2=1? (да – И3=1? (да – МК\_9, нет – МК\_8), нет – МК\_15)

МК\_8

V27 – РB=Р1

МК\_9

V10 – PB[k]=1

V12 – PB[k-1:0]=!P1[k-1:0]

МК\_10

V14 – PB=PB+1

МК\_11

V39 – CМ[n:k+1]=РА+РВ

И4=1? (да – И6=1? (да – МК\_13, нет – МК\_12), нет – МК\_12)

МК\_12

V24 – СМ=СМ>>1

МК\_13

V24 – СМ=СМ>>1

МК\_14

V11 – СМ[n]=1

МК\_15

V22 – CМ[n:k+1]=РА

МК\_16

V24 – СМ=СМ>>1

И5=1? (да – МК\_14, нет – МК\_17)

МК\_17

V17 – Pd=Pd>>1

V18 – Pсд=P2[0]

V25 – PA=CМ[n:k+1]

И1=1? (да – МК\_18, нет – МК\_19)

МК\_18

V19 – P2=P2>>1

V35 – CЧЦ=СЧЦ-1

4 этап. Запись результата.

МК\_19

V38 – СМ[n]=Тзн

МК\_20

V28 – РЧП=CМ[n:k+1]

МК\_21

V4 – Зп(РЧП)

V21 – РАП=РБУФ

МК\_22

V26 – РЧП=CМ[k:0]

МК\_23

V4 – Зп[РЧП]

где, МК – микрокоманда, РАП – регистр адреса памяти, Pd – регистр di, РА – регистр А, СМ – регистр суммы, Рсд – регистр сдвига, СЧК – счётчик команд, СЧЦ – счётчик цикла, РК – регистр команд, Чт – чтение, Зп – запись, РЧП – регистр чтения памяти, Р1 – регистр 1, Р2 – регистр 2, РБУФ – регистр буфер, Тзн – триггер знака, РА – регистр А, РВ – регистр В, И – индикатор условия.

## 2.2 Блок-схема алгоритма умножения

На рисунке 2.1 и рисунке 2.2 изображена блок-схема алгоритма, применяемого в разрабатываемом вычислительном устройстве.



Рисунок 2.1 Блок-схема алгоритма



Рисунок 2.2 Блок-схема алгоритма (продолжение)

## 2.3 Структурная схема

На рисунке 2.3 приведена структурная схема вычислительного устройства для команды умножения для чисел с фиксированной запятой методом Лемана.

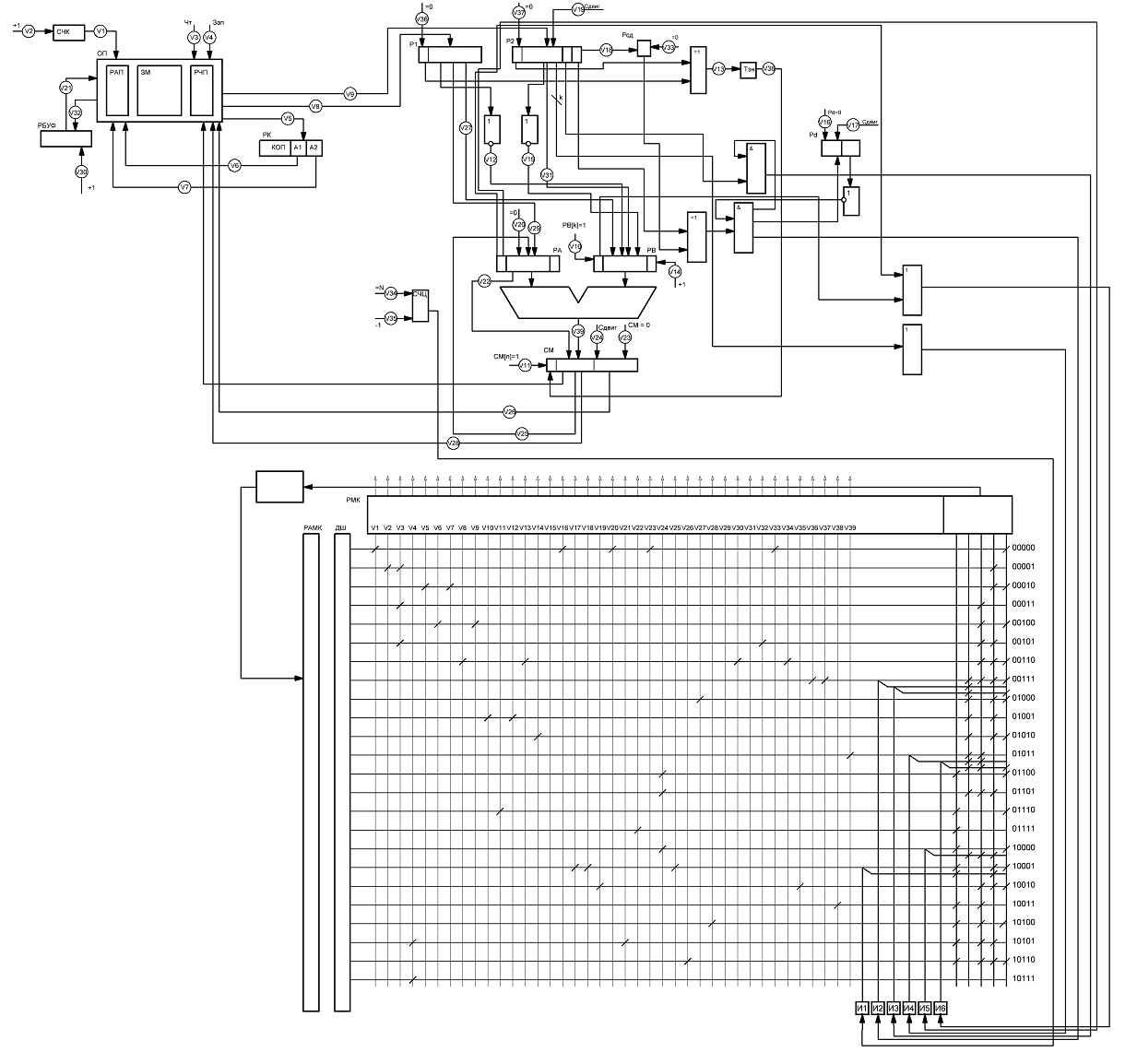


Рисунок 2.3 Структурная схема

## 2.4 Последовательность микрокоманд

В таблице 1 приведен список микрокоманд разработанного вычислительного устройства.

Таблица 2.1

Список микрокоманд вычислительного устройства

|  |  |  |
| --- | --- | --- |
| Микрокоманда | Управляющий сигнал | Следующая микрокоманда |
| 0 | V1 V16 V20 V23 V33 | 1 |
| 1 | V2 V3 | 2 |
| 2 | V5 V7 | 3 |
| 3 | V3 | 4 |
| 4 | V6 V9 | 5 |
| 5 | V3 V32 | 6 |
| 6 | V8 V13 V30 V34 | 7 |
| 7 | V36 V37 | 15 или 8 или 9 |
| 8 | V27 | 11 |
| 9 | V10 V12 | 10 |
| 10 | V14 | 11 |
| 11 | V39 | 12 или 13 |
| 12 | V24 | 17 |
| 13 | V24 | 14 |
| 14 | V11 | 17 |
| 15 | V22 | 16 |
| 16 | V24 | 17 или 14 |
| 17 | V17 V18 V25 | 18 или 19 |
| 18 | V19 V35 | 7 |
| 19 | V38 | 20 |
| 20 | V28 | 21 |
| 21 | V5 V21 | 22 |
| 22 | V26 | 23 |
| 23 | V5 | 0 |

## 2.5 Написание микропрограммы

Открываем программу Quartus Prime 17.1 Lite Edition и создаем новый проект (см. Рисунок 2.4), нажатием на “New Project Wizzard”.

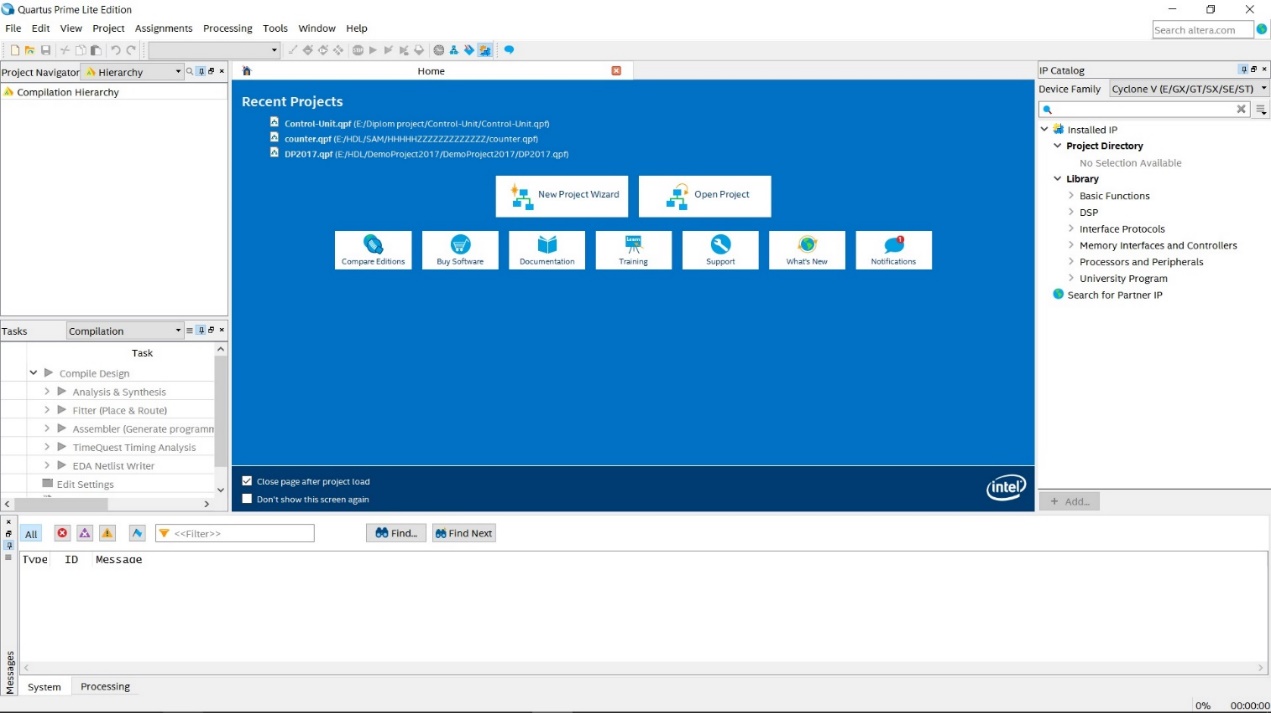


Рисунок 2.4 Окно программы Quartus Prime 17.1 Lite Edition

Появится мастер создания проекта (см. Рисунок 2.5). В нем нажимаем кнопку “Next” и перейдем к настройкам имени и директории проекта (см. Рисунок 2.6).

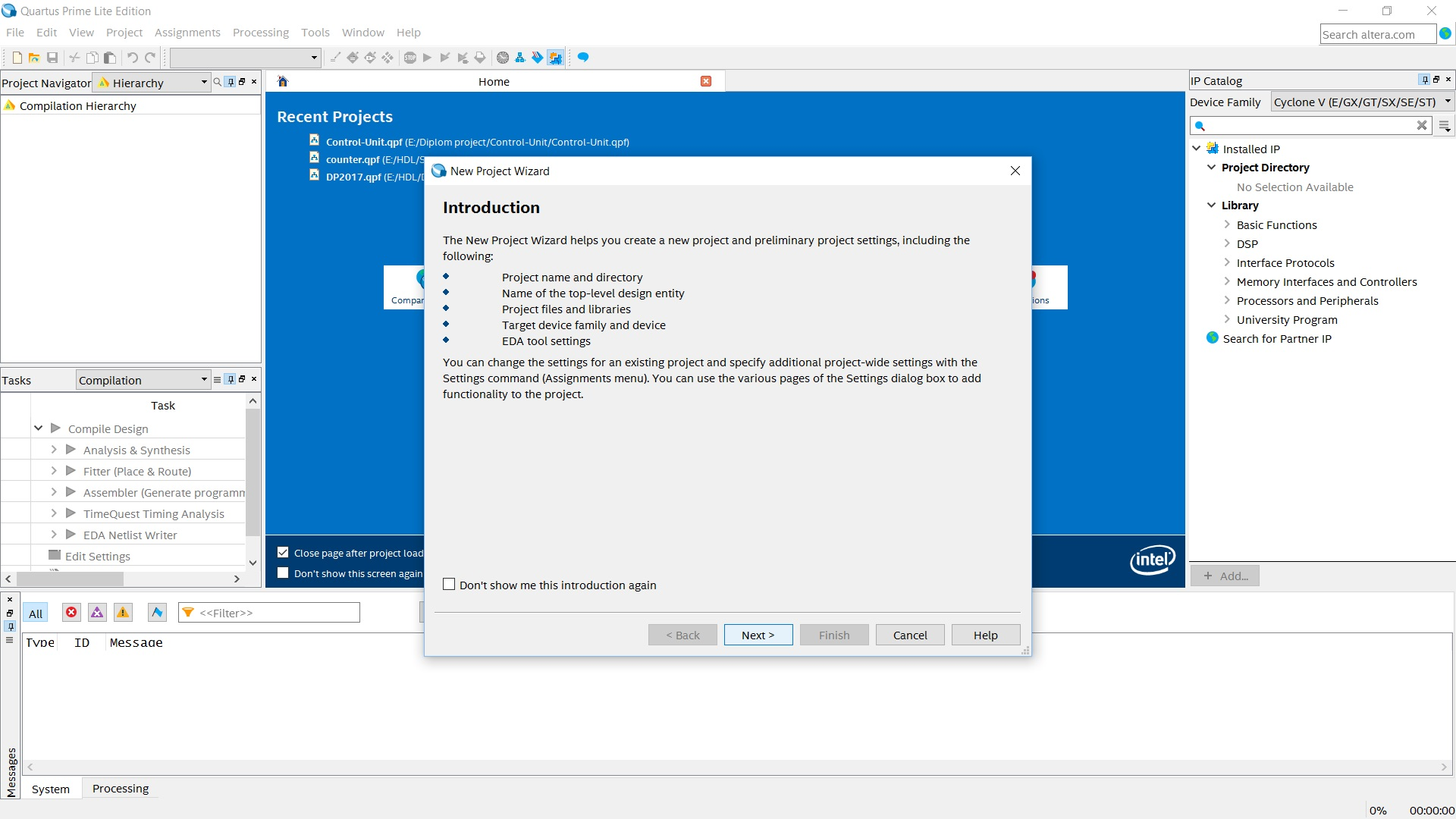


Рисунок 2.5 Окно мастера создания проекта

Назовём проект “Control-Unit”, а также допишем к предлагаемой директории проекта “/Control-Unit”, что позволит создать отдельную папку для проекта.

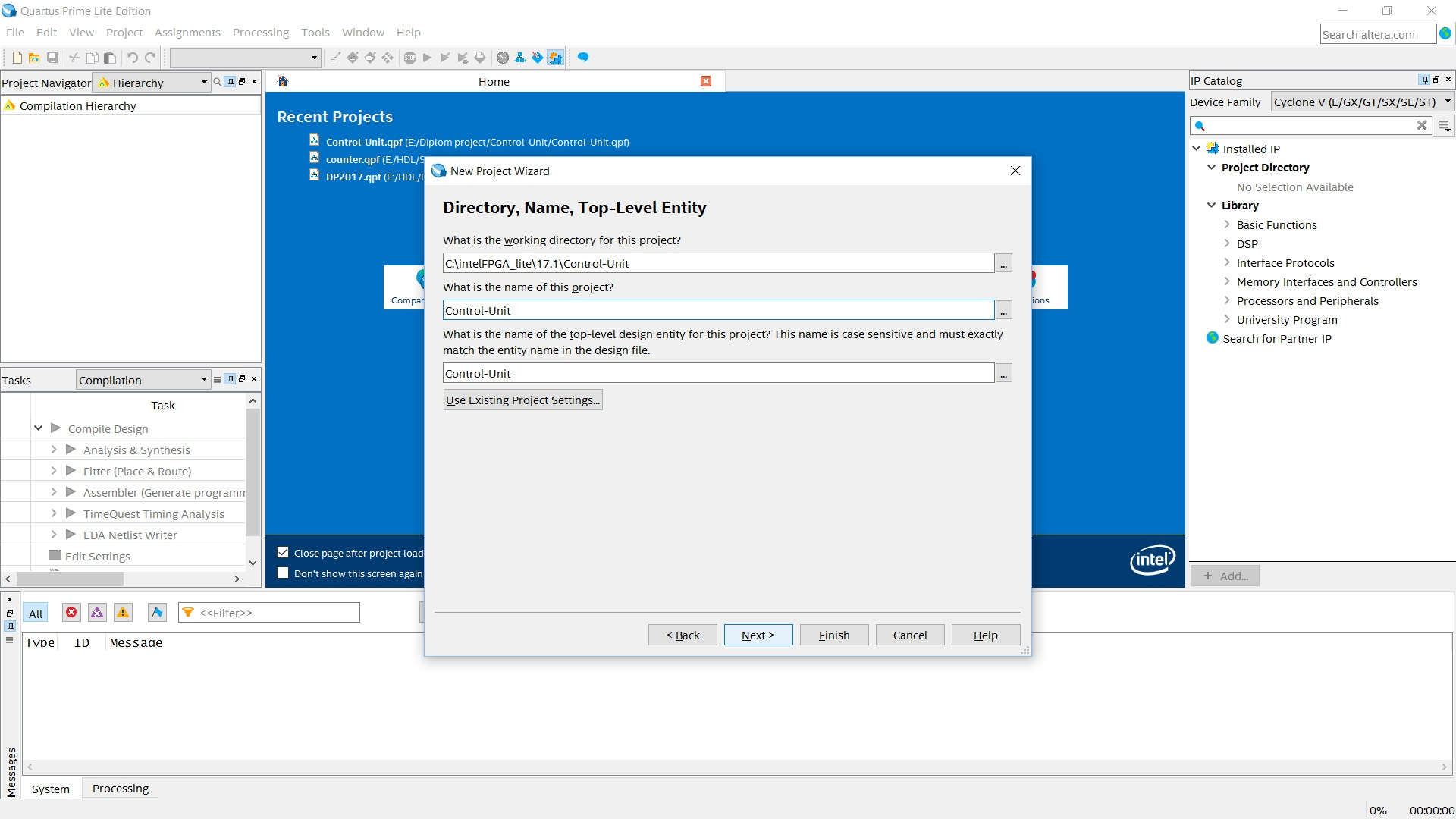


Рисунок 2.6 Окно мастера создания проекта (Директория, имя, объект верхнего уровня)

Далее нажимаем кнопку “Next”, чтобы перейти к выбору типа проекта (см. Рисунок 2.7). Создаем новый проект, поэтому ничего не меняем и нажимаем кнопку “Next”.

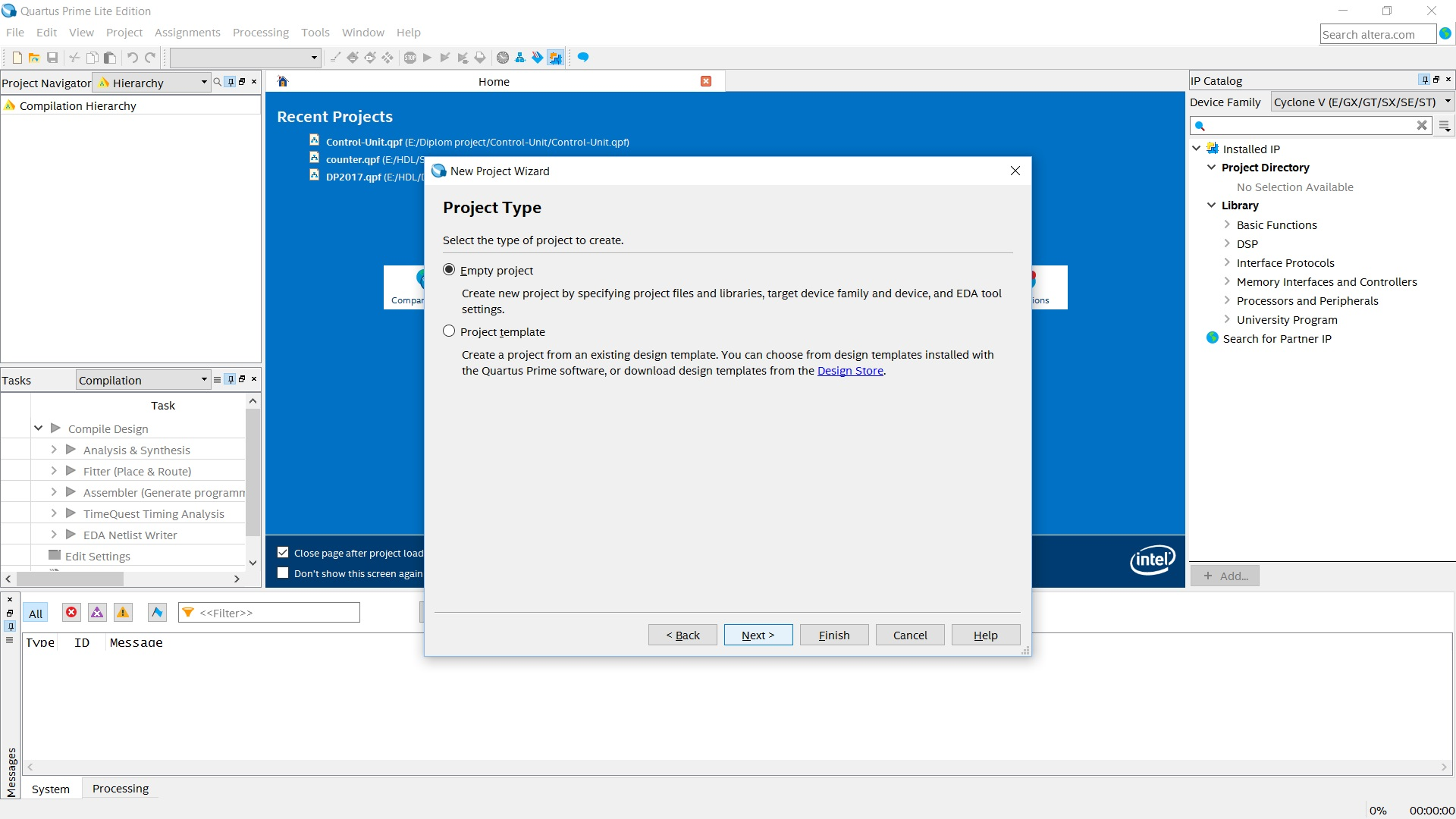


Рисунок 2.7 Окно мастера создания проекта (тип проекта)

Теперь предлагается добавить файлы в проект (см. Рисунок 2.8), но поскольку проект новый и добавить нечего, то пропускаем данный шаг и нажимаем кнопку “Next”.

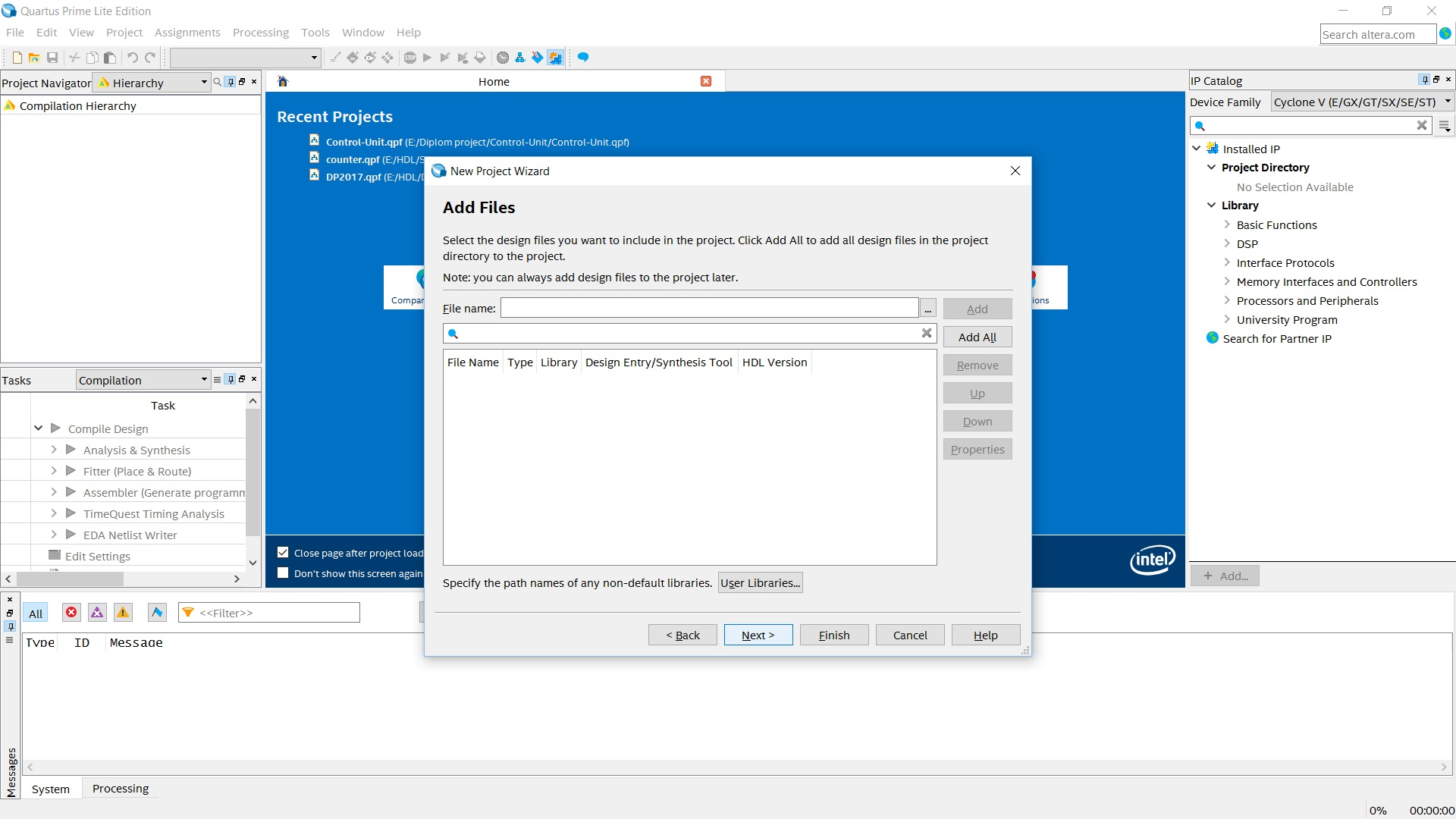
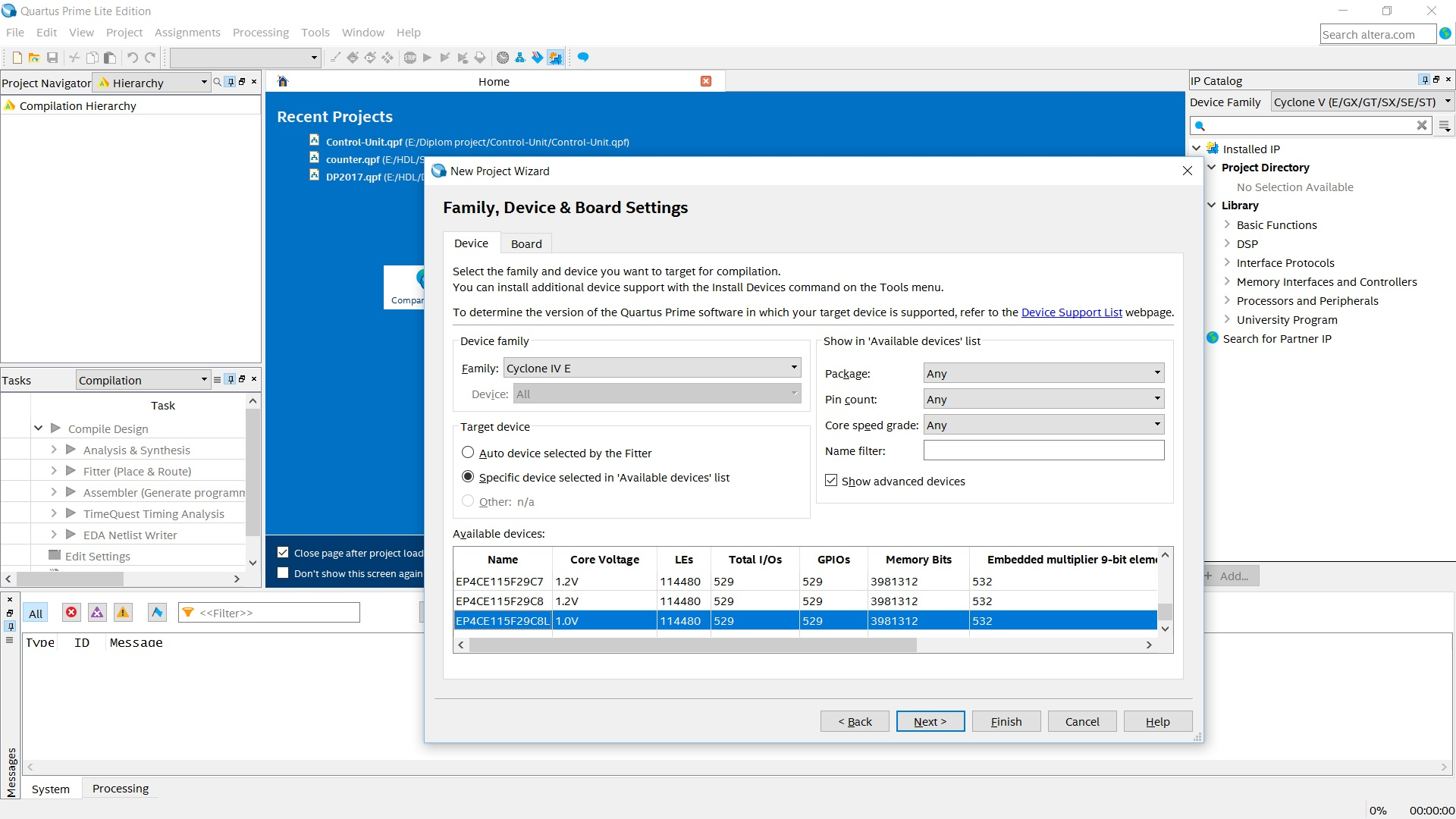
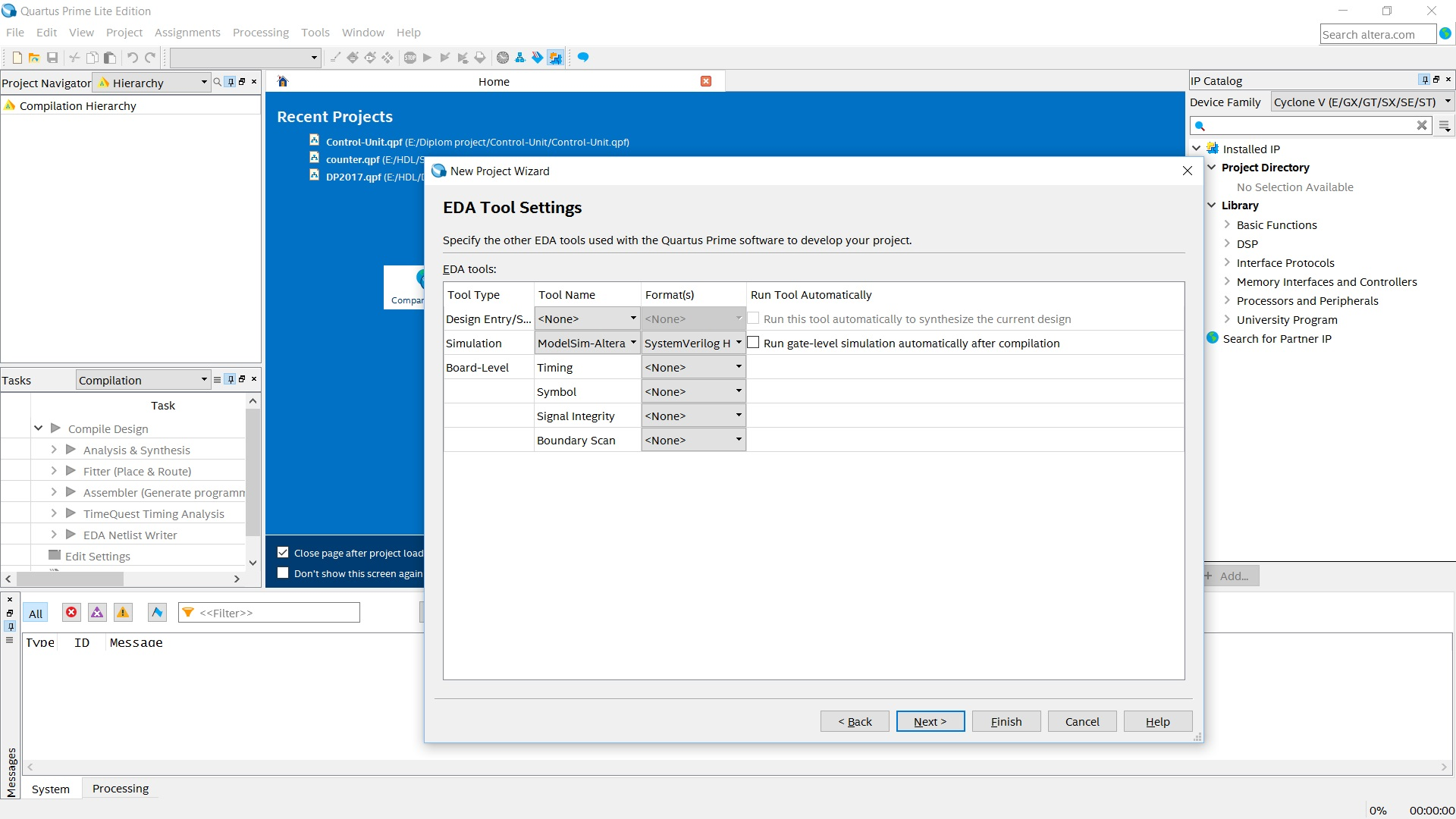


Рисунок 2.8 Окно мастера создания проекта (добавление файлов)

Откроется окно выбора семейства устройств, и настройки платы (см. Рисунок 2.9). В дипломном проекте в качестве тестовой платы используется плата DE2-115, принадлежащая к семейству Cyclone IV E. В списке “Available devices” выбираем модель микросхемы EP4CE115F29C8L – эта модель используется в плате DE2-115.

  
Рисунок 2.9 Окно мастера создания проекта (семейство устройства, и настройки платы)

Нажимаем “Next” и переходим к настройкам инструментов EDA (см. Рисунок 2.10). В строке “Simulation” выбираем инструмент ModelSim-Altera, а формат файлов “SystemVerilog HDL”. Нажимаем “Next.”

  
Рисунок 2.10 Окно мастера создания проекта (настройкам инструментов EDA)

Попадаем на последнюю вкладку (см. Рисунок 2.11). Здесь показаны все выбранные настройки, с которыми будет создан наш проект.

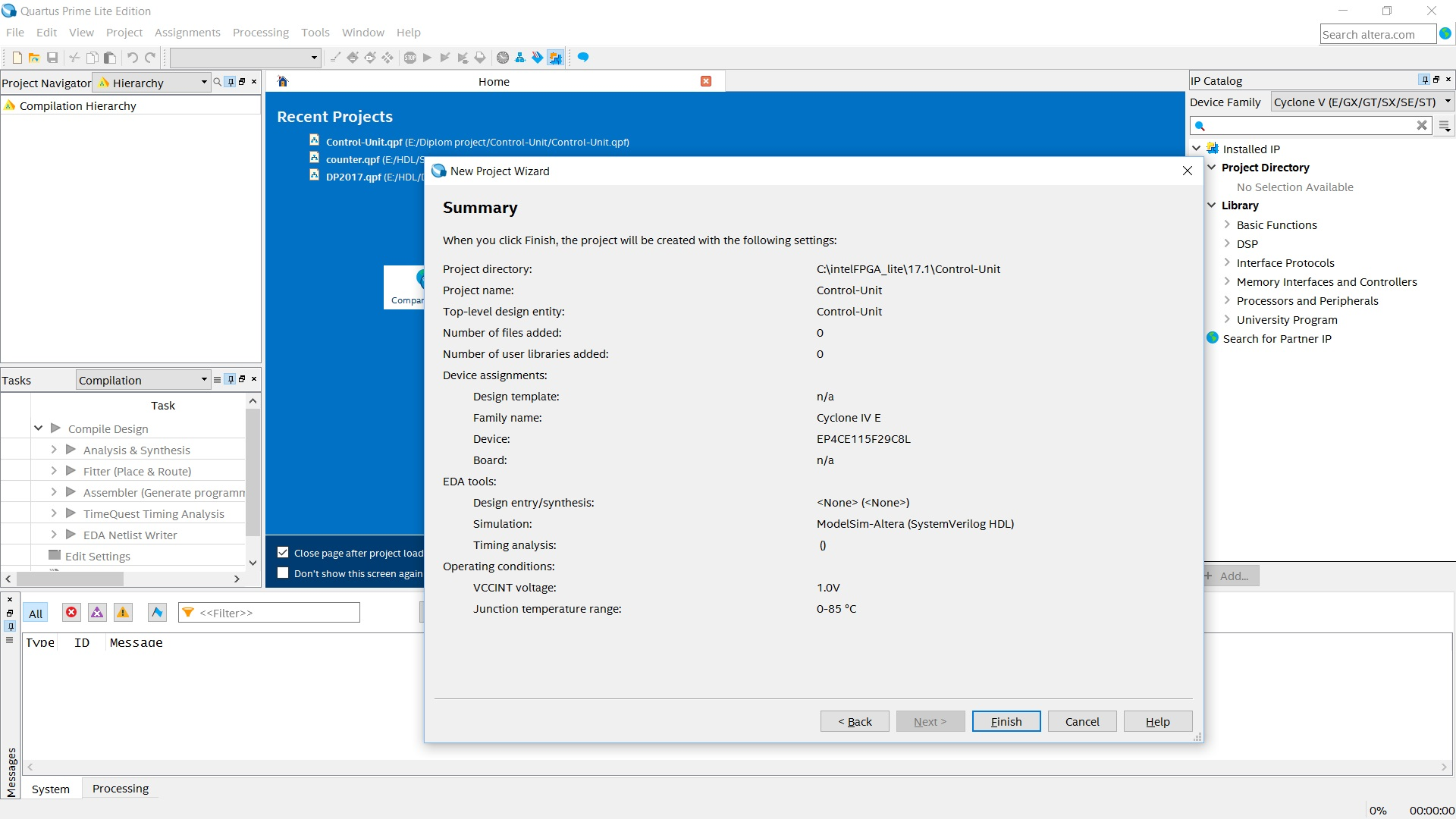


Рисунок 2.11 Окно мастера создания проекта (настройкам инструментов EDA)

Нажимаем кнопку “Finish”. Мастер настройки закроется и теперь необходимо создать файл для программы. Для этого на панели быстрого доступа нажимаем File/New… и появится окно с выбором файлов для создания (см. Рисунок 2.12), в котором выбираем “SystemVerilog HDL File” и нажимаем кнопку “OK”. Созданный файл сразу же откроется во встроенном редакторе кода (см. Рисунок 2.13). Здесь записываем код программы (см. Приложение 1). После его написания выполняем компиляцию кода встроенным компилятором. Если ошибок не будет обнаружено создастся файл прошивки, в противном случае в окне сообщений будет записан результат компиляции со всеми ошибками и предупреждениями. Для отладки программы пишем тест (см. Приложение 2)



Рисунок 2.12 Окно создания нового файла

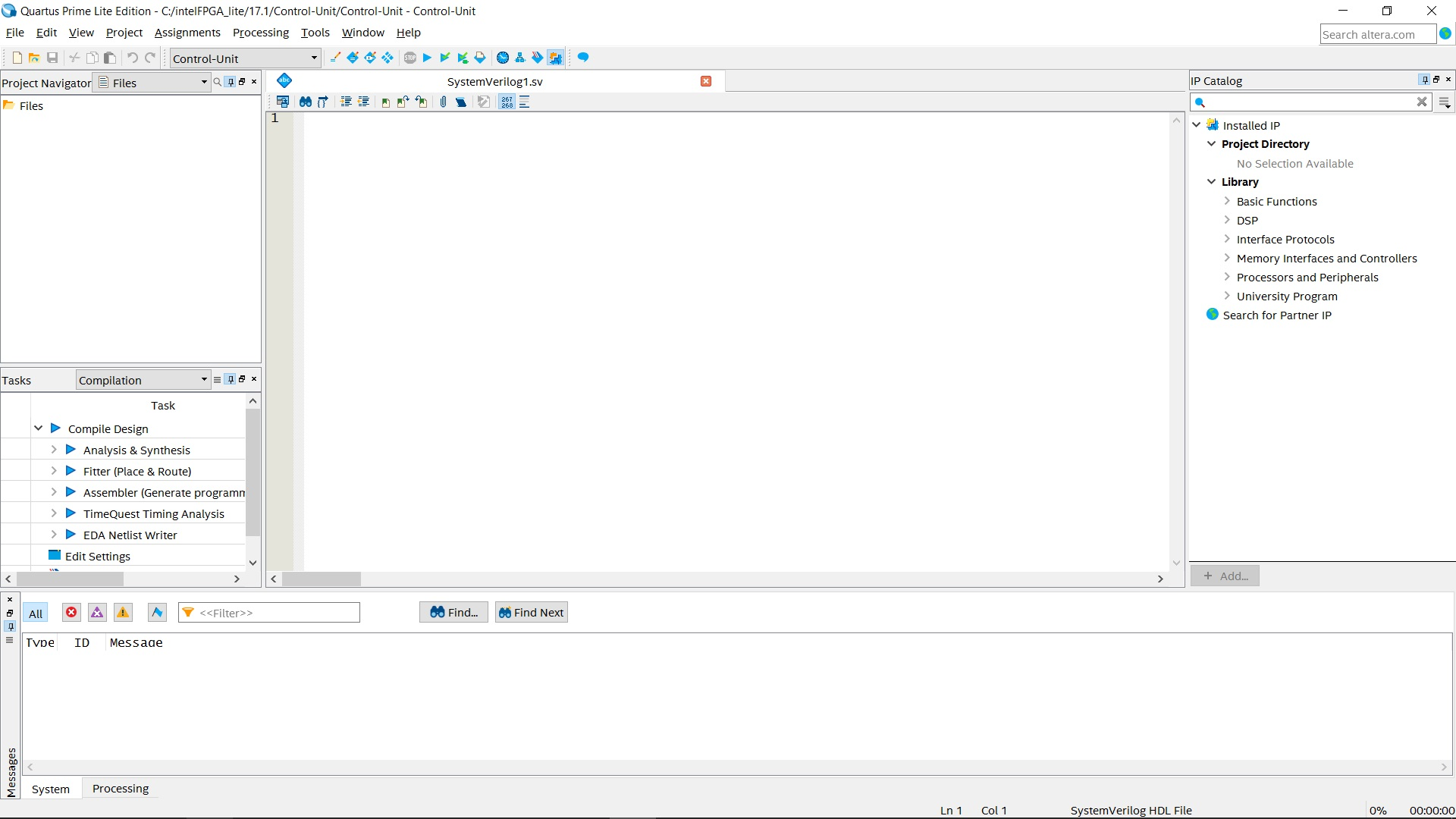


Рисунок 2.13 Редактор кода

# ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта все поставленные цели и задачи дипломного проекта были выполнены в полном объёме. Разработано микропрограммное устройство управления вычислительного устройства для команды ускоренного умножения чисел с фиксированной запятой методом Лемана и его реализация на ПЛИС. В процессе разработки были определены и проанализированные исходные данные, разработан алгоритм решения поставленных перед вычислительным устройством. В ходе анализа алгоритма была предложена структурная схема разрабатываемого вычислительного устройства, определены его основные функциональные блоки. Реализована требуемая функциональность на языке SystemVerilog в виде синтезируемой логической модели. Полученная модель была успешно синтезирована в технологический базис целевой микросхемы ПЛИС Intel (Altera) DE2-115. В качестве дальнейшего развития темы дипломного проекта, можно провести сравнительный анализ способов ускорения умножения.

Следует особо отметить, что разработанная синтезируемая логическая модель является технологически независимой от выбранной элементной базы и может быть синтезирована для размещения в любой микросхемt ПЛИС достаточной емкости и быстродействия или выполнена в виде заказной СБИС. Это позволяет в дальнейшем повысить быстродействие разработанного узла, минимизировать его энергопотреблении, увеличить стойкость к внешним воздействиям.

В Теоретической части было проанализировано следующее:

* арифметико-логическое устройство, его структура и принцип его работы;
* особенности микроприказов в микропрограммном устройстве управления;
* блок устройства управления, его типы и способы реализации;
* адресные пространства, способы адресации, форматы команд;
* общие сведения о программируемых логических интегральных схемах (ПЛИС);
* особенности класса программируемых схем – FPGA;
* алгоритм ускоренного умножения, предложенный Леманом.

В практической части было разработано микропрограммное устройство управления вычислительного устройства для команды ускоренного умножения чисел с фиксированной запятой методом Лемана и его реализация на ПЛИС. Проведена отладка модели с помощью разработанной системы тестирования. В ходе отладки были устранены ошибки в коде модели и проверено ее соответствие исходным требованиям.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Техническая литература:

1. Угрюмов, Е.П. Цифровая схемотехника 3-е издание: учебное пособие для вузов/ Е.П. Угрюмов. – Санкт-Петербург: БХВ-Петербург, 2010. – 816 с.;
2. Брехов, О.М Принципы построения процессоров для авиационных комплексов; учебное пособие/ О.М. Брехов. – Москва: МАИ, 1984. – 71 с.;
3. Звонарева, Г.А., Корнеенкова, А.В. Учебное пособие для проведения практических занятий, лабораторных работ, выполнения курсовых и расчётно-графических работ по курсам: «Организация ЭВМ», «Архитектура ЭВМ», «ВМ, системы и сети»: учебное пособие/ Г.А. Звонарева, А.В. Корнеенкова. – Москва: МАИ, 2016. – 36 с.;
4. Орлов, С.А., Цилькер, Б.Я. Организация ЭВМ и систем: учебник для вузов/ С.А. Орлов, Б.Я. Цилькер. – Санкт-Петербург: Питер, 2011. – 688 с.