МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра конструирования и технологии электронно-вычислительных средств

ИСПОЛЬЗОВАНИЕ КОМАНДНЫХ ФАЙЛОВ ДЛЯ МОДЕЛИРОВАНИЯ ЭЛЕМЕНТОВ ПРОГРАММИРУЕМОЙ ЛОГИКИ

Методические указания по выполнению лабораторной работы по дисциплине
«Информационные технологии проектирования электронно-вычислительных средств»
для студентов специальности 210202.65

Курск 2012

УДК 681.325

Составитель: Т. И. Аспидова

Рецензент
Кандидат технических наук, доцент *О. Г. Бондарь*

Использование командных файлов для моделирования элементов программируемой логики : методические указания по выполнению лабораторной работы по дисциплине «Информационные технологии проектирования ЭВС/ Юго-Зап. гос. ун-т; сост.: Т. И. Аспидова. Курск, 2012. 19 с.

Содержатся теоретические сведения, касающиеся использования командных файлов при моделировании элементов программируемой логики. Указывается порядок выполнения лабораторной работы.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальностям автоматики и электроники (УМО АЭ).

Предназначены для студентов специальности 210202.65.

Текст печатается в авторской редакции

Подписано в печать . Формат 60× 84 1/16.
Усл. печ. л. 1,10. Уч.-изд. л. 1,00. Тираж 30 экз. Заказ . Бесплатно.

Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение командных файлов для моделирования элементов программируемой логики в среде САПР Foundation Series.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

При развертывании во времени событий и процессов, происходящих в реальном устройстве и его модели, различают четыре типа времени.

1. Реальное время — в нем протекают реальные процессы в реальном устройстве.

2. Системное (или модельное) время — абстрактное время, в котором протекают процессы или события при их моделировании на ЭВМ. В программах это время представляется отдельной переменной, которую пользователь или разработчик программы может менять по своему усмотрению. Применение системного времени позволяет изменять масштаб реального времени или даже останавливать его для изучения событий путем более тщательного расчета в отдельные моменты времени. Системное время — это реальное время, учитываемое в программе в некотором масштабе(секунды, микросекунды, наносекунды и т. д.). На распечатках или на экране дисплея это время отображается цифрами, обозначающими реальное время, но расположение этих цифр может быть произвольным — редким, частым, с неравномерным расстоянием друг от друга — в зависимости от шага печати, заданного в программе вывода данных.

3. Машинное время — реальное время моделирования на ЭВМ. Обычно в нем выделяют время трансляции программы (перевода с языка програмирования в коды команд ЭВМ) и процессорное время (время исполнения задачи после трансляции). Машинное время регистрируется специальным устройством ЭВМ — таймером.

4. Автоматное время — абстрактное время, обычно выраженное целыми числами 0, 1, 2, ... и регистрирующее лишь порядок следования событий независимо от их длительности.

Механизм задержки позволяет представлять время распространение сигналов в описываемых систем.

Имеется два механизм задержки: инерциальная задержка (значение по умолчанию) и транспортная задержка.

Транспортная задержка определена зарезервированным словом транспорт (transport) и характерна для линий передачи. Новое значение сигнала назначено с заданной задержкой независимо от ширины импульса в форме волны (то есть сигнал распространяется через линию).

Инерциальная задержка определена зарезервированным словом инерциальный (inertial) и используется для модели устройства, которая является по существу инерциальной. Практически это означает, что импульсы более короткие, чем специфицированное время переключения - не передаются

Если механизм задержки не определен, то по умолчанию это инерциальная задержка.

Этапы функционального и временного моделирования являются обязательными при проектировании элементов программируемой логики.

Командные файлы логического симулятора представляют собой задания на моделирование.

Это текстовые файлы, состоящие из набора так называемых макрокоманд, описанных ниже.

Преимуществом командных файлов является их гибкость, отсутствие необходимости задания воздействий при каждом сеансе моделирования и простота формирования сложных тестовых воздействий.

Предпочтительным средством для создания таких файлов является Script Editor, вызов которого осуществляется непосредственно из программы моделирования.

Выполнение командного файла производится через меню Execute самого Script Editora.

Минимально необходимый набор макрокоманд, употребляемых в командных файлах, приведены ниже. Если вы не уверены в синтаксисе команды, обратитесь к режиму Macro Assistant, где можно найти шаблоны и примеры применения макрокоманд, а также образец макрофайла.

After

Формат: after time do (cmd1;cmd2;...cmdn)

Сокращение: нет

Действие:

AFTER задерживает выполнение перечисленного в скобках набора команд до того, как будет достигнуто указанное время (оно задается относительно текущего модельного времени). Команды выполняются в порядке перечисления. Однако, если в одно и тоже время выполняется более одной последовательности команд, определенных AFTER, порядок выбора последовательности не определен.

Команды в скобках должны быть разделены точкой с запятой

Пример:

after 10ns do (assign DATA0 1; low s3)
sim

После 10ns моделирования (относительно текущего времени), сигналу DATA0 присваивается константное значение «1», сигнал s3 устанавливается в низкий уровень.

Assign

Формат: assign signal_or_bus new_state
assign signal_or_bus <data_file

Действие:

Значение второго параметра присваивается первому непосредственно во время моделирования. В качестве второго параметра может быть использовано конкретное состояние сигнала или значение сигнала из файла, содержащего тестовые вектора.

Пример:

assign U1.A1 Z | задать состояние High_Z на контакте U1.A1

assign ADBUS1 AB03\H | задать состояние AB03 на шине ADBUS1
assign DATA <data_bus.dat | задать сигналу DATA значение, взятое из файла data_bus.dat

Break

Формат: break
break signal_or_vector
break signal_or_vector event
break signal_or_vector event do (sequence_of_macro_commands)

Действие:
Эта макрокоманда очищает все точки останова, являющиеся результатом моделирования. Она также останавливает моделирование в том случае, если выполнены заданные условия для заданного сигнала или тестового вектора.

· break – очищает все точки останова
· break signal_or _vector - очищает все точки останова для указанного сигнала, шины или контакта.
· break signal_or_vector event – останавливает процесс моделирования в том случае, если заданное условие выполнено. После этого выполняется следующая макрокоманда из файла. Если эта команда является частью цикла, то выполнение цикла прерывается и выполняется следующая макрокоманда.
· break signal_or _vector event do (sequence_of_macro_commands) - останавливает процесс моделирования в том случае, если заданное условие выполнено breaks the simulation when the selected breakpoint condition is met, and executes the macro и выполняет последовательность макрокоманд, перечисленных в скобках после «DO». После выполнения этой последовательности процесс моделирования продолжается.

Формат записи условий:

· ? - означает, что любое изменение является заданным условием

· condition - условие выполнено при достижении заданного состояния

· condition1-condition2 - условие выполнено при переходе из condition1 к condition2

Примеры:

break - удалить все точки останова
break clock - удалить все точки останова для сигнала clock
break clock ? - остановить моделирование после любого изменения состояния сигнала clock
break enable Z-X - остановить моделирование после того, как состояние сигнала enable изменится от High_Z к Unknown_X

break clock 0-1 do (assign <DATAfile.dat; print) - остановить моделирование после изменения состояния сигнала clock от низкого к высокому уровню и присвоить сигналу DATA новое значение из файла file.dat. Затем вывести состояние всех сигналов.

Cycle

Формат: cycle
cycle n

Действие:
Симулятор выполняет заданное количество циклов (Задание тактовой частоты см. макрокоманду CLOCK). Если тактовая чистота не задана, выполняется n шагов моделирования. Если задано несколько тактовых частот, то выполняется n циклов наибольшей длительности. Если значение n не задано, то

Пример:

Cycle 5
c 5

Эта макрокоманда задает выполнение пяти циклов тактовой частоты. Значение частоты должно быть определено ранее макрокомандой Clock.

Clock

Формат : clock signal_or_bus_specification signal_state_1 ... signal_state_n

Действие:
Макрокоманда clock задает периодическую последовательность логических состояний, которая применяется к сигналу или шине. Каждое состояние сигнала длиться в течение одного шага моделирования. Присваивая шагу различные значения, можно получать различные тактовые частоты

Задание таким образом тактовой частоты вызывает автоматическую синхронизацию всех сигналов и шин, сгенерированных макрокомандой Pattern. Каждое изменение состояния сигнала, заданное этой командой, выполняется программой моделирования в начале цикла тактовой частоты.
Clock causes automatic synchronization of all signals and buses generated by the Pattern macro command. Each new signal state, defined by the Pattern command, is applied to the simulator at the beginning of the clock cycle. If the clock cycle is comprised of four simulation steps (or logical states), then the Pattern signals will be applied every four simulation steps.

Пример:

step 20ns | задание шага моделирования
clock c 0 1 0 1 1 0 0 1 | задание тактовой частоты (период 160ns)
pattern b 0 1 | задание шаблона для входного сигнала b
low a | установка входного сигнала a в низкий уровень
run | запуск моделирования

Delete Signals

Формат: delete_signals list_of_signals
delete_signals

Действие:
Эта макрокоманда предназначена для удаления сигналов, перечисленных в list_of_signals из окна просмотра временных диаграмм. Если перечень сигналов не указан, то то будут удалены все сигналы.

Пример:

```
delete_signals Q1 Q2 Q3
dels
```

Delete Waveforms

Формат: delete_waveforms list_of_signals
delete_waveforms

Сокращение: delw

Действие:
Эта макрокоманда предназначена для удаления временных диаграмм сигналов, перечисленных в list_of_signals из окна просмотра временных диаграмм. Если перечень сигналов не указан, то будут удалены все временные диаграммы.

Пример:

```
delw clock data
delete_waveforms
```

Every

Формат: every time do (cmd1;cmd2;...cmdn)

Сокращение: none

Действие:
Набор макрокоманд, перечисленный в скобках, выполняется в заданном порядке по завершению каждого временного интервала заданной длительности. Для того, чтобы эта макрокоманда выполнялась, она должна предшествовать таким командам моделирования, как Sim, Run и Cycle. Если в один и тот же момент времени должны быть выполнены две макрокоманды Every, то последовательность их выполнения не определена.

Пример:

```
w DATA0                 | выбор сигнала, состояния которого будут выводится
every 10ns do (display)     | вызов команды every
sim                         | начало моделирования
time = 50ns  DATA0 = 0    | результаты выполнения команды display
time = 60ns  DATA0 = 1    | выдаются в окно команд
time = 70ns  DATA0 = 0    | каждые 10ns
time = 80ns  DATA0 = 1
time = 90ns  DATA0 = 0
time = 100ns  DATA0 = 1
```

Global Reset

Формат: global_reset

Действие:
Макрокоманда Global_reset macro выполняет операцию глобального сброса.

High

Формат: high signal_or_bus_specification

Сокращение: h

Действие:
Устанавливает заданный сигнал или линию шины (определяемые параметром signal_or_bus_specification) в высокое состояние.

Пример:

h U21.INPUT

Устанавливает высокое логическое состояние на контакте U21.INPUT.

Pattern

Формат: pattern signal_or_bus_listing explicit_signal_state_listing
pattern signal_or_bus_listing file_signal_state_listing

Сокращение: pat

Действие:
Заданные этой макрокомандой сигналы и шины устанавливаются в состояния, перечисленные в списке explicit_signal_state_listing или в файле file_signal_state_listing. Переход от одного состояния к другому происходит в начале тактового цикла. Если тактовая частота не задана, то переход производится в начале цикла моделирования (Step).

Примеры:

pattern U23.Y2 0 1 AB\H Z X H 12\D

контакт U23.Y2 устанавливается последовательно в следующие логические состояния:
L (параметр 0)
H (параметр 1)
H (младший значащий бит параметра AB\H)
High_Z (параметр Z)
Unkn_X (параметр X)
L (младший значащий бит 12\D).

Если вместо одиночного сигнала была задана шина, тоAB\H и 12\D интерпретируются как шестнадцитиричные значения состояния шины

pat B34.CLK < clock.dat

В этом примере контакт B34.CLK устанавливается в состояния, последовательно перечисленные в файле clock.dat.

Radix

Формат: radix base vector1 vector2 ... vectorN

Сокращение: none

Действие:
Эта макрокоманда задает систему счисления для просмотра и печати состояний каждой из шин. В качестве параметра base могут быть использованы следующие сокращения:

- bin – двоичное представление
- oct – восьмиричное представление
- dec – десятичное представление
- hex – шестнадцатиричное представление

Пример:

radix oct Data_In
radix hex Address_Bus

Состояние шины Data_In будет представлено в восьмиричном формате, а шины Address_Bus – в шеснадцатиричном.

Restart

Формат: restart

Сокращение: none

Действие:
Выполняет операцию включения питания.
Пример:

restart

Выполняет операцию включения питания. Настройка этой операции должна быть заранее произведена через меню программы моделирования (в том числе состояния по включению питания - All Low, All High, Random, Unknown).

Run

Форматs: run n
run

Сокращение: none

Действие:
Начинает моделирование. Программа моделирования выполняет n циклов тактовой частоты и устанавливает сигналы в состояния, определеннее макрокомандами Waveform и Pattern . Если тактовая частота не задана, выполняется n шагов моделирования (steps). Если параметр n не задан, То процесс моделирования выполняется до того момента, когда выполнится наиболее длинная заданная последовательность установок состояний сигнала.

Save Waveform

Формат: save_wave testvector_file_name

Сокращение: swv

Действие:
Эта макрокоманда сохраняет временные диаграммы (тестовые вектора) в файле, наименование которого задается параметром testvector_file_name. Информация сохраняется в двоичном формате.

Примеры:

save_waveform aftclk
swv aft25ns

Set Simulation Mode

Формат: set_mode sim_mode

Сокращение: simmd

Действие:

Эта макрокоманда устанавливает режим для последующего моделирования. Параметр sim_mode задает один из следующих режимов:

functional - функциональное моделирование,
timing - моделирование с учетом временных задержек,
glitch - «glitch»,
unit - моделирование с единичными временными задержками.

Примеps:

Set_mode glitch
Simmd unit

Set Stimulator

Формат: set_stim stim_symbol signal_name_list

Сокращение: stim

Действие:

Эта макрокоманда назначает сигналам или векторам, указанным параметром signal_name_list, предопределенные источники стимулирующих воздействий.

Наименования сигналов или шин при перечислении разделяются пробелом.

Значение параметра stim_symbol является наименованием одного из следующих предопределенных источников стимулирующих воздействий:

B0, B1, ... B15 - разряды двоичного счетчика,
N0, N1, ... N15 - инверсные разряды двоичного счетчика,
F0, F1, ... F15 - формульные стимуляторы,
C1, C2, C3 and C4 - тактовые стимуляторы,
Cs - заданная пользователем временная диаграмма.

Допустимо использовать шестнадцатиричное представление наименования стимуляторов.
Например BA, BB, … , BF instead вместо B10, B11, … , B15.

Sim

Формат: sim time
sim

Сокращение: none

Действие:
Эта макрокоманда начинает процесс моделирования. Время моделирования задается параметром time. Если значение этого параметра не определено, то будет промоделирован один шаг (step).

Примеры:

sim 53ns
sim

Эти команды задают моделирования в течение заданного отрезка времени: 53 нс и одного шага соответственно.

Stepsize

Формат: stepsize time

Действие:

Эта макрокоманда устанавливает величину шага моделирования, который определяет длительность всех остальных действий – кратными ему являются тактовые импульсы, время моделирования, временные воздействия и т.д.

Примеры:

stepsize 15ns
step 260ps

Длительность дискрета (шага) при моделировании установлена в 15 нс and 260 пс соответственно.

Unknown

Формат: x signal_or_bus_name

Действие:

Эта макрокоманда устанавливает сигнал или шину, заданную параметром signal_or_bus_name в неопределенное логическое состояние (Unkn_X).

Пример:

x U21.INPUT

На контакте U21.INPUT установлено неопределенное логическое состояние.

Vector

Формат: vector bus_name bus_element_1 ... bus_element_n
vector bus_name bus_element_range

Действие:
Эта макрокоманда предназначена для описания шины. Используя эту команду, можно создать краткую запись для набора сигналов. В шину могут быть включены как отдельные сигналы, так и фрагменты других шин.
Сформированные таким образом шины автоматически выводятся в поле сигналов.
Любой сигнал может быть включен в состав только одной шины.

Примеры:

vector Data_In D0 D1 D2 D3
v ADDRESS_BUS AD[0 : 16]

Watch

Формат: watch signal_name_1 signal_name_2 ... signal_name_n

Действие:
Этой макрокомандой задается, какие сигналы или шины будут видимы в окне просмотра временных диаграмм и сохраняться макрокомандой Print

Примеps:

watch Data_Input
w rst clr enable

Первая макрокоманда включает в перечень сигналов для просмотра сигнал Data_Input, вторая - сигналы rst, clr и enable.

3. ПОДГОТОВКА К РАБОТЕ

1. Изучить методические указания к данной работе.
2. Установить пакет в собственном персональном компьютере или освоить работу с пакетом в лаборатории кафедры.

4. РАБОЧЕЕ ЗАДАНИЕ

1. Создайте командный файл для моделирования схемы управления семисегментным индикатором из предыдущей работы. Шаг моделирования установите равным 10 ns, длительность моделирования – 5 мкс. Командный файл и результаты моделирования включите в отчет.
2. Опишите на VHDL счетчик из схемы управления семисегментным индикатором, откорректируйте имеющуюся схему, заменив макроэлемент со схемным описанием на элемент, реализованный на VHDL. Промоделируйте схему. Сравните результаты моделирования и убедитесь в том, что описанный на VHDL элемент работает аналогично элементу со схемной реализацией. Определите, какой вариант реализации (в целом) занимает больше ресурсов кристалла. Текст описания, результаты моделирования и количество использованных ресурсов для обоих вариантов реализации включите в отчет.
3. Опишите на VHDL заданный фрагмент, оформите его в виде макроэлемента, включенного в схему, подготовьте командный файл для моделирования и промоделируйте схему. Текст описания, командный файл и результаты моделирования включите в отчет.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Этапы проектирования элементов программируемой логики.

2. Понятия реального, модельного системного и автоматного времени.

3. Объясните разницу между режимами транспортной и инерциальной задержки.

4. В чем разница между сквозным и событийным моделированием?

5. Что такое VHDL, для чего он используется и в чем преимущества перед схемным проектированием ПЛИС?

6. В чем разница между поведенческим и структурным описанием на VHDL?

7. Этапы проектирования с использованием VHDL.

8. Что такое EDIF?

9. Что такое синтез VHDL-описания?

10. Что такое несинтезируемые конструкции VHDL? Приведите примеры.

11. Что такое FPGA?

12. Стили описания в VHDL.

13. Структура проекта в VHDL.

14. Что такое PACKAGE?

15. Что такое ENTITY?

16. Что такое ARCHITECTURE?

17. Типы данных в VHDL.

18. Атрибуты в VHDL.

19. Параллельно выполняемые операторы.

20. Последовательно выполняемые операторы.

21. Подробно объяснить разработанный на лабораторной работе фрагмент на VHDL.