621.382.8(07)
Р 851

№ 3412

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Таганрогский государственный радиотехнический университет

КАФЕДРА КОНСТРУИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

РУКОВОДСТВО К ЛАБОРАТОРНОЙ РАБОТЕ

МОДЕЛИРОВАНИЕ СЕНСОРНЫХ И АКТЮАТОРНЫХ ЭЛЕМЕНТОВ МИКРОСИСТЕМНОЙ ТЕХНИКИ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА VHDL-AMS

Для студентов специальностей 201900, 220500
и направлений 550700, 551100

ФЭП

Таганрог 2003

УДК 621.382.8(07)

Составители: И.Е. Лысенко, Е.А. Рындин

Руководство к лабораторной работе "Моделирование сенсорных и актюаторных элементов микросистемной техники с использованием языка VHDL-AMS". Таганрог: Изд-во ТРТУ, 2003. 26 с.

Цикл лабораторных работ по освоению студентами методов моделирования сенсорных и актюаторных элементов микросистемной техники с использованием языка VHDL-AMS.

В данной работе излагаются сведения, необходимые для моделирования элементов микросистемной техники с использованием языка VHDL-AMS. Описан маршрут моделирования сенсорных и актюаторных элементов МСТ в программе hAMSter.

Ил. 8. Библиогр.: 4 назв.

Рецензент В.Г. Ивченко, канд. техн. наук, доцент кафедры КЭС ТРТУ.

ВВЕДЕНИЕ

Микросистемная техника является одной из наиболее современных, динамично развивающихся и востребованных отраслей науки и техники. Элементы современных сверхбольших интегральных схем (СБИС) и микрооптико-электромеханических систем (МОЭМС) представляют собой сложные структуры, в основу функционирования которых положены разнообразные физические эффекты. Одним из важнейших этапов разработки элементов МОЭМС является математическое моделирование, основанное на решении систем алгебраических и дифференциальных уравнений.

На данный момент на рынке программного обеспечения существует большое количество пакетов программ численного моделирования интегральных элементов (в том числе элементов МОЭМС) и технологических процессов. Но огромное разнообразие эффектов, положенных в основу функционирования МОЭМС, как правило, требует создания программ моделирования, характеризующихся достаточно узкой специализацией. Существующие универсальные программные средства чаще всего представляют собой среду программирования с широким набором функций, включая функции автоматической генерации конечно-разностных и триангулярных координатных сеток, решения дифференциальных и алгебраических уравнений, графического отображения полученных результатов и др. Использование подобных универсальных средств позволяет весьма эффективно решать научно-исследовательские задачи, но не всегда удобно в процессе проектирования, когда необходимо получить решение с требуемой точностью за достаточно короткий промежуток времени.

Расширение стандарта VHDL (VHDL-AMS – Very High Speed Integrated Circuits Hardware Description Language Analog-Mixed Signals), предназначенное для описания и моделирования как цифровых, так и аналоговых приборов, причем использующих не только электрические сигналы, но также оптические, химические, механические и др., является важным шагом на пути создания универсальных САПР СБИС/МОЭМС, осуществляющих автоматическую компиляцию топологических решений на основе высокоуровневых текстовых описаний. Уже созданы и постоянно развиваются САПР, поддерживающие стандарт VHDL-AMS. Одной из таких систем является программа hAMSter ver.2.0 (High performance AMS Tool for Engineering and Research) компании Ansoft Corporation, предназначенная для работы на персональном компьютере [1].

В настоящем методическом пособии приводятся сведения о стандарте VHDL-AMS, подчеркнуты его основные отличительные особенности по сравнению с VHDL, описан маршрут моделирования сенсорных и актюаторных элементов микросистемной техники в программе hAMSter с использованием VHDL-AMS, предложена методика приобретения практических навыков моделирования элементов микрооптикоэлектромеханических систем в программе hAMSter.

1. ОСОБЕННОСТИ ЯЗЫКА VHDL-AMS

Основной язык VHDL-1076-1993 разработан организацией IEEE для описания и моделирования цифровых быстродействующих электронных систем.

Однако развитие микросистемной техники вызвало необходимость принятия IEEE нового стандарта – VHDL Std 1076.1-1999, который принято неформально называть VHDL-AMS (Very High Speed Integrated Circuits Hardware Description Language Analog-Mixed Signals) [2,3].

Этот стандарт основного языка и расширений предназначены для описания и моделирования схем и систем [2,3]:

• консервативных и неконсервативных;

• непрерывных и непрерывно-дискретных;

• цифровых, аналоговых и цифро-аналовых;

• электрических и неэлектрических;

• во временном, частотном и операторном представлениях с учетом характеристик шума.

Перечисленные возможности важны при моделировании многих микросистем, представляющих собой смешанно-сигнальные электронные или смешанные электрические/неэлектрические системы.

Особенности языка VHDL-AMS Std 1076.1-1999 [1,2,3]:

1) расширенная структурная семантика языка предполагает: консервативность в моделях физических систем (т.е. законы Кирхгофа для электрических цепей) и допускает не консервативность для абстрактных моделей; совмещение цифровых и аналоговых портов, т.е. смешанно-сигнальных интерфейсов;

2) в моделирующее ядро добавлены: новая имитационная модель, поддерживающая непрерывное поведение на основе дифференциально-алгебраических уравнений (ДАУ); подсистема моделирования аналоговых схем; предусмотрена оптимальность решения систем ДАУ аналоговым моделирующим ядром;

3) язык поддерживает моделирование малосигнального режима в частотном представлении, статистическое моделирование, построение шумовых моделей и их имитацию;

4) по сравнению со стандартом языка VHDL введены новые положения в поведенческое описание схем и систем; новые интерфейсные объекты и новые типы объектов; цикл имитации поддерживает смешанный режим, а среда моделирования дополнена новыми атрибутами.

2. ЭЛЕМЕНТЫ ЯЗЫКА VHDL-AMS

2.1. Зарезервированные слова

Зарезервированными словами называются идентификаторы, используемые для специального назначения. Отношение тех или иных идентификаторов к зарезервированным словам определяется стандартом языка [1, 2, 3].

Согласно стандарту Std 1076.1-1999 зарезервированными словами языка

VHDL-AMS являются все зарезервированные слова языка VHDL, а также дополнительно [2]: across, contribution, ground, nature, quantity, reference, terminal, through, tolerance.

Зарезервированное слово не должно использоваться в качестве меток, имен моделей и переменных различного типа, имен конструкций entity и architecture.

2.2. Арифметические операции

Язык VHDL-AMS Std 1076.1-1999 поддерживает следующие арифметические операции [1, 2]:

== – присваивание значений неизвестным;
:= – присваивание значений переменным и константам;
<= – меньше или равно, назначение сигнала;
>= – больше или равно;
+ – сложение переменных, констант и неизвестных;
- – вычитание переменных, констант и неизвестных;
* – умножение переменных, констант и неизвестных;
/ – деление переменных, констант и неизвестных;
** – возведение в степень переменных, констант и неизвестных;

real sin(real x), real cos(real x), real tan(real x)	– синус, косинус, тангенс действительного аргумента x;
real asin(real x), real acos(real x), real atan(real x)	– арксинус, арккосинус, арктангенс действительного аргумента x;
real sinh(real x), real cosh(real x), real tanh(real x)	– гиперболические синус, косинус, тангенс действительного аргумента x;
real log(real x), real log10(real x)	– натуральный и десятичный логарифмы действительного аргумента x;
real exp(real x)	– экспонента действительного аргумента x;
real sqrt(real x)	– квадратный корень действительного аргумента x;
real pow(real a, real x)	– возведение действительного аргумента x в степень a;
integer abs(integer x)	– модуль целого аргумента x.

2.3. Типы данных

VHDL-AMS поддерживает все типы данных, определенные в VHDL. Эти типы можно использовать в VHDL-AMS без каких-либо дополнительных ссылок [2,3]:

- real – действительные числа;
- integer – целые числа;
- natural – неотрицательные целые числа;
- positiv – положительные целые числа;

- bit – битовый тип {‘0’, ‘1’};
- boolean – логический тип данных {false, true};
- vector – массив типа real;
- bit_vector – массив типа bit;
- time – физический тип “время”.

Кроме того, для моделирования аналоговых систем в VHDL-AMS предусмотрены дополнительные типы данных, описывающих различные физические параметры (domain) [3]: electrical_systems; mechanical_systems; fluidic_systems; radiant_systems; thermal_systems; chemical_systems.

Дополнительные типы данных располагаются в отдельных пакетах.

2.4. Классы данных

В языке VHDL-AMS предопределены следующие классы данных [2,3].

1. **Constant** – константы. Значение константы определяется при ее объявлении и не может быть изменено. Константы могут иметь любой из поддерживаемых типов данных, включая пользовательские.

2. **Variable** – переменные. Значение переменной изменяется столько раз, сколько встречается присваивание данной переменной. Переменные могут иметь любой из поддерживаемых типов данных.

3. **Signal** – сигналы. Сигналы представляют значения, передаваемые по проводам и определяемые присвоением сигналов (отличным от присвоения переменных).

4. **Quantity** – неизвестные. Значения неизвестным присваиваются в результате решения дифференциальных и алгебраических уравнений. Система моделирования аналоговой части схемы выполняет предварительно для всех зависящих от времени неизвестных в различных уравнениях дифференциальной части ДАУ начальное преобразование их в специфические величины времени, используя подходящие методы дискретизации.

Quantity могут иметь только скалярный тип с плавающей точкой.

Рассмотрим характеристики quantity. Характеристики составных quantity являются просто совокупностью характеристик своих скалярных субэлементов. Поведение каждого скалярного субэлемента не зависит от других.

Quantity могут быть объявлены там же, где может объявляться сигнал, за исключением пакета.

Quantity могут также быть объявлены как элементы интерфейса в списке портов. Интерфейсным элементом quantity называется порт quantity, аналогичный сигнальным портам. Интерфейсные quantity имеют режим, подобный режиму сигнала интерфейса (IN, OUT, INOUT):

```
ENTITY example IS
PORT (  QUANTITY inl, in2: IN real;
        QUANTITY out1: OUT real);
END example;
```

Дополнительно предопределены следующие quantity:

- Q'dot – производная quantity Q по времени;
- Q'integ – интеграл от quantity Q по времени от нуля до текущего момента;
- Q'delayed(t) – значение quantity Q в прошедший момент времени, отстоящий от настоящего на фиксированный интервал (идеальная задержка $t \geq 0$);
- ANOW – представление текущего времени моделирования;
- Q'slew (max_rising_slope, max_falling_slope) –quantity Q, производная по времени которого (крутизна фронтов) ограничена сверху указанными значениями;
- Q'ltf(num,den) –функция интегрального преобразования по Лапласу от скалярного quantity Q как функции времени. Здесь num – константа типа REAL_VECTOR, являющаяся коэффициентом полинома числителя; den – константа типа REAL_VECTOR, являющаяся коэффициентом полинома знаменателя. Функция преобразования по Лапласу ставит в соответствие функции времени Q функцию частоты в виде отношения полиномов по степеням операторной переменной s. Во временной области это соответствует ДАУ с постоянными коэффициентами. В расширении языка функция преобразования по Лапласу определена как атрибут quantity**,** подобный атрибутам Q'Dot или Q'lnteg;
- Q'zoh (T,initial_delay) –дискретный эквивалент quantity Q. Тип результата совпадает с типом Q. T – положительное значение вещественного типа, задающее период дискретизации; initial_delay – неотрицательное значение вещественного типа, задающее начальный момент дискретизации в секундах (по умолчанию устанавливается значение 0.0);
- S'ramp (tr,tf) – quantity, повторяющее сигнал S, но с заданными длительностями переднего и заднего фронтов;
- S'slew (max_rising_slope, max_falling_slope) – quantity, повторяющее сигнал S, но с производными по времени (крутизной фронтов), ограниченными сверху указанными значениями;
- Q'ztf (num, den, T, initial_delay) –функция интегрального Z-преобразования quantity Q как функции времени. Здесь num – константа типа REAL_VECTOR, являющаяся коэффициентом полинома числителя; den – константа типа REAL_VECTOR, являющаяся коэффициентом полинома знаменателя; T – положительная константа типа real, задающая период дискретизации; initial_delay– неотрицательная константа типа real, определяющая задержку начала дискретизации (по умолчанию устанавливается 0.0). Функция Z-преобразования ставит в соответствие функции времени Q с задержкой на константу T отношение двух полиномов от переменной 1/z. Подобно функции преобразования по Лапласу, функция Z-преобразования определена как атрибут quantity. Первый скалярный элемент массива знаменателя должен быть ненулевым. Тип числителя и знаменателя определен в пакете STANDARD как неограниченный массив с натуральным индексом.

5. **Terminal** (терминал) описывает переменные составных типов **nature** со скалярными элементами, для которых должны выполняться законы консерва-

тивности. Branch_quantity (неизвестная ветви) объявляется между двумя терминалами.

Каждый простой тип nature представляется как определенные физические параметры – electrical_systems, mechanical_systems и т.д.

Существует два типа branch_quantity:

- **across quantity** – воздействие, подобное, например, напряжению, температуре или давлению;
- **through quantity** – потоковое воздействие, подобное, например, току, потоку тепла или потоку жидкости.

Простой тип nature определяет свойство терминала быть представленным парой скалярных типов с плавающей точкой (рис. 1). Например [1]:

```
SUBTYPE voltage IS real;
SUBTYPE current IS real;
NATURE electrical IS voltage ACROSS; current THROUGH;
TERMINAL plus, minus: electrical;
QUANTITY v ACROSS i1, i2 THROUGH plus TO minus;
```

(v – across-величина, представляет разность напряжений на терминалах tl и t2: v = vt1 - vt2; i1, i2 – through-величины, представляют токи в двух параллельных ветвях, текущие от plus к minus).

Приведенное описание объявляет:

- два подтипа – подтип voltage (напряжение) и подтип current (ток);
- простой тип nature electrical;
- два терминала типа nature electrical и два quantity across и through между двумя терминалами.

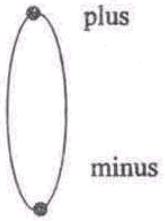

Рис.1. Иллюстрация к примеру объявления типа nature electrical

Across quantity представляют разность потенциалов между двумя терминалами, а quantity through описывают различные параллельные ветви с током (токи текут от терминала plus к терминалу minus).

Тип branch_quantity явно не объявлен. Чаще всего он отражает nature ее терминалов. Он может быть составным типом. В примере quantity across v – тип напряжения, а тип quantity through il и i2 – ток. Как и в случае скалярного quantity, характеристики составного являются поименованным агрегатом характеристик его скалярных элементов. Терминалы должны иметь элементы того же простого типа nature и должны быть согласованы в других специфических особенностях. Терминалы quantity характеризуют терминал "плюс" и

терминал "минус" и направление ветви от плюса к минусу (в электрической системе это направление положительного течения тока).

Терминал может быть объявлен там же, где допущено объявление сигналов. В конкретном случае терминал может быть элементом интерфейса в списке объявления портов. Тогда он называется терминальным портом:

PORT (TERMINAL anode, cathode: electrical);

Связывание терминальных портов используется для создания узлов в иерархических описаниях.

Объявление простого типа nature создает так называемый опорный терминал, который распространяется на все терминалы с элементами этой простой nature.

Опорный терминал T природы N определен как N'Reference. Само объявление T создает два quantity (рис. 2) [1]:

- эталонная величина T'Reference является quantity across между плюсовым терминалом T природы N и минусовым терминалом N'Reference (т.е. потенциалом относительно "земли");
- суммарное quantity T'Contribution является quantity through, величина которой равняется сумме всех through quantity, приложенных к T (с соответствующим знаком). Если терминал T появляется как фактический, то суммарные branch_quantity соответствующей природы добавляются к сумме.

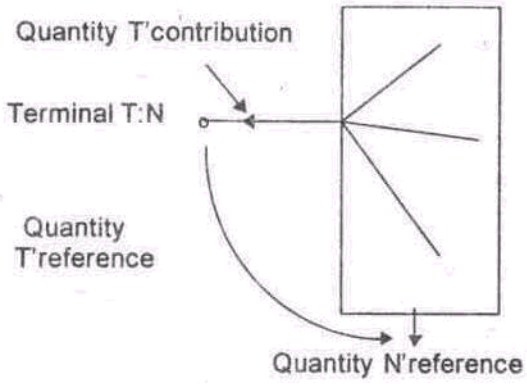

Рис.2. Опорный терминал

Как T'Reference, так и T'Contribution являются составными, если T составной. В этом случае правила относятся к каждому скалярному элементу T.

Величина каждого скаляра across quantity ограничена разницей опорных (referance) quantity своих терминалов. Узел является комплектом скалярных терминалов, созданных деревом терминальных соединений. Все опорные quantity терминалов узла ограничены и равны между собой, а суммарное quantity терминала в корне дерева равно нулю.

Из приведенного выше следует, что переменные класса quantity могут быть как “свободными” переменными, т.е. не связанными непосредственно с терминалами (портами и узлами) схемы, так и переменными состояния схемы (branch_quantity), т.е. в электрическом контексте – токами и напряжениями ветвей схемы. Для branch_quantity должны выполняться законы консервативности

схемы (законы Кирхгофа).

2.5. Модули модели

Модель в VHDL-AMS представлена совокупностью иерархически связанных текстовых фрагментов [3].

В начале каждой модели в VHDL-AMS должно находится указание на библиотеки и пакеты, информация из которых используется для построения данной модели.

Пакеты (package) служат для объявления типов, констант и функций, которые используются в одном или нескольких моделях. В VHDL-AMS не внесено существенных изменений в представление пакетов, поэтому рассмотрим лишь их синтаксис [4]:

```
PACKAGE <имя_пакета>  IS
        <объявление констант, типов, сигналов, интерфейсов функций>;
END <имя_пакета>;

PACKAGE BODY <имя_пакета> IS
        <тела функций>;
END <имя_пакета>;
```

Пример пакета electrical_system [2]:

```
PACKAGE electrical_system IS
        TYPE current IS RANGE -1.0e+6 TO 1.0e+6;
           UNITS
              A;
              mA = 1.0e-3 A;
              uA = 1.0e-3 mA;
              kA = 1.0e+3 A;
           END UNITS;
        TYPE voltage IS RANGE -1.0e+9 TO 1.0e+9;
           UNITS
              V;
              mV = 1.0e-3 V;
              uV = 1.0e-3 mV;
              kV = 1.0e+3 V;
           END UNITS;
        NATURE electrical IS voltage ACROSS current THROUGH;
END electrical_system;
```

Чтобы использовать при описании модели объекты, декларированные в некотором пакете, достаточно включить в файл проекта ссылку на библиотеку, в которой находится нужный пакет [4]:

LIBRARY <имя_библиотеки>;
USE <имя_библиотеки>.<имя_пакета>.<имя_модуля>;

Последняя запись определяет место хранения используемой информации. Вариант записи, когда в качестве имени модуля используется зарезервированное слово ALL, обеспечивает доступ ко всем модулям объявленного пакета [4]:

USE <имя_библиотеки>.<имя_пакета>.ALL;

Пример подключения библиотеки модели:

LIBRARY IEEE;
USE ieee.electrical_systems.ALL;

Для описания общих элементов модели устройства, прежде всего ее внешнего интерфейса – портов устройства, используется конструкция entity. В entity описываются также параметры модели, общие для всех архитектурных тел, связанных с данной entity. Описание entity состоит из port_clause (декларация портов) и generic_clause (декларация глобальных констант) [2,3].

Синтаксис описания entity [4]:

ENTITY <имя_модели> IS
    GENERIC (<декларация_глобальных_констант_модели>);
    PORT (<декларация_портов_модели>);
END <имя_модели>;

В AMS расширении языка VHDL включены новые классы портов: порты классов terminal и quantity. Через порты класса terminal модель может принимать и передавать значения переменных базового типа natural, в частности, электрических переменных (токов и напряжений). Через порты класса quantity модель может принимать или передавать значения непрерывных переменных числовых типов (в отличие от сигналов, которые имеют дискретные значения) [2,3].

Синтаксис портов описания entity имеет следующий вид [3]:

SIGNAL <имя> : <режим> <тип> (:= <значение_по_умолчанию>);
TERMINAL <имя> : <тип>;
QUANTITY <имя> : < режим > <тип>;

Для описания режима используются зарезервированные слова IN (входной порт), OUT (выходной порт), INOUT (двунаправленный порт).

Пример описания entity:

ENTITY resistor IS
    GENERIC (CONSTANT r : resistance);

```
        PORT (TERMINAL node1, node2 : electrical);
    END resistor;
```

Однако конструкция entity может не содержать декларации глобальных констант и портов модели. В этом случае конструкция entity объявляет только имя модели.

Функционирование устройства описывается в архитектурном теле (architecture_body).

Конструкция architecture_body предназначена для описания “поведения” модели, т.е. является основной содержательной частью описания модели. Она состоит из заголовка, разделов деклараций и операторов [2,3,4].

В разделе деклараций вводятся описания типов данных, переменных, констант и локальных функций. А соотношения между переменными записываются в разделе операторов [2,3,4].

Синтаксис описания architecture_body имеет следующий вид [4]:

```
ARCHITECTURE <имя_архитектуры> OF <имя_модели> IS
        <раздел_деклараций>;
BEGIN
        <раздел_операторов>;
END <имя_архитектуры>;
```

Основное отличие VHDL-AMS от VHDL – это наличие в разделе деклараций объявлений терминалов (terminal_declaration) и неизвестных (quantity_declaration) [2,3].

Terminal_declaration объявляет переменные составных типов natural. Эти декларации позволяют ввести переменные класса branch_quantity, связанные с переменной класса teminal.

Применительно к электрическим цепям terminal_declaration – это декларация узла электрической схемы, а branch_quantity – это потенциал и/или токи ветвей, соединенных с объявленными терминалами.

Quantity_declaration объявляет переменные класса quantity, т.е. непрерывные во времени переменные, являющиеся неизвестными системы алгебраических и дифференциальных уравнений, описываемых в разделе операторов.

Пример описания architecture_body:

```
ARCHITECTURE behavioral OF resistor IS
        QUANTITY v ACROSS i THROUGH node1 TO node2;
BEGIN
        v == i*r;
END behavioral;
```

Одному entity может соответствовать несколько architecture_body, отличающихся степенью детализации описания модели [3,4].

2.6. Операторы

В разделе операторов архитектурного тела могут присутствовать два типа параллельных операторов: **simultaneous_statement** и **concurrent_statement** [3].

Операторы simultaneous_statement появились лишь в последней версии языка и предназначены специально для описания аналоговой части модели.

Совокупность всех simultaneous_statement – это система алгебраических и дифференциальных уравнений, возможно с управляющими операторами. Неизвестными этой системы являются переменные класса quantity. Система должна быть совместной: число уравнений, решаемых в каждый момент времени моделирования, должно равняться числу неизвестных переменных класса quantity. Кроме уравнений, явно записываемых в разделе операторов архитектурного тела, существуют неявные уравнения, поддерживающие законы консервативности системы. Эти уравнения формируются автоматически при трансляции VHDL-AMS описания для переменных классов terminal.

В разделе операторов описания architecture_body допускается использовать следующие типы simultaneous_statement [3].

1. Операторы simple_simultaneous_statement – это простые уравнения, в правой и левой части которых записываются выражения. Операндами выражений могут быть переменные классов quantity или signal, константы, декларированные в архитектурном теле, или в области видимости архитектурного тела, а также функции и атрибуты переменных. Все операнды выражений, входящих в simple_simultaneous_statement, должны иметь базовый тип real или integer. Операторами в этих выражениях могут быть обычные арифметические операторы: +, -, *, /, **. Для изменения порядка выполнения действий в выражениях можно применять круглые скобки.

Пример записи simple_simultaneous_statement:

```
a'DOT == 2.0*(b + x**3.0);
```

2. Операторы simultaneous_if_statement и simultaneous_case_statement применяются для модификации системы уравнений в зависимости от значений управляющих переменных в условиях. В качестве управляющих переменных могут использоваться как переменные класса quantity, так и signal. Это открывает возможность влияния цифровой части модели на ее аналоговую часть и наоборот.

Синтаксис simultaneous_if_statement:

```
IF <условие_1> USE
    <раздел_операторов_1>;
ELSIF <условие_2> USE
    <раздел_операторов_2>;
…
ELSE
    <раздел_операторов_n>;
END USE;
```

Пример simultaneous_if_statement:

```
IF sig = '1' USE
    var == a + b;
ELSE
    var == a + b**2.0;
END USE;
```

В примере simultaneous_if_statement переменная sig – сигнал типа bit. В зависимости от ее значения аналоговая переменная var класса quantity вычисляется по различным формулам.

Синтаксис simultaneous_case_statement:

```
CASE <выражение> USE
    WHEN <значение_1> USE
        <выражение_1>;
    WHEN <значение_2> USE
        <выражение_2>;
    …
END CASE;
```

Пример simultaneous_case_statement:

```
CASE sig USE
    WHEN 0 USE
        var == a + b;
    WHEN 1 USE
        var == a + b**2.0;
    WHEN 2 USE
        var == a + b**3.0;
    WHEN OTHER USE
        var == a;
END CASE;
```

В примере simultaneous_case_statement sig – сигнал типа integer. В зависимости от значения сигнала sig аналоговая переменная var класса quantity вычисляется по различным формулам.

Разумеется, что <условие> оператора simultaneous_if_statement и <выражение> оператора simultaneous_case_statement могут зависеть не только от сигналов, но и от переменных класса quantity.

Concurrent_statement предназначены для описания цифровой части модели. Они служат для формирования значений дискретных сигналов и для формирования иерархических описаний. Также операторы concurrent_statement являются средством взаимодействия аналоговой и цифровой частей модели [3].

К concurrent_statement относятся следующие операторы [3].

1. Оператор процесса (process) определяет независимый последовательный процесс, представляющий поведение некоторой части модели цифро-аналогового устройства. Выполнение оператора процесса состоит из повторяющегося выполнения последовательности операторов, записанных в нем.

В VHDL-AMS не внесено существенных изменений в представление оператора process, поэтому рассмотрим лишь его синтаксис [3]:

```
(<метка_процесса>)
PROCESS (<список_чувствительности>)
    <раздел_деклараций>;
BEGIN
    <раздел_операторов>;
END PROCESS (<метка_процесса>);
```

2. Оператор selected_signal_assignment является одним из операторов, в котором аналоговые переменные оказывают влияние на цифровые сигналы.

Синтаксис оператора selected_signal_assignment [3]:

<имя_сигнала> <= <назначение_1> WHEN <условие_2> ELSE
<назначение_2> WHEN <условие_2> ELSE
…
<назначение_(n-1)> WHEN <условие_n> ELSE <назначение_n>;

Основным способом изменения значения цифрового сигнала в зависимости от значения аналоговой переменной является использование атрибута ‘ABOVE. Значение атрибута q’ABOVE(v) переменной q класса quantity есть неявный сигнал типа boolean, который принимает значение TRUE, если q > v, и FALSE, если q < v, где v – постоянное значение.

Пример использования оператора selected_signal_assignment:

```
s <=  9 WHEN as'ABOVE(4) ELSE
      8 WHEN as'ABOVE(3) ELSE
      …
      1 WHEN as'ABOVE(-4) ELSE 0;
```

преобразует непрерывную переменную as в целочисленный дискретный сигнал s.

3. Оператор concurrent_break_statement также является средством взаимодействия аналоговой и цифровой частей модели. Записывается он по следующим правилам [3]:

BREAK <назначение_1>; …; <назначение_n> <условие>;

Примеры использования оператора concurrent_break_statement:

1) BREAK a => 0 WHEN s == ‘1’;

аналоговой переменной a присваивается значение 0 только тогда, когда значение дискретного сигнала s становится равным ‘1’;

2) BREAK a => 0; b <= c on x;

в моменты, когда сигнал x изменяет свое значение, переменной a присваивается значение 0, а переменной b – значение c.

3) BREAK ON q1,q2,q3;

в моменты времени, когда любой из сигналов Q1,Q2,Q3 изменяет свое значение, подсистема моделирования получает предупреждение о возможном нарушении непрерывности в системе уравнений.

4) BREAK v => -v, s => 0 WHEN NOT s'ABOVE(0.0);

в моменты времени, когда аналоговая переменная s становится не положительной, ей присваивается значение 0, а переменная v меняет знак.

2.7. Стили описания моделей в языке VHDL-AMS

1. Структурное моделирование (structural modeling) непрерывных систем производится с помощью переменных составных типов nature, причем каждый простой тип nature представляется как определенные физические параметры – electrical_systems, mechanical_systems и т.д. При структурном описании модель представляется в виде иерархии связанных компонентов, в которой каждый

компонент представляет собой определенный физический параметр [3].

2. Поведенческое моделирование (behavioral modeling) непрерывных систем производится с помощью наборов ДАУ, описывающих взаимосвязь различных физических параметров. Поведенческое описание модели позволяет разработчику избежать операции декомпозиции модели по физическим параметрам и сосредоточить основное внимание на одной физической величине [3].

3. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ hAMSter

Программа hAMSter ver.2.0 (**H**igh performance **AMS T**ool for **E**ngineering and **R**esearch) компании Ansoft Corporation является первой программой моделирования VHDL-AMS описаний, предназначенной для работы на персональном компьютере [1].

Программа hAMSter ver.2.0 содержит текстовый (Model Editor) и графический (hAMSterView) редакторы [1].

Текстовый редактор Model Editor предназначен для ввода VHDL-AMS модели исследуемого устройства, сохранения его описания в формате *.vhd, компиляции и моделирования. Задание на моделирование VHDL-AMS описания элемента микросистем располагается в файле *.cfg [1].

Графический редактор hAMSterView предназначен для представления результатов моделирования в графическом виде. Данный редактор позволяет производить различные преобразования полученных результатов: изменение цвета, толщины и маркировки графиков, шкалы представления. Результаты изменений сохраняются в файле с расширением *.vtc. В дальнейшем при моделировании той же VHDL-AMS модели элемента в редакторе hAMSterView полученные графики будут выводиться согласно настройкам [1].

Программа hAMSter ver.2.0 не поддерживает [1]:

- зарезервированные слова alias, array, generate, file, limit, record, tolerance;
- смешанные типы nature;
- предопределены переменные класса quantity: Q'tolerance, Q'ltf, Q'ztf;
- атрибуты сигналов: S'delayed, S'stable, S'quiet, S'transaction, S'last_event, S'driving, S'driving_value.

4. ПРИМЕР МОДЕЛИРОВАНИЯ ЭЛЕМЕНТА МСТ В hAMSter

В качестве примера рассмотрим маршрут моделирования механического микрозеркала с помощью программы hAMSter.

VHDL-AMS модель микрозеркала имеет следующий вид [1]:

```
LIBRARY IEEE;
USE IEEE.MATH_REAL.ALL;

ENTITY mirror_new IS
END ENTITY mirror_new;
```

```
ARCHITECTURE behav OF mirror_new IS
        QUANTITY alpha, M1, M2, U1, U2 : REAL;
        SIGNAL u1_signal : REAL := 0.0;
        SIGNAL u2_signal : REAL := 0.0;
        CONSTANT Eps : REAL := 8.85419e-12;
        CONSTANT d : REAL := 370.0e-6;
        CONSTANT La : REAL := 3.02121e-3;
        CONSTANT br : REAL := 3.02121e-3;
        CONSTANT C : REAL := 2.5e-5;
        CONSTANT K : REAL := 1.0e-10;
        CONSTANT E : REAL := 4.851e-13;
BEGIN
--electric part
--triangular
   u1_signal  <= 1000.0 AFTER 0 ms ,    0.0 AFTER    50 ms;
   U1 ==  u1_signal'RAMP(50.0e-3,50.0e-3);

   u2_signal  <= -1000.0 AFTER  100 ms, 0.0 AFTER  150 ms;
   U2 ==  u2_signal'RAMP(50.0e-3,50.0e-3);

--mechanical part
   IF (abs(alpha) < 1.0e-5) USE
      M1 == -(Eps * La / 16.0) * (U1 * br / d)**2;
      M2 == -(Eps * La / 16.0) * (U2 * br / d)**2;
   ELSE
      M1 == -(Eps * La * U1 ** 2) / (2.0 * tan(alpha) ** 2 ) * (br * tan(alpha)/(2.0 * d - br *
tan (alpha)) + log ((2.0 * d - br * tan(alpha)) / (2.0 * d)));
      M2 == -(Eps * La * U2 ** 2) / (2.0 * tan(alpha) ** 2 ) * (br * tan(alpha)/(2.0 * d - br *
tan (alpha)) + log ((2.0 * d - br * tan(alpha)) / (2.0 * d)));
   END USE;

   (alpha'dot'dot) == ((M1 - M2) - K * alpha'dot - C *alpha) * 1.0/E;

END ARCHITECTURE behav;
```

Основные этапы моделирования:

1) запуск программ hAMSter;
2) ввод текста модели элемента МСТ;
3) компиляция модели элемента МСТ;
4) моделирование элемента МСТ;
5) просмотр результатов моделирования;
6) сохранение результатов моделирования.

Для запуска программы hAMSter необходимо запустить файл hamster.exe в каталоге, указанном преподавателем.

После запуска программы на экране появляется окно текстового редактора Model Editor (рис. 3). В рабочем окне текстового редактора необходимо ввести приведенную выше VHDL-AMS модель микрозеркала.

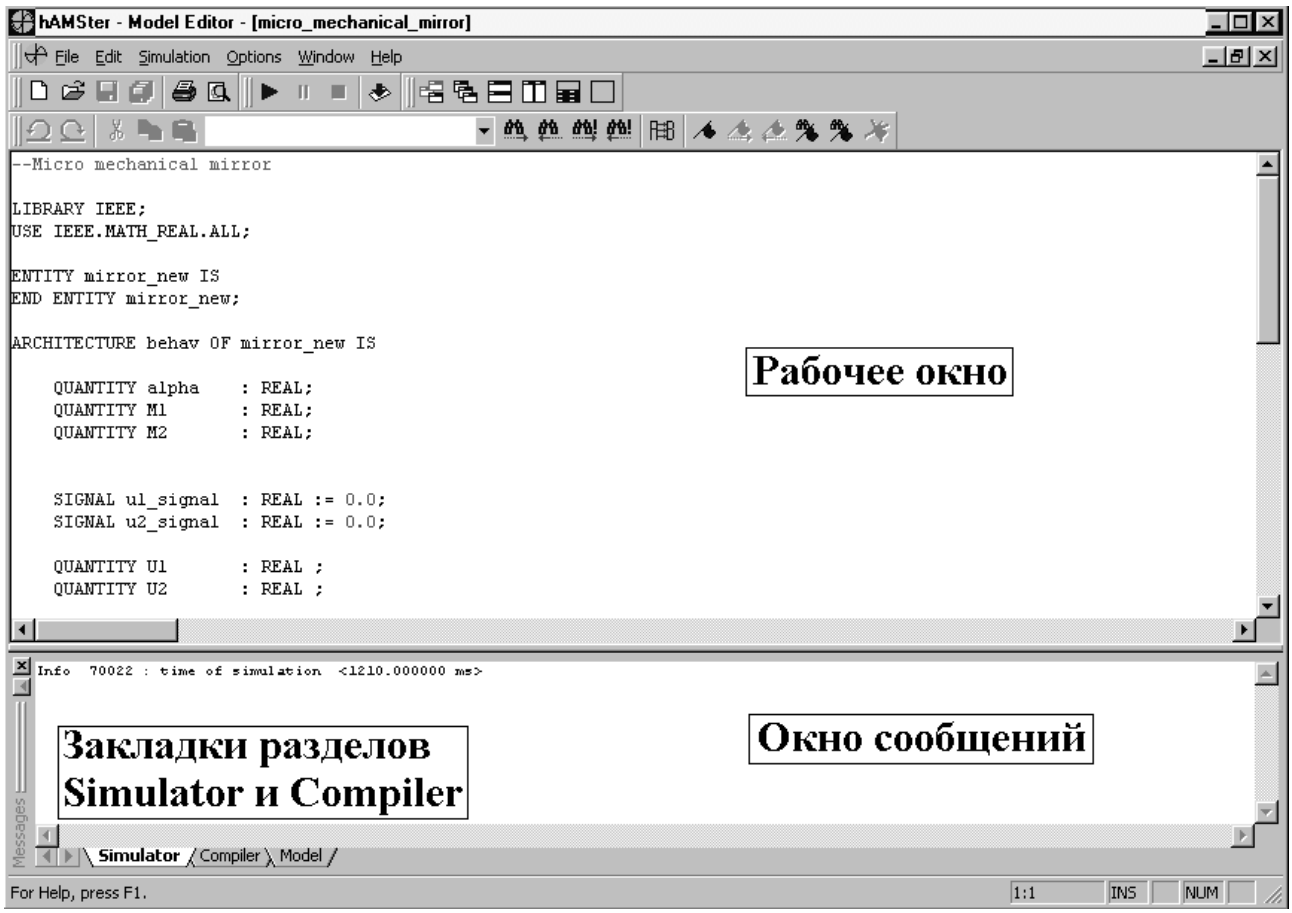

Рис.3. Рабочее окно текстового редактора Model Editor

Для сохранения введенной модели микрозеркала необходимо щелкнуть левой кнопкой манипулятора «мышь» на пункте меню File\Save или нажать комбинацию клавиш Ctrl+S. В появившемся окне необходимо указать диск и каталог для размещения файлов модели, а также ввести имя файла модели. Имя файла модели необязательно должно совпадать с именем модели, указанным в модуле entity.

Для проверки корректности ввода текста VHDL-AMS модели элемента и компиляции необходимо щелкнуть левой кнопкой «мыши» на пункте меню Simulation\Compile.

При этом в окне сообщений текстового редактора Model Editor необходимо открыть раздел Compiler (см. рис. 3), щелкнув левой кнопкой «мыши» на соответствующей закладке.

В разделе Compiler будет содержаться перечень ошибок и замечаний по тексту модели. При нажатии левой кнопкой «мыши» на строке ошибки в разделе Compiler окна сообщений курсор в рабочем окне редактора перейдет на строку, содержащую эту ошибку. Пример сообщения об ошибке:

Error : Type mismatch *(в выражении используются переменные разного типа)*

Error : Compile/link failed because of 1 errors <дата и время компиляции>

Error 70015 : error with compiling

В случае успешной компиляции VHDL-AMS модели элемента в окне со-

общений текстового редактора должно быть следующее сообщение:

Info : Entity/Package "mirror_new" overwritten

Info : Compile OK <дата и время компиляции>

Далее необходимо выполнить моделирование элемента.

Для моделирования микрозеркала в текстовом редакторе программы hAMSter необходимо нажать левой кнопкой «мыши» на пункте меню Simulation\Start или нажать клавишу F12.

На экране компьютера появится окно Simulation parameters (рис. 4).

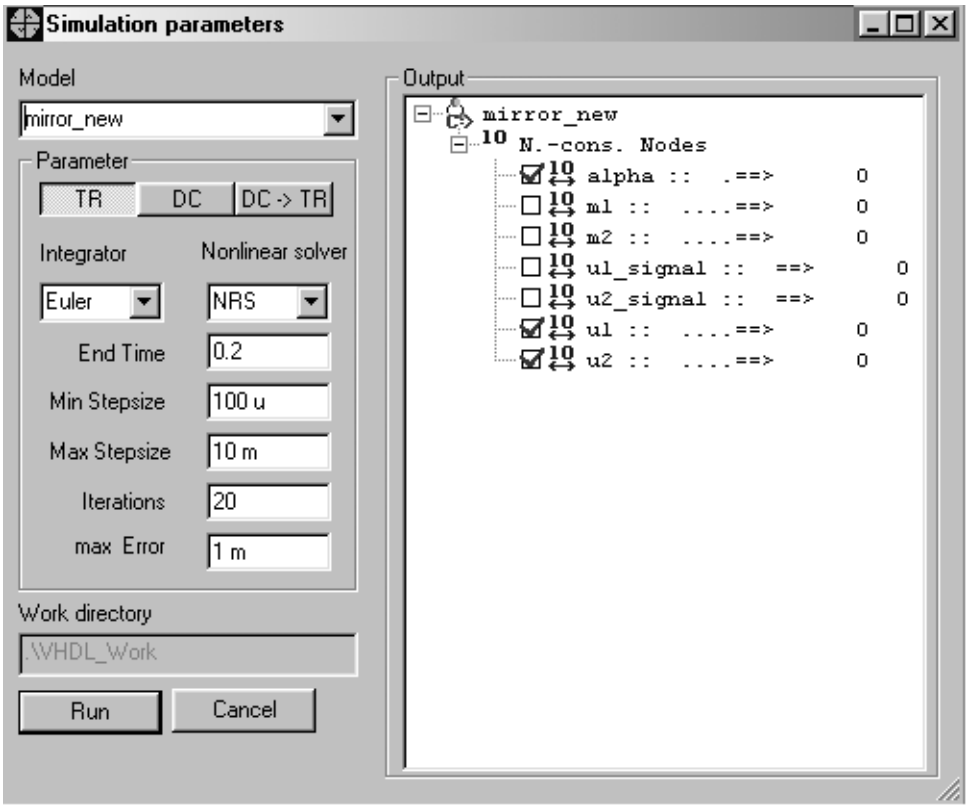

Рис.4. Окно Simulation parameters

В разделе Model окна Simulation parameters указывается имя модели.

В разделе Parameter окна Simulation parameters необходимо указать:

1. Вид моделирования:
   - TR (Transient analysis) – моделирование переходных процессов;
   - DC – расчет статического режима;
   - DC -> TR – последовательное выполнение расчета начальных условий и моделирования переходных процессов.
2. Метод интегрирования (Integrator):
   - Эйлера (Euler);
   - трапеций (Trapez).
3. Итерационный метод решения нелинейных систем для расчета статического режима (Nonlinear solver):
   - модифицированный метод Ньютона-Рафсона (NRS);
   - метод Ньютона-Рафсона (NR);

- метод Левенберга-Марквадта (MQR).

4. Для моделирования переходных процессов:
   - время моделирования (End time);
   - минимальный шаг моделирования (Min Stepsize);
   - максимальный шаг моделирования (Max Stepsize);
   - количество итераций (Iterations);
   - максимальное возможное количество ошибок при моделировании (Max Error).

В разделе Output окна Simulation parameters необходимо указать результаты моделирования каких переменных модели необходимо вывести в окне графического редактора hAMSterView. Для этого необходимо левой кнопкой «мыши» щелкнуть на квадратике, расположенном слева от названия переменной. После выполнения этого действия в квадратике должна появится галочка. Щелчок левой кнопкой «мыши» на квадратике с галочкой приводит к снятию задания на вывод результатов моделирования этой переменной в окне графического редактора hAMSterView.

После выполнения всех назначений в окне Simulation parameters необходимо левой кнопкой «мыши» нажать на клавише Run, расположенной в левом нижнем углу окна Simulation parameters (см. рис. 4).

При этом в окне сообщений текстового редактора Model Editor в разделе Simulator (см. рис. 3) будет указана причина ошибки, возникшей при моделировании элемента. Пример сообщения об ошибке:

Error 70116 : not enough equations -> under-determined model
(не выполнено условие консервативности ДАУ)
Error 70024 : simulation broken off

После исправления указанных ошибок в VHDL-AMS модели необходимо вновь повторить операцию компиляции и запустить процесс моделирования элемента.

В случае успешного моделирования элемента в разделе Simulator текстового редактора появится следующая запись:

Info 70022 : time of simulation <время моделирования>

Одновременно на экране компьютера появится рабочее окно графического редактора hAMSterView программы hAMSter с результатами моделирования элемента (рис. 5).

Окно графического редактора hAMSterView содержит две области (рис. 5):

- дискретных переменных (signal);
- аналоговых переменных (переменные типа constant, variable, quantity).

Для представления графиков во всем диапазоне времени моделирования необходимо левой кнопкой «мыши» щелкнуть на иконке  , расположенной на панели инструментов. На экране компьютера должно появиться меню Select Diagram (рис. 6).

В появившемся меню можно выбрать либо одиночные графики, либо графики по типу переменных (Digital/Analog diagram), либо все графики (All diagrams).

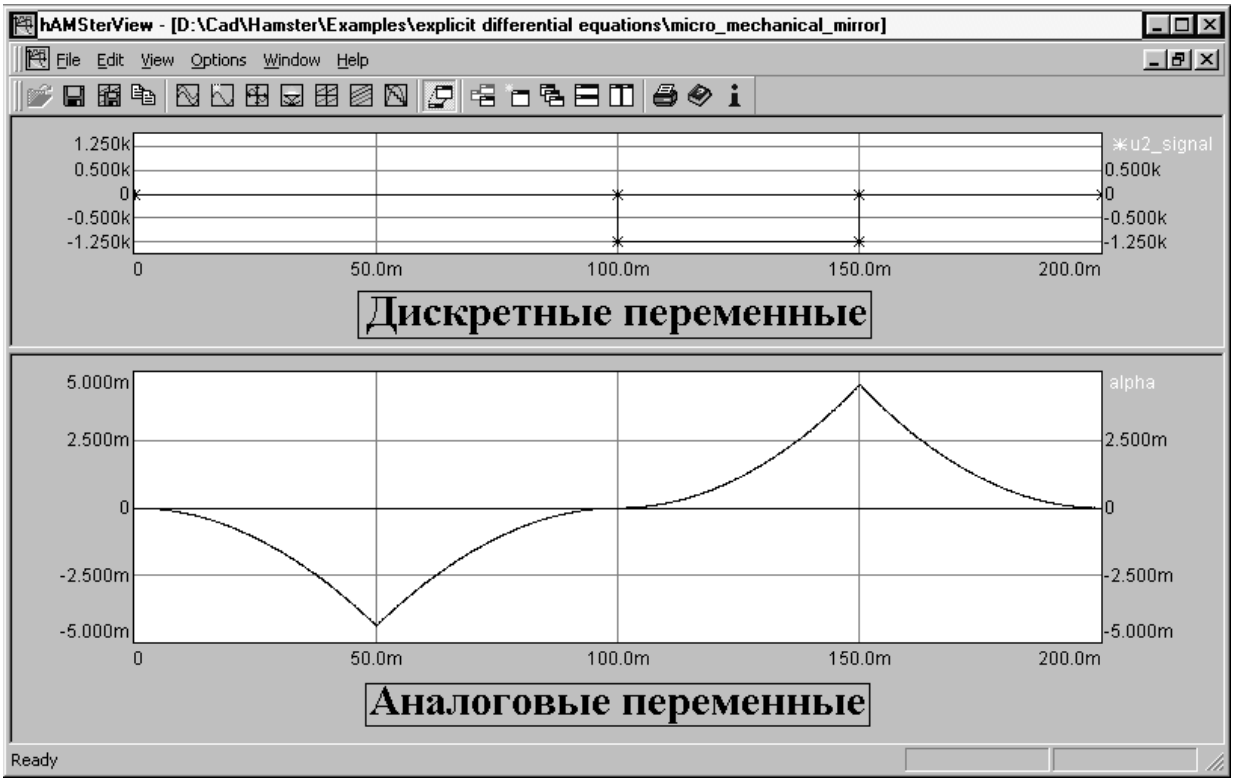

Рис.5. Рабочее окно графического редактора hAMSterView

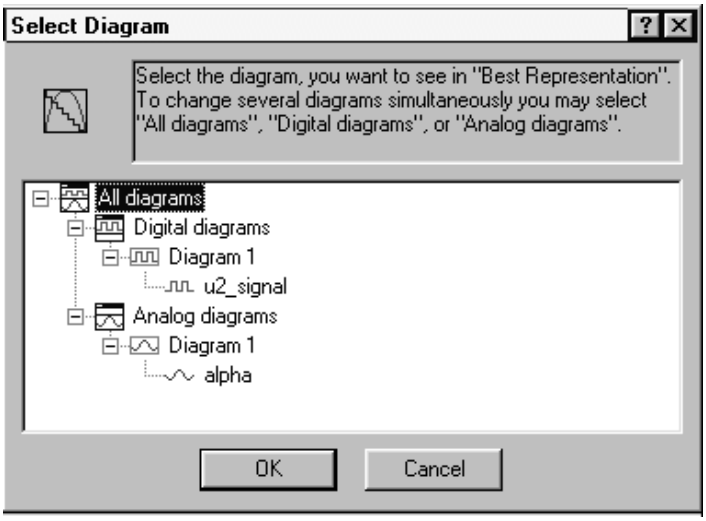

Рис.6. Меню Select Diagram

Для изменения атрибутов графиков необходимо правой кнопкой «мыши» щелкнуть на названии графика, расположенном в его верхнем правом углу (рис. 7).

Выбрав в появившемся меню атрибутов графика соответствующие пункты можно изменять цвет (Color), толщину и маркировку линий (Representation), шкалу представления (Scale).

В случае представления зависимостей переменных модели от времени на одном графике, выбрав пункт Separate Diagram меню атрибутов, можно разделить их на разные графики.

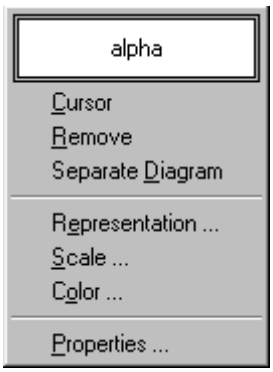

Рис.7. Меню атрибутов графика

Для более детального анализа результатов моделирования с помощью пункта Cursor меню атрибутов можно вызвать курсор, показывающий значение переменной в выбранный момент времени.

Для сохранения конфигурации графиков необходимо нажать левой кнопкой «мыши» на пункте меню File\Save Configuration.

Для сохранения результатов моделирования необходимо нажать левой кнопкой «мыши» на пункте меню File\Save Data. При этом на экране компьютера появится меню Save Selective (рис. 8).

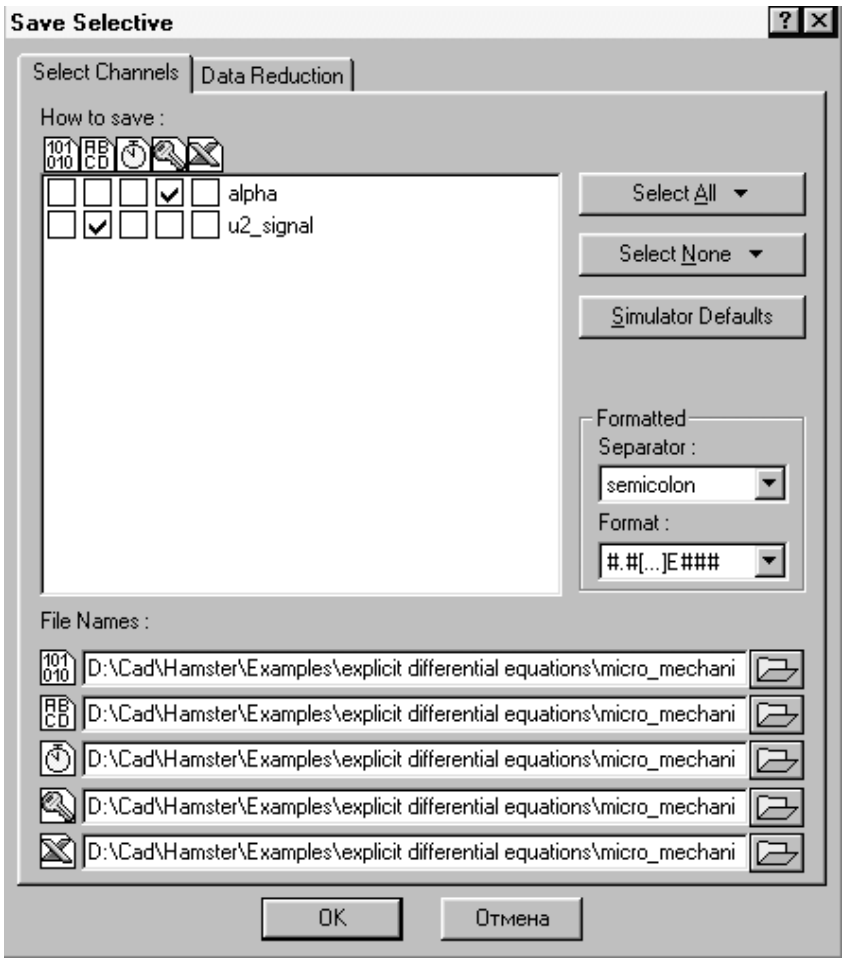

Рис.8. Меню Save Selective

В разделе How to save меню Save Selective необходимо указать формат сохранения результатов моделирования. Выбор формата осуществляется нажатием левой кнопкой «мыши» на квадрате, расположенном слева от названия переменной. После выполнения этого действия в квадратике должна появится галочка (см. рис. 8). Щелчок левой кнопкой «мыши» на квадратике с галочкой приводит к снятию выбора формата сохранения результатов моделирования.

В разделе File Names меню автоматически указывает путь сохранения результатов моделирования в рабочий каталог, который был установлен при сохранении VHDL-AMS модели устройства в текстовом редакторе Model Editor. Также предлагается названия файлов результатов моделирования. При необходимости данные настройки могут быть изменены.

Рекомендуется сохранить результаты моделирования в формате программы Microsoft Excel ().

5. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Ознакомиться со сведениями о языке VHDL-AMS, изложенными в настоящем руководстве.
2. Получить у преподавателя вариант задания по лабораторной работе.
3. Выполнить последовательно все этапы моделирования, изложенные в разделе 4.
4. Показать результаты моделирования преподавателю.
5. Сохранить файлы модели на дискете.
6. Оформить отчет о выполнении лабораторной работы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Модули модели в языке VHDL-AMS.
2. Переменные класса quantity.
3. Что такое terminal?
4. Переменные класса branch_quantity.
5. Способы представления моделей в языке VHDL-AMS.
6. Атрибуты переменных класса quantity.
7. Отличие языка VHDL-AMS от VHDL.
8. Зарезервированные слова языка VHDL-AMS.
9. Основные типы данных языка VHDL-AMS.
10. Тип данных nature.
11. Какие операторы используются для взаимодействия дискретных и аналоговых переменных?
12. Синтаксис оператора if.
13. Синтаксис оператора case.
14. Синтаксис оператора process.
15. Синтаксис оператора break.
16. Синтаксис модуля library.
17. Синтаксис модуля entity.

18. Синтаксис модуля architecture.
19. Синтаксис модуля package.
20. Синтаксис объявления портов класса terminal.
21. Синтаксис объявления портов класса quantity.
22. Синтаксис объявления портов класса signal.
23. Стили описания моделей в языке VHDL-AMS.
24. Редакторы программы hAMSter.
25. Порядок моделирования элементов МСТ в программе hAMSter.
26. Файлы модели в программе hAMSter.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. User Manual hAMSter (URL: http://www.hamster.com).
2. Авдеев Е.В. Аналоговые и смешанно-сигнальные расширения VHDL: Учебное пособие. М.: МИЭТ, 2000.– 92с.
3. A.Dewey, J.H.Hillman, B.Hillman, H.Dussault, G.Fedde, E.Christen, K.Bakalar, H.Carter, B.F.Romanowics. VHDL-AMS modeling considerations and styles for composite systems. Version 2.0 (URL: http://www.hamster.com).
4. Ивченко В.Г. Применение языка VHDL при проектировании специализированных СБИС: Учебное пособие. Таганрог: Изд-во ТРТУ, 1999. 80 с.

СОДЕРЖАНИЕ

Лысенко Игорь Евгеньевич
Рындин Евгений Адальбертович

Руководство
к лабораторной работе

МОДЕЛИРОВАНИЕ СЕНСОРНЫХ И АКТЮАТОРНЫХ ЭЛЕМЕНТОВ МИКРОСИСТЕМНОЙ ТЕХНИКИ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА VHDL-AMS

Ответственный за выпуск *Лысенко И.Е.*
Редактор *Селезнева Н.И.*
Корректор *Селезнева Н.И.*

ЛР 020565 от 23 июня 1997 г. Подписано к печати
Печать офсетная Бумага офсетная
Формат 60 x 841/16
Усл. п. л. – 1,6 Уч.- изд. л. – 1,4
Заказ N_____ Тираж 150 экз.

"С"

Издательство Таганрогского государственного
радиотехнического университета
ГСП 17А, Таганрог, 28, Некрасовский, 44
Типография Таганрогского государственного
радиотехнического университета
ГСП 17А, Таганрог, 28, Энгельса, 1