Z - затраты в инфраструктуре;

E - издержки движения.

Все это оценивается в денежных единицах.

Если исследуемая стоимость близка к минимально возможной, то считается, что система работает оптимально, без потерь. Если же эта стоимость не минимальна, то имеют место ***потери***, *под которыми понимают превышение исследуемой стоимости над минимально возможной*:

$$П = C + C_{min}, \quad (2)$$

$П$ - потери в исследуемой системе;

C - исследуемая стоимость транспортного обслуживания (транспортной услуги);

C_{min} - минимально возможная стоимость.

Потери от издержек, равно как и сами издержки, можно разделить на четыре вида – экономические, экологические, аварийные и социальные. Все эти виды довольно тесно связаны между собой, и иногда бывает трудно провести между ними четкую границу.

Экономические потери [2,3] в дорожном движении связаны с необязательными задержками (снижением скорости в сравнении с нормативной), остановками и перепробегом транспорта, задержками пассажиров и пешеходов, перерасходом топлива, износом или повреждением транспортного средства из-за некачественных условий движения и т.д. Сюда же относятся потери прибыли участниками движения и потери в смежных отраслях из-за невыполнения принятых обязательств, например, из-за опозданий или поломок в дороге и т.д.

Экологические потери – это превышающие минимально возможные выбросы вредных веществ в атмосферу, загрязнение воды и почвы, воздействие шума, вибрации и электромагнитных излучений. Основными причинами повышенного уровня экологических потерь являются: перегрузки отдельных участков улично-дорожной сети; повышенный уровень маневрирования интенсивных потоков, включая торможения, остановки и разгоны; вынужденное снижение скорости и движение на неэкономичных режимах; перепробег в любых его проявлениях; неудовлетворительное техническое состояние транспортных средств и т.д.

Под ***аварийными*** понимают все потери от аварий любых видов и любой тяжести последствий, а также судебные и иные издержки, связанные с авариями.

Под ***социальными*** понимают все потери, связанные с нарушением прав и свобод человека, законопослушанием и духовным развращением личности.

Сопоставление потерь может производится по «*приведенным*» потерям, включающим в себя обе составляющие – экономическую и социальную:

$$П = П_e + П_c, \quad (3)$$

$П$ - приведенные потери данного вида руб./год;

$П_e$ - экономическая составляющая потерь данного вида, руб./год;

$П_c$ - социальная составляющая потерь данного вида, руб./год.

Выводы. Все виды потерь являются социально-экономическими и имеют две составляющие – материальную и духовную или экономическую и социальную. Экономическая составляющая – это та часть потерь, которая имеет однозначный денежный эквивалент, например, стоимость поврежденных машин или грузов при аварии, или оплата листков нетрудоспособности из-за экологических воздействий на человека и т.д.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Врубель Ю.А. Организация дорожного движения.- Мн.: Фонд безопасности движения МВД Республики Беларусь. - Часть 1, 1996. - 20 с.
2. Луконин В.Н. и др. Автотранспортные потоки и окружающая среда.- М.: ИНФРА-М, 200. – С. 35-38.
3. Врубель Ю.А.Потери в дорожном движении. – Мн.: БНТУ, 2003. – С. 84-91.

Материал поступил в редакцию 22.01.08

KAPSKI D.V. The classification analysis of losses in road movement

In clause all kinds of losses which are considered are socio economic and have two components - material and spiritual, or economic and social. Economic making is that part of losses, which has a unequivocal money's worth, for example, cost of the damaged machines or cargoes at failure, or payment листков of invalidity because of ecological influences on the man etc.

УДК 681.3

Муравьев Г.Л., Шуть В.Н., Мухов С.В.

АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ ПО ИХ ПРОЦЕССНЫМ ОПИСАНИЯМ

Введение. Сложные технические системы предполагают наличие большого числа элементов и типов элементов, взаимного влияния узлов и процессов, наличие управления. К их числу относятся цифровые системы, устройства и их проекты, для моделирования которых нужны эффективные средства. При реализации таких моделей следует ориентироваться на процессный способ описания систем и процессную организацию имитации протекающих в них процессов [1-2].

В работе исследуется подход к построению результативных моделей систем, для которых получены процессные описания. Рассматриваемые вопросы иллюстрируются применительно к языку VHDL (стандарты в области автоматизации проектирования VHDL'93 - ANSI/IEEE Std 1076-1993 и VHDL-AMS - Std 1076.1-1999) [3-7].

VHDL предоставляет разнообразные средства и стили построения описаний, включая структурные, потоковые и поведенческие. Показано, что исходные модельные описания могут быть преобразованы в однородную промежуточную модель на базе процессных описаний, удобную для дальнейшего построения исполнимых имитационных моделей [8-12].

Процессное описание и его реализация. Общая схема получения результативных моделей предполагает следующие этапы.

1. Описание моделируемой системы (проекта), получение исходных текстов в произвольном стиле языка VHDL в виде композиции параллельных операторов (concurrent_statement) типа block, вызов процедуры procedure_call, утверждение assertion, параллельных операторов назначения сигнала concurrent_signal_assignment и др.
2. Входной контроль описания. Построение информационной базы описаний проекта, удобной для проведения дальнейших преобразований.
3. Трансформация исходного описания в процессное с целью дальнейшей генерации функционально-адекватных исполнимых описаний на языке программирования высокого уровня. Процессное описание состоит из параллельных операторов одного типа – process.

Муравьёв Геннадий Леонидович, *кандидат технических наук, профессор кафедры интеллектуальных информационных технологий Брестского государственного технического университета (БрГТУ).*

Шуть Василий Николаевич, *кандидат технических наук, доцент кафедры интеллектуальных информационных технологий БрГТУ.*

Мухов Сергей Владимирович, *доцент кафедры интеллектуальных информационных технологий БрГТУ.*

Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.

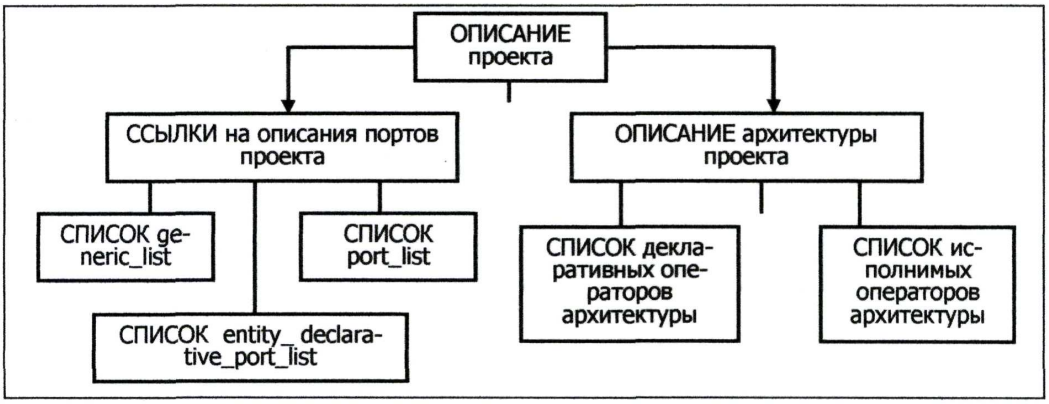

Рис. 1. *Описание проекта (системы)*

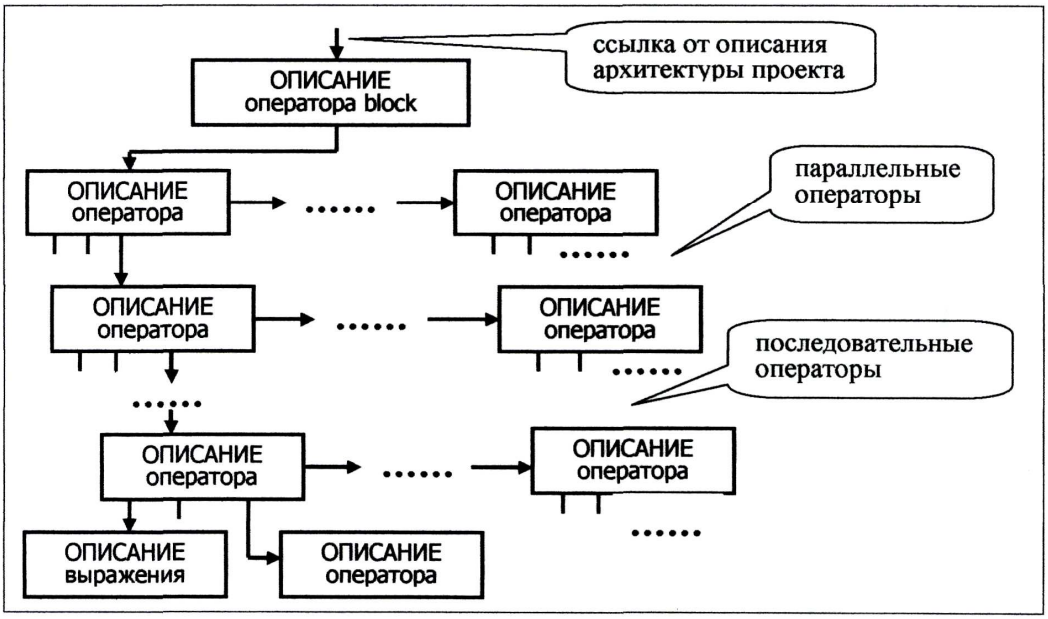

Рис. 2. *Описание тела архитектуры*

{ [label] : PROCESS [(< SENSITIVITY_list>)]
begin
{ sequential_STATEMENT }
{ sequential_WAIT_statement }
{ sequential_SIGNAL_ASSIGMENT_statement }
end process }.

Это произвольные процессы, описанные пользователем и содержащие любое число состояний. В том числе, это процессы со списком чувствительности < SENSITIVITY_list > к изменениям значений в формирователях сигналов. Это процессы-эквиваленты других параллельных операторов (типа procedure_call, assertion, разных вариантов использования оператора signal_assignment и т.п.) как частный случай процесса со списком чувствительности и единственным собственным состоянием. Процесс-эквивалент всегда активен, не блокируется на время, всегда содержит охранное выражение <GUARD_expression>.

4. Трансформация процессного описания в результативное модельное с целью обеспечения его имитационного моделирования [10]. Реализуется путем генерации исполнимых описаний параллельных операторов process и последовательных операторов процессов (sequential_statement). На языке программирования (например, на С) процесс может быть представлен адекватной функцией. Для произвольного процесса, описанного пользователем, это

```
process_<имя_процесса> (void)
{ <описание_управления_процессом>
                    <операторная_часть_процесса> },
```

где сегмент <описание_управления_процессом> определяет выбор точки входа в операторную часть процесса в зависимости от его состояния, а сегмент <операторная_часть_процесса> реализует алгоритм преобразования информации. Процессу со списком чувствительности, процессу-эквиваленту соответствует функция

```
process_<имя_процесса> ( )
{
 if (State = = 1) goto M1;
 M0: <операторная_часть_процесса>
 State =1;
 return;
 M1: if (LIST_Stable (…)) return;
 goto M0;
}.
```

Если параллельный оператор не запускается при инициализации, то он записывается

```
process_<имя_процесса> (void)
{
        if ( GUARD_Stable (…) && LIST_Stable (…) ) return;
        <операторная_часть_процесса>
 }.
```

Здесь State хранит текущее состояние процесса, а функции GUARD_Stable (…) и LIST_Stable (…) “следят” за изменением сигналов, входящих соответственно в охранное выражение или список чувствительности процесса.

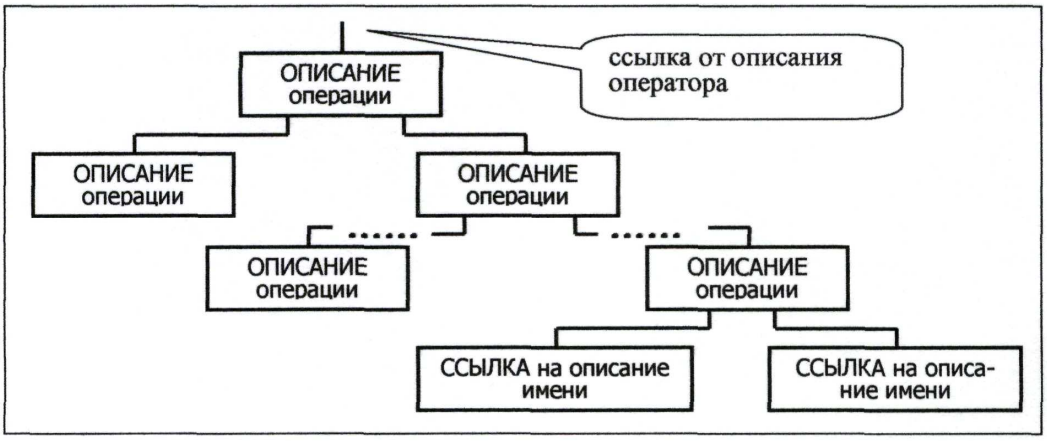

Рис. 3. *Описание выражения*

При моделировании последовательных операторов следует учитывать особенности специфических операторов типа wait_statement и sequential_signal_assigment_statement. Каждый i-й оператор wait определяет Mi состояние процесса. В зависимости от способа описания можно выделить разные комбинации использования wait, использующие для задания условий блокировки процесса значение времени t, выражение f, список чувствительности <SENSITIVITY_list>. Соответственно модельный эквивалент оператора может включать задание типа блокировки [block_type = ...;], времени [delta = t;], состояние state = Mi; [return;] label_Mi: [if ([{LIST_Stable (…) }] [&&] [not f]) return]; .

Последовательный оператор назначения сигналов описывается <имя_сигнала> <= [transport] <новые_компоненты_сигнала>, где каждый компонент – это пара (<значение_сигнала> ; <время>). Он может быть представлен вызовом функции, циклически модифицирующей формирователь соответствующего сигнала путем обновления его содержимого для каждой компоненты (вставки новых значений, удаления неактуальных значений).

Информационная база процессных описаний. Информационная база описаний представляет собой набор взаимосвязанных структур данных (например, типа struct языка C), отображающих компоненты, элементы компонентов системы в терминах проекта VHDL: интерфейсов, архитектур, параллельных операторов блоков, процессов, последовательных операторов и т.п. [11-12].

Основу базы описаний образует структура данных типа ПРОЕКТ (рис.1). Она ссылается на другие структуры данных. Это структуры, отображающие декларативную информацию (описания интерфейса, сигналов и т.п.), и структуры, отображающие тело архитектуры (block) в виде составляющих его параллельных и последовательных операторов.

Здесь каждый элемент списка (описание отдельного оператора архитектуры) ссылается на структуры с описаниями операторных выражений типа ВЫРАЖЕНИЕ (рис. 3) и может содержать ссылку на последующий оператор архитектуры.

Функциональность операторов раскрывается через описание составляющих их выражений. Само выражение может быть представлено деревом (деревом термов), которое может храниться в виде списковых структур. Листья дерева - структуры, описывающие объекты, участвующие в выражении (имена функций, переменных, элементы массивов, сечения и т.д.).

Соответственно каждый элемент списка (описание одной операции - <код_операции>) имеет формат <код_операции> <аргумент1> <аргумент2> и т.д., где в качестве аргументов используются ссылки на другие операции выражения либо ссылки на описание используемых в выражении имен объектов, представленных структурами данных, отображающими их декларацию.

Алгоритмы, система обработки процессных описаний. Генерация имитационных моделей выполняется путем анализа процессной модели системы (проекта), описанной на языке VHDL, и ее обработки по заранее установленным правилам преобразования конструкций модели в конструкции языка C [10-12].

Система может быть организована в виде головной программы ОБРАБОТЧИК ОПИСАНИЙ ПРОЕКТА и вызываемых ею модулей для инициирования процедур анализа и обработки текстов соответствующих частей процессной модели. Соответственно система должна включать модули: ОБРАБОТЧИК заголовка, ОБРАБОТЧИК тела проекта (architecture – внешнего оператора block), ОБРАБОТЧИК подпрограмм, используемых в проекте.

ОБРАБОТЧИК тела проекта базируется на многократном использовании ГЕНЕРАТОРОВ кодов параллельных и последовательных операторов. Они в свою очередь рекурсивно используют АНАЛИЗАТОРЫ выражений, описывающих операторы. Поскольку операторы представлены списками, описывающими деревья термов, то соответствующий текст описания исполнимых операторов генерируется модулем АНАЛИЗИРОВАТЬ-ТЕРМ, используемым рекурсивно для анализа дерева термов каждого выражения.

Кроме этого, для отображения деклараций, используемых в проекте, применяются ГЕНЕРАТОРЫ описаний портов (входных - выходных сигналов), описаний типов объектов проекта (сигналов, переменных). Обобщенные алгоритмы работы системы представлены ниже:

```
ОБРАБОТАТЬ-ОПИСАНИЯ-ПРОЕКТА
        Генерировать заголовок проекта
        Генерировать раздел описаний объектов проекта
        Создать описания формирователей сигналов
        НомерПараллельногоОператора = 1
        ОБРАБОТАТЬ-БЛОК (НомерПараллельногоОператора)
        Завершить обработку проекта,

ОБРАБОТАТЬ-БЛОК (НомерПараллельногоОператора)
        Генерировать заголовок блока
        Генерировать раздел описаний объектов блока
        ЦИКЛ-ПОКА не обработаны все параллельные операторы
блока
    Анализировать текущий оператор
        ЕСЛИ это block ТО
                Установить НомерБлока
                ОБРАБОТАТЬ-БЛОК (НомерБлока)
    ИНАЧЕ ЕСЛИ это process ТО
                Установить НомерПроцесса
                ОБРАБОТАТЬ-ПРОЦЕСС (НомерПроцесса)
        ИНАЧЕ ЕСЛИ это формирователь ТО
                Генерировать текст формирователя
        КОНЕЦ-ЕСЛИ
    Перейти к следующему оператору
    КОНЕЦ-ЦИКЛА
        Завершить обработку блока,

ОБРАБОТАТЬ-ПРОЦЕСС (НомерПроцесса)
        Генерировать заголовок процесса
        Генерировать раздел описаний объектов процесса
        Установить НомерТерма = НомерПервогоТермаПроцесса
        АНАЛИЗИРОВАТЬ-ТЕРМ (НомерТерма)
```

```
        Завершить обработку процесса,

    АНАЛИЗИРОВАТЬ-ТЕРМ (НомерТерма)
        Анализировать и запомнить КодОперацииТерма указанного терма
        Начать формировать текст соответствующего оператора
    Анализировать первый ОперандТерма
        ЕСЛИ ОперандТерма - объект (константа, имя переменной и т.п.) ТО
    ОБРАБОТАТЬ-ОПЕРАНД
        ИНАЧЕ /* ОперандТерма – ссылка на следующий терм */
        Установить НомерТерма = ОперандТерма
        АНАЛИЗИРОВАТЬ-ТЕРМ (НомерТерма)
        КОНЕЦ-ЕСЛИ
        [ Анализировать второй ОперандТерма … ]
        [ Анализировать третий ОперандТерма … ]
Завершить обработку выражения.
```

Заключение. В работе сформулированы подходы к получению текстов моделей, ориентированных на имитационный расчет, по их процессным описаниям. Рассмотрены исполнимые формы процессных описаний, ориентированные на реализацию средствами языков программирования высокого уровня.

Применительно к языку VHDL рассмотрена структура программного обеспечения преобразования входного описания проекта в имитационную модель. Это включает перевод произвольных описаний в процессные с последующей трансформацией процессных описаний и их операторов в функционально-адекватный текст на языке С.

Система предназначена для использования в САПР в составе проектной процедуры моделирование. Может служить для обеспечения задач моделирования всех уровней описания проекта, допустимых языком VHDL. В широком смысле обеспечивает независимость модели проекта, используемой для проведения имитационных экспериментов, от формы его исходного представления.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Максимей И.В. Имитационное моделирование на ЭВМ. - М.: Радио и связь, 1988. – 232 с.
2. VHDL для моделирования, синтеза и формальной верификации аппаратуры/Пер. с англ. - М.: Радио и связь, 1995. – 360 с.
3. Бибило П.Н. Синтез логических схем с использованием языка VHDL. - М.: “СОЛОН-Р”, 2002. – 384 с.
4. Бибило П.Н. Основы языка VHDL. - М.: “СОЛОН-Р”, 2000. – 210 с.
5. Сергиенко А.М. VHDL для проектирования вычислительных устройств. - К.: “Корнейчук”, 2003. – 208 с.
6. Армстронг Дж. Р. Моделирование цифровых систем на языке VHDL. - М.: Мир, 1992. – 175 с.
7. Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. - М.: “СОЛОН-Пресс”, 2003. – 320 с.
8. Муравьев Г.Л., Шуть В.Н. Интерпретация VHDL-описаний, согласованная с процессным способом моделирования// Вестник БГТУ.– 2005.– № 5(35).– С. 81-84.
9. Прихожий А.А., Муравьев Г.Л. Система проектирования цифровых СБИС с языком VHDL на ПП ЭВМ. Разработка и использование ПЭВМ// Труды международного симпозиума INFO' 89. - Мн., 1989. - т.2, ч.1. – 6 с.
10. Муравьев Г.Л. Трансформация VHDL-проектов СБИС в модели на языке Си// Сборник трудов 10-й научно-технической конференции “Новые технологии в машиностроении и вычислительной технике”. – Брест: Брест. политехн. ин-т, 1998.
11. Прихожий А.А., Муравьев Г.Л. и др. Система автоматизированного проектирования СБИС. Процедуры проектирования. Материалы по математическому обеспечению ЭВМ. - Мн., ИТК АН РБ, 1992. – 98 с.
12. Дудкин А.А., Головко В.А. , Муравьев Г.Л. и др. Алгоритмы и подсистемы автоматизированного логического проектирования цифровых СБИС. Материалы по математическому обеспечению ЭВМ. - Мн., ИТК АН РБ, 1994. – 120 с.

Материал поступил в редакцию 04.02.08

MURAVJEV G.L., SHUT V.N., MUCHOV S.V. Interpretation Automation of construction of imitating models on their processes to the descriptions

The approaches to reception of the executed texts of models on their processes to the descriptions are considered. The forms to processes of the descriptions focused on realization by means of the programming languages of a high level are resulted.

УДК 528.063

Грищенков Е.В., Зуева Л.Ф., Синякина Н.В.

ПРИМЕНЕНИЕ МЕТОДОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ ПРОЕКТИРОВАНИИ НАЗЕМНЫХ ПРОСТРАНСТВЕННЫХ ГЕОДЕЗИЧЕСКИХ ЗАСЕЧЕК

Введение. Статья посвящена реализации на ПК методов оптимального проектирования линейной и угловой пространственных засечек. Основной целью является сравнение различных методов нелинейного программирования по трем основным критериям:

- область сходимости итераций;
- надежность решения при плохообусловленных системах нелинейных уравнений;
- точность локализации минимума целевой функции за наименьший промежуток времени.

Большая часть указанных характеристик известна из литературных источников [1, 2, 3]. Например, метод релаксации имеет радиус области сходимости $6S_{ср}$. Достигается минимум за большое количество приближений, но общее затрачиваемое время сопоставимо практически со всеми известными методами нелинейного программирования. Так как вычисления на ПК осуществляются с большой скоростью, различия во времени вычислений в реальности незначительно.

Преимущества других методов нелинейного программирования снижаются из-за потребности в нахождении первых и вторых частных производных, точное вычисление которых проблематично.

В работе [2] впервые указан недостаток метода Коши, заключающийся в пилообразности траектории минимизации, в окрестности минимума плохообусловленной целевой функции.

Метод Якоби может дать деление на ноль при отсутствии ускорения в изменении целевой функции.

Метод Ньютона, как указано во многих литературных источниках, имеет самую небольшую область сходимости, равную $0{,}3S_{ср}$,

Грищенков Евгений Викторович, *аспирант кафедры прикладной геодезии и фотограмметрии Полоцкого государственного университета (ПГУ).*

Беларусь, ПГУ, 211440, г. Новополоцк, ул. Блохина, 29.

Зуева Людмила Федоровна, *доцент кафедры оснований, фундаментов, инженерной геологии и геодезии Брестского государственного технического университета (БрГТУ).*

Синякина Наталья Васильевна, *доцент кафедры оснований, фундаментов, инженерной геологии и геодезии (БрГТУ).*

Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.