7. Арипов, М.Н. Передача дискретной информации по низкоскоростным каналам связи/ М.Н. Арипов - М: Связь, 1980.
8. Злотник, Б.М. Помехоустойчивые коды в системах связи /Б.М. Злотник - М.: Радио и связь, 1989.
9. Прокис, Джон Цифровая связь./ Дж. Прокис.- Пер.с англ./ Под ред. Кловского Д.Д. – М.: Радио и связь. 2000.

*К.т.н., доцент **Матвеев Б.В.** тел. (4732)-43-76-65, Воронежский государственный технический университет, кафедра радиотехники*

УДК 004.322

ИССЛЕДОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ ДЛЯ СОЗДАНИЯ МОДЕЛЕЙ ПРОТОКОЛОВ ПЕРЕДАЧИ ДАННЫХ И ИХ ВЕРИФИКАЦИИ

В.Л. Оленев

Предложен алгоритм перехода от спецификации протокола передачи данных к созданию архитектурной диаграммы модели и далее – скелета этой модели на языке программирования высокого уровня. Так же предложен алгоритм верификации архитектурной диаграммы с целью выявления ошибок, способных осложнить компиляцию скелета модели протокола.

Ключевые слова: связь, протоколы, моделирование, верификация

Для безопасной, качественной и быстрой организации связи между устройствами используют протоколы передачи данных. Это наборы правил, которые определяют обмен данными между различными устройствами и программными средствами. Протокол определяет временные характеристики сигналов и структуру передаваемых данных. Сетевые протоколы определяют также и правила взаимодействия устройств в составе сети.

В настоящее время моделирование протоколов передачи данных является очень трудоемкой и ресурсоемкой задачей. Их проектирование сопряжено с рядом трудностей, связанных с увеличением сложности проектов, повышением требований к надежности и потребляемой мощности изделий, работающих по данному протоколу, а так же необходимостью завершения проекта в кратчайшие сроки. Работая в данной сфере, важно использовать различные инструменты автоматизации, которые способны ускорить и облегчить процесс разработки и написания модели [1, 2]. В данной статье будет описан подобный метод, который позволит:

Упростить проектирование модели, основываясь на спецификации;

Описать архитектуру модели протокола, используя набор стандартных модулей, каждый из которых несет свою конкретную функциональность;

Проверить полученную модель на наличие ошибок проектирования;

Сгенерировать скелет модели на SystemC, основываясь на спроектированной архитектуре.

Полученный код должен компилироваться без ошибок.

Постановка задачи

Для разработки системы моделирования протоколов передачи данных для начала нужно определить набор блоков (модулей), из которых будет состоять модель. Для этого нужно проанализировать, какая же функциональность описана в спецификациях протоколов и какие могут понадобиться модули для разработки архитектурных диаграмм.

Спецификация – это базис для создания протокола, так как реализация протокола должна отвечать всем требованиям, описанным в этой спецификации. Как правило, в достаточно подробной спецификации протокола описаны следующие части и компоненты:

Общее представление о протоколе, сфера его применения;

Описание данного протокола по слоям;

Описание набора функций, которые несет в себе слой;

Возможно, описание интерфейсов взаимодействия с другими слоями (SAP'ов);

Возможно, описание функций менеджмента слоя;

Описание взаимодействия между устройствами в составе сети; может быть частично раскидано по слоям;

Подробное описание конкретных наиболее сложных процедур;

В ненормативной части могу быть описаны пожелания к реализации тех или иных моментов, описанных выше по тексту.

Из этой общей структуры можно сделать вывод о тех модулях, которые необходимы и достаточны для моделирования любого протокола передачи данных.

Модуль инициализации. В данном модуле инициализируются все остальные модули. Использование модуля инициализации позволяет применять вложенность, то есть делить модули на подмодули. Это позволяет структурировать модель и логически разбивать на секции;

Стандартный модуль (N портов и M интерфейсов) изображен на рисунке 1.

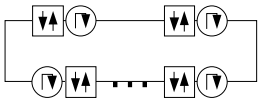

Рисунок 1 - Стандартный модуль

Стандартный модуль, который имеет несколько входов и выходов, а именно портов и интерфейсов, которые дают наибольшую независимость модулей при разработке моделей [3]. Особенности работы каждого модуля пишутся разработчиком вручную.

Полу-дуплексный и полнодуплексный каналы, которые показаны на рисунке 2.

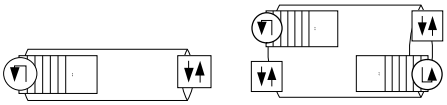

Рисунок 2 - Полу-дуплексный и полнодуплексный каналы

Модули принимают и передают данные одного или несколько типов в одном или двух направлениях. Объекты передаются через канал путем записи в FIFO и чтения из него. Используется для моделирования передачи данных между устройствами или какими-то отдельными модулями, в которых требуется накопление данных.

Сервисная точка доступа (SAP) изображена на рисунке 3.

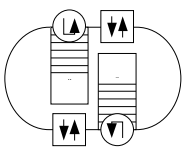

Рисунок 3 - Сервисная точка доступа

Организация SAP является более сложной, чем канал структурой, которая способна передавать различные примитивы. SAP должен использоваться в сочетании с двумя стандартными модулями для организации передачи данных от разных модулей через один SAP.

Таким образом, мы определили основные модули, которые требуются для моделирования протоколов передачи данных.

Особенности алгоритмической реализации модели

На самом деле реализовать модель протокола можно и на комбинации модулей инициализации и стандартных модулей. А потом уже вручную дописывать каналы и SAP, организовывать в них FIFO. Поэтому необходимым набором для моделирования является лишь набор из первых двух модулей. Но наличие модулей каналов и SAP в значительной степени упростит и ускорит работу.

В любой спецификации в описании любого уровня всегда присутствует список основных функций, выполняемых уровнем, от которого и можно отталкиваться при дальнейшей разработке. По названиям и количеству функций можно определить нужное количество и название модулей, необходимых для моделирования слоя.

Для иллюстрации работы алгоритма предположим, что дан сетевой уровень какого-либо протокола L3, в котором описан ряд функций, а также два SAP к выше- и нижележащему уровням:

Сбор пакета сетевого уровня из нескольких фреймов канального;

Выделение и обработка адреса;

Генерация сообщения об ошибке адреса.

Таким образом, для моделирования этого слоя понадобится:

Модуль инициализации, отвечающий за сам слой L3

Три стандартных модуля;

Два модуля SAP;

Два стандартных модуля для преобразования и создания примитивов для SAP.

Блок packet_constructor должен осуществлять сбор пакета сетевого уровня из нескольких фреймов канального уровня, а соответственно и обратное деление пакета на фреймы. Поэтому вместо стандартного модуля удобнее использовать блок «Полнодуплексный канал» для возможности накопления пакетов. Предварительная архитектура слоя показана на рисунке 4.

Для верификации систем с конечным числом состояний часто применяется метод верификации моделей (или проверки на модели). Обычно процедура верификации за-

ключается в исчерпывающем обходе пространства состояний системы, для того чтобы выяснить, выполняется ли спецификация. При наличии достаточных ресурсов эта процедура всегда завершается с ответом «да» или «нет». Более того, она может быть реализована достаточно эффективным алгоритмом, способным работать на вычислительных машинах невысокого класса. Метод верификации моделей применим ко многим важным классам вычислительных систем, к каким относятся и коммуникационные протоколы [4]. Однако такой метод сложно применить, если заранее не известно количество модулей, которое будет использовано в модели и каким образом они будут связаны.

Методика построения автомата

В данной части статьи предлагается алгоритм перехода от архитектурной диаграммы модели к сети Петри (автомату Петри) для последующей верификации.

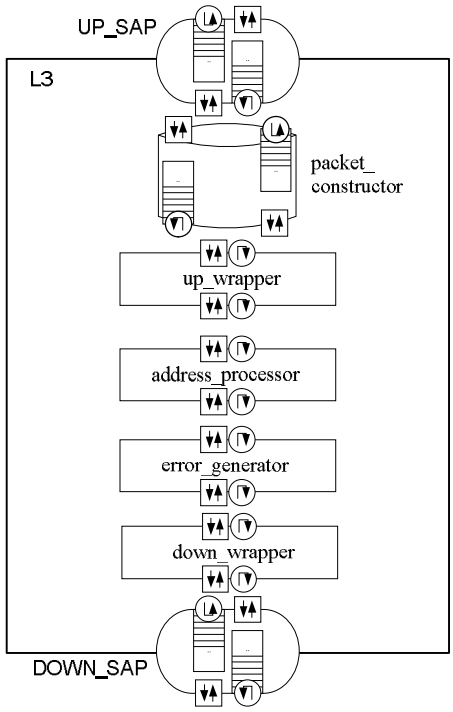

Рисунок 4 - Предварительная архитектура слоя L3 без расставленных связей

После этого необходимо будет расставить взаимосвязи между модулями и определить количество интерфейсов и портов для каждого модуля, кроме SAP'ов и каналов, для которых их количество фиксировано.

Далее нужно определить ту информацию, которая является входной и выходной для каждого модуля. Для этого строится таблица связей. В нее для каждого модуля выписываются методы, отвечающие за передачу и прием данных. Для каждого метода ставится признак, говорящий о выполняемых им функциях:

→ метод, принимающий данные

← метод, отправляющий данные

После описания методов и их признаков следует расписать данные, которые передаются или принимаются при помощи этого метода. После этого обозначаются порты и интерфейсы модулей. Название дается следующим образом: если это порт, то название заканчивается на «_port», а если это интерфейс, то на «_if». Названия интерфейсов должны быть уникальны, так как для каждого из них создается отдельный файл заголовка.

Стоит обратить внимание, что если порт и интерфейс связаны с друг другом, то вызываемая портом и предоставляемая интерфейсом функция – это один и тот же метод. Соответственно, они будут иметь одно и то же название в таблице.

Далее по связям порт-интерфейс ставятся метки (от 0 до бесконечности), которые идентифицируют связи между блоками. Это в дальнейшем упростит процесс верификации архитектурной диаграммы.

Таким образом, составив подобную таблицу, мы получим основные данные, необходимые для создания скелета модели:

Назначение и названия модулей;

Методы и названия этих методов, отвечающих за межмодульное взаимодействие;

Назначение и названия портов и интерфейсов между модулями;

Методы, которые должны быть описаны в соответствующих интерфейсах;

Порты, которые должны быть связаны с соответствующими интерфейсами;

Также подобная таблица связей, составленная по архитектурной диаграмме слоя, дает данные, необходимые для верификации диаграммы.

Отметим, что таблица строится параллельно с тем, как производится проектирование архитектурной диаграммы. Это позволяет сразу решить такие проблемы как:

повторные имена интерфейсов;

одинаковые имена портов в рамках одного модуля;

отсутствие названий портов, интерфейсов или методов;

получение списка передаваемых данных для каждой связи, что понадобится в дальнейшем для проверки диаграммы;

получение множества передаваемых данных в рамках диаграммы в целом;

После этого шага схема уровня будет выглядеть как показано на рисунке 5.

Но во время составления такой схемы очень велика вероятность допущения ошибки. Поэтому необходим механизм верификации схемы, который может исключить ошибки определенных типов, поскольку наличие таких ошибок не позволит скомпилировать автоматически сгенерированный код.

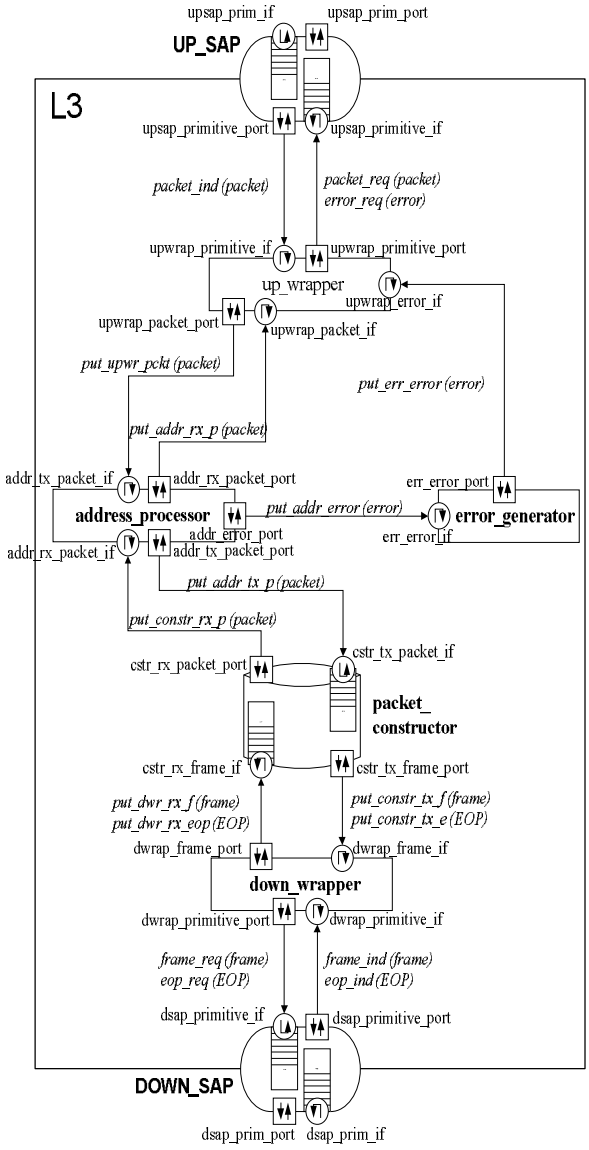

Рисунок 5 - Архитектурная диаграмма слоя с расставленными связями

Верификация автомата

Теоретико-графовым представлением сети Петри является двудольный ориентированный мультиграф (автомат), а ее структура представляет собой совокупность позиций (состояний) и переходов. В соответствии с этим, граф сети Петри обладает двумя типами узлов. Кружок (O) является позицией, а планка (I) – переходом [5].

Из полученной в предыдущем пункте архитектурной диаграммы слоя составим сеть Петри по следующему принципу:

каждый модуль, участвующий в передаче данных, будет являться состоянием сети;

порт ( ) – переход к условию;

интерфейс ( ) – переход от условия;

условие – типы данных, которые должны быть переданы от модуля к модулю;

Каждый SAP или точка вывода данных из модели снабжается дополнительным состоянием и одним переходом в него. Условия перехода (передаваемые данные) будут соответствовать множеству всех условий перехода в предыдущее состояние. При отсутствии SAP точка ввода данных определяется вручную.

После составления сети Петри можно осуществить первую проверку на наличие всех связей между портами и интерфейсами, поскольку язык SystemC чувствителен к этому показателю и все порты должны быть связаны.

Если все состояния связаны друг с другом соответствующим образом и нет несвязанных портов и интерфейсов, то первая проверка пройдена.

После этого необходимо разметить сеть Петри для ее запуска и проверки. Для этого составляется список данных, передаваемых по сети Петри:

Data={data_1, data_2, data_3,…, data_n}

Каждому типу данных присваивается своя цветная фишка. Переходу, где осуществляется только вызов метода без передачи каких-либо данных в модуль (<вызов>), присваивается черная фишка. Такие фишки не анализируются в процессе проверки.

В соответствии с теорией сетей Петри примем следующие обозначения [6]:

P – множество всех состояний сети;

t1…tn – совокупность условий для перехода в конкретное следующее состояние, количество зависит от числа состояний, в которые можно совершить переход из данного состояния;

T – набор из совокупностей условий для перехода из текущего в каждое следующее состояние;

T={t1, t2,…,tn}

M – совокупность поставленных в данном состоянии фишек;

{t1b…tnb} – совокупность условий предыдущего состояния для перехода в конкретное текущее состояние;

Для каждого состояния сети должно выполняется условие:

$$M = \sum_{1}^{n} T - t_{ib} \cdot$$

Если полученная совокупность Mi не будет пустой, то фишки, полученные в Mi, ставятся в это состояние.

После этого в узле ставим фишки соответствующих цветов, руководствуясь следующим правилом. Начинаем из каждого SAP или модуля, определенного как точка ввода данных. В состояние, соответствующее SAP, ставятся фишки, необходимые для срабатывания всех переходов из SAP. Далее осуществляется срабатывание по всем возможным переходам. В состояниях, в которые был осуществлен переход, ставятся фишки, по правилу, описанному ранее. После расстановки фишек снова осуществляются все возможные переходы в следующие состояния.

Данная операция производится до тех пор, пока не сработают все возможные переходы в модели.

После этого по тому же правилу проводится разметка и срабатывание сети из каждого SAP или точки ввода данных.

Если во время вычисления Mi будет иметь данные со знаками минус. Это означает, что данные остаются в данном блоке и не идут дальше по модели. Таким образом, есть три варианта дальнейших действий:

либо проектировщик забыл указать связи для дальнейшего перехода и ошибку следует устранить;

либо данные действительно остаются в этом блоке и в дальнейшем в модели они не нужны;

либо входные данные связаны какой-либо закономерностью с выходными, и необходимо такие данные указать в таблице соответствий.

По окончании работы сети оценивается, в каких состояниях находятся фишки. И если данное представление соответствует тому, что хочет видеть разработчик, то диаграмма считается составленной верно.

Результаты моделирования и их обсуждение

Составим по архитектурной диаграмме с рисунка 5 сеть Петри по вышеописанному принципу (рисунок 6).

Далее определим множество данных, передаваемых по сети Петри, на основе выборки из столбца «данные» таблицы состояний.

PetriData={packet, error, frame, EOP}

Каждому типу данных присваиваем свою цветную фишку.

Таблица 1 Присвоение цвета фишкам

Тип данных	Цвет фишки	Пример
packet	Красный	●
error	Синий	◍
frame	Желтый	○
EOP	Зеленый	⊕

Далее произведем маркировку сети Петри и осуществим ее работу.

Маркировка будет приведена только для первых 4-х состояний автомата.

Сеть имеет 9 состояний:

P={p0, p1, p2, p3, p4, p5, p6, p7, p8};

Состояние p0: Переход из состояния невозможен. Оно добавлено с целью накопления фишек, пришедших в SAP, для дальнейшего анализа.

В данное состояние переходим из состояния p1, при этом из p1 удаляется красная фишка packet и синяя фишка error.

T= Ø;

M= Ø;

P={p0, p1, p2, p3, p4, p5, p6, p7, p8} - p0={p1, p2, p3, p4, p5, p6, p7, p8};

Состояние p1: Состояние соответствует модулю SAP, поэтому разметка будет начинаться с него.

M = {packet, error, packet} - Ø;

В состояние ставится три фишки: 2 фишки packet и 1 фишка error. При переходе в состояния p0 и p2 осуществляется удаление 3-х всех фишек.

P={p1, p2, p3, p4, p5, p6, p7, p8} - p1={p2, p3, p4, p5, p6, p7, p8};

В состояние p0 переход уже был осуществлен. Поэтому далее рассматриваем только состояние p2.

Состояние p2: В данное состояние переходим из состояний p1, p3, p4. Переход осуществляется в состояния p3, p1

t1={error};

t2={packet};

t3={packet};

M = {packet, packet, error} - {packet}={packet, error};

P={p2, p3, p4, p5, p6, p7, p8} - p2={p3, p4, p5, p6, p7, p8};

В состояние p1 переход уже был осуществлен. Поэтому далее рассматриваем только состояние p3.

После разметки, начавшейся с UP_SAP (состояния p1) сеть Петри будет выглядеть, как показано на рисунке 6.

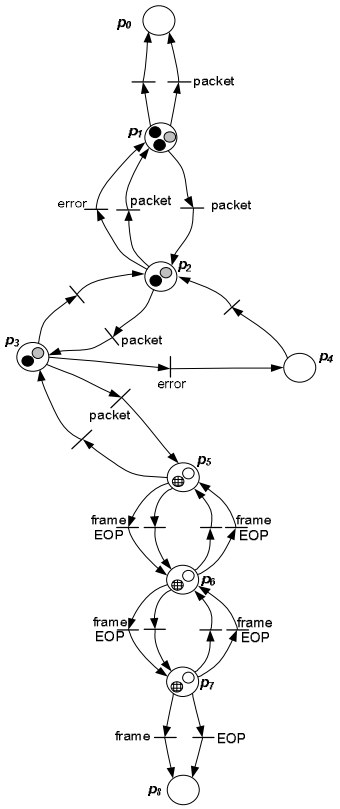

Рисунок 6 - Маркированная Сеть Петри

Переходы, инициированные из DOWN_SAP, осуществлять в данном примере не будем.

Далее осуществляется запуск модели, начиная с состояния p0. Модель работает до того момента, пока есть хотя бы один разрешенный переход.

После осуществления всех переходов модель будет выглядеть, как показано на рисунке 7.

По полученной в итоге разметке видно, что данные благополучно дошли до состояний p0 и p8. А значит, явных ошибок в схеме не обнаружено, и при компиляции возникнуть ошибок не должно.

После исследования схемы таким методом, мы смогли провести данные по всем модулям (состояниям) и переходам и увидеть, в каких из них данные остаются, а какие данные доходят до SAP и уходят дальше. Так можно обнаружить:

Ошибки организации логики работы схемы;

Ошибки отсутствующих связей, когда в блоке остаются фишки, которые не должны там оставаться;

Ошибки организации взаимодействия данных, когда:

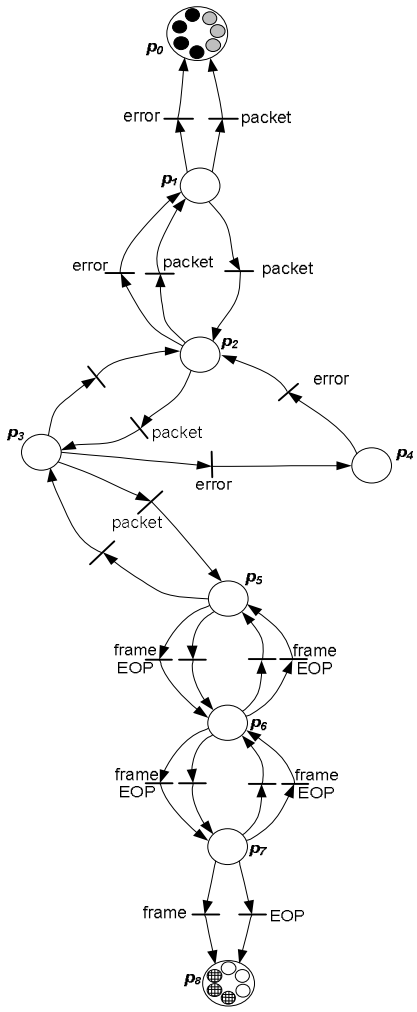

Рисунок 7 - Маркированная Сеть Петри после срабатывания

- в блоке остаются фишки, которые не должны там оставаться;
- данные приходят не в те SAP или блоки, которым они нужны;

Потенциальные зацикливания, когда данные будут ходить между двумя или более блоками, не попадая в SAP.

Таким образом, разработанный и приведенный в данной статье алгоритм способен в несколько раз ускорить и упростить разработку моделей протоколов передачи данных.

СПИСОК ЛИТЕРАТУРЫ

1. Бухтев, А. Методы и средства проектирования систем на кристалле. // А. Бухтев/ - CHIP NEWS. 2003 г. №4 (77). С. 4-14.

2. Немудров, В., Мартин Г. Системы на кристалле. Проблемы проектирования и развития./ В. Немудров, Г.М. Мартин: М.: Техносфера. 2004.
3. Оленев, В. Методы межмодульного взаимодействия при моделировании протоколов встроенных систем/ В. Оленев, Л.Онищенко, А. Еганян // Научная сессия ГУАП. –С-Пб: СПбГУАП. 2008. С. 98-99.
4. Кларк, Э. Верификация моделей программ: Model Checking /Э.Кларк и др.- М.: Московский центр непрерывного математического образования, 2002.
5. Питерсон, Дж. Теория сетей Петри и моделирование систем / Дж. Питерсон. - N.J.: Prentice-Hall. 1981.
6. Котов, В.Е. Сети Петри/ В.Е. Котов.- М.: Наука, 1984.

Аспирант, магистр техники и технологии ***Оленев В.Л.****, тел. 8-911-797-0533, e-mail: Valentin.Olenev@guap.ru, Государственный университет аэрокосмического приборостроения (г. Санкт-Петербург).*

УДК 004.322

СТАТИЧЕСКИЕ МОДЕЛИ ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ УСТРОЙСТВ КАНАЛЬНОГО УРОВНЯ

Е.Н.Яблоков

В данной статье рассмотрены существующие статические модели для оценки устройств канального уровня. В качестве альтернативы предложена новая статическая модель оценки канала, приближенная к физической реализации, что позволяет более точно смоделировать задержки при передаче данных.

Ключевые слова: канальный уровень, приемник, передатчик, статическая модель, анализ.

С помощью статических моделей можно оценить производительность системы как функцию, зависящую от различных параметров. Самая главная проблема при построении такой модели – определение необходимых параметров и их взаимодействия. Составление модели производится в несколько этапов. В самом начале берутся задержки на самом высоком уровне синтеза для встроенных систем реального времени. Это позволяет рассмотреть задержки после отображения всех высокоуровневых задач на процессорные узлы, для проверки того, что реальные задержки не превышают заданных ограничений. Если задачи на одном процессорном узле должны взаимодействовать – то добавляются небольшие задержки сверху. В то же время, если задачам разных процессорных узлам надо взаимодействовать - им необходимо соединятся друг с другом или путем шины, или соединениями точка-точка с соответствующими задержками. Эти задержки зависят от количества информации, необходимой для передачи между процессорными узлами. Обычно при таком походе общее количество передаваемых данных на линии оценивается статически, которое потом изменяется путем умножения на масштабный коэффициент коммуникационной архитектуры для получения общих коммуникационных затрат [1].

Последующие подходы фокусируются на оценке коммуникационных задержек, чтобы удостоверится, что программное обеспечение и аппаратная часть удовлетворяет ограничениям по производительности. Можно попробовать описать простую модель и описывающие коммуникационные задержки для протокола, используя простое выражение, включающее в себя количество информации, которую необходимо передать, отношение ширины канала к размеру передаваемого элемента данных, и время передачи одного элемента данных. Также можно попробовать расширить данную модель путем создания более детальной оценочной модели для программно-аппаратной архитектуры [2].

Обзор известных моделей

На рисунке 1а показана модель Кнудсена и Мадсена [3, 4], разделенная на 3 части – модель передатчика, модель канала, модель приемника. Такой подход позволяет оценить не только задержки, наложенные каналом, но и так же позволит оценить понижение производительности, наложенное программно/аппаратными драйверами. Время на установку соединения не учитывается (например, арбитраж) не моделируются, так как они не важны для соединений точка-точка.

Рисунок 1б показывает параметры драйвера, которые будут использоваться при расчете задержек на передающей модели драйвера. Драйвер получает nt слов от процессорного узла для передачи через канал и соответственно делает из них nc слов канала.