Конфигурируемое тестовое окружение для RTL-симуляции и оценки производительности сети на кристалле как части системы на кристалле

С. А. Чусов, Н. А. Моргаль, О. В. Габец, А. А. Герявенко

Национальный исследовательский университет «МИЭТ»

Аннотация: В данной статье была представлена конфигурируемая тестовая среда, предназначенная для потактовой симуляции сети на кристалле в составе СнК, целью которой является сбор и предоставление статистики о поведении сети в ходе тестирования. Среда предназначена для оценки производительности сети в составе определенной СнК на этапе разработки, когда имеется полное или частичное RTL-описание системы. Были рассмотрены возможности конфигурации среды, область ее применимости. Также были приведены общая схема работы и вычисляемые средой характеристики производительности сети на кристалле. В заключение был приведен пример применения для анализа производительности конкретной сети на кристалле в составе конкретной СнК.
Ключевые слова: сеть на кристалле, система на кристалле, оценка производительности, верификация.

Введение

Ускоренное развитие в области микроэлектроники приводит не только к общему усложнению дизайна вычислительных узлов СнК, но и к стремительному росту их количества в рамках одной системы. Современные высокопроизводительные СнК могут включать в себя сотни вычислительных блоков [1]. Задача обеспечения максимально эффективной связи между вычислительными блоками СнК в современных реалиях постепенно приобретает статус основной. Именно перемещение данных в системе, а не арифметика и общая логика управления, становится фактором, ограничивающим характеристики системы [2].

Одним из возможных решений данной задачи является построение СнК с использованием Сетей на Кристалле (Network on Chip, NoC) с различными топологиями и алгоритмами распределения данных в сети. Как правило, чем больше вычислительных узлов в проектируемой СнК, тем большее преимущество показывает NoC над стандартными типами соединений (такими, как, к примеру, общая шина) относительно

возможностей масштабирования, повторного использования дизайна и оптимизации электрических параметров [3]. Выбор топологии NoC определяется множеством факторов, а известные алгоритмы маршрутизации могут модифицироваться разработчиками для реализации всех необходимых функциональных возможностей сети.

Уже на начальном этапе проектирования, то есть при отсутствии описания дизайна NoC, перед разработчиком ставится задача оценить производительность сети, учитывая топологию и реализуемые алгоритмы маршрутизации. Для оценки производительности на начальном этапе используются симуляторы NoC.

Большинство симуляторов NoC основано на статистическом тестировании, где для каждого входного воздействия на сеть определена своя вероятность [4, 5]. Стоит отметить, что данный способ подходит для получения общих характеристик сети в условиях, лишь частично соответствующих условиям реального использования NoC с определенной СнК.

Часть симуляторов реализует подход, в котором используется набор моделей вычислительных блоков и процессоров, моделируя СнК в связке с NoC определенной топологии [6]. Данный подход дает более полную картину поведения сети, нежели способ, основанный на статистическом тестировании, так как входные воздействия определяются не статистически, а основаны на определенном поведении узлов СнК.

Стоит отметить, что не менее важна оценка производительности NoC в составе СнК на этапе разработки, когда имеется полное или частичное RTL-описание системы. Причем, в течение разработки, дизайн сети может итерационно изменяться по мере добавления новых типов вычислительных узлов в систему, а также новых протоколов связи между ними. Важно отметить, что, как правило, интерфейс, связывающий ячейку сети и

вычислительный узел системы, остается неизменным, либо изменяется в минимальной степени.

Тестовая среда, представленная в данной статье, позволяет разработчику оценить производительность RTL-дизайна NoC в составе СнК на текущем этапе разработки. При этом условия тестирования являются максимально приближенными к реальным условиям использования NoC. Это становится возможным благодаря:

- функциональной RTL-симуляции с использованием реального исходного кода NoC, учитывающего все особенности ее построения;
- использованию реального RTL-описания вычислительных блоков или же HDL-моделей, их заменяющих (что во втором случае дает возможность разработчикам использовать простые несинтезируемые модели вычислительных блоков при отсутствии их RTL-описания);
- настройке типа и положения каждого вычислительного блока в сети.

Общие сведение и возможности среды

Среда предоставляет пользователю возможность проведения функциональной RTL-симуляции NoC в составе SoC с использованием RTL-описания NoC и вычислительных узлов. Также имеется возможность использования пользователем несинтезируемых моделей вычислительных узлов SoC в случае отсутствия их RTL-описания. Пользователь может собственноручно определять набор узлов или же моделей, доступных для подключения, и задавать их позиции в сети. Среда предоставляет возможность определять и добавлять в среду параметры, необходимые для создания экземпляров сети и узлов/их моделей. Результатом RTL-симуляции является набор лог-файлов, содержащих в себе информацию о потоках транзакций в сети и поведении каждого ее узла во время симуляции. Среда

предоставляет пользователю инструменты извлечения и анализа данных файлов с целью получения информации о характеристиках сети.

За реализацию функционала среды отвечают следующие компоненты:

- Python-скрипт, отвечающий за конфигурацию параметров и расположение вычислительных узлов в сети;
- SystemVerilog - генератор подключений вычислительных узлов/их моделей;
- SystemVerilog - генератор уникальных идентификаторов транзакций;
- SystemVerilog - UVM-окружение [7, 8], отвечающее за входные воздействия, сбор информации о поведении сети и генерацию лог-файлов;
- Matlab-класс, реализующий извлечение данных из лог-файлов и предоставляющий набор методов обработки полученной информации.

Описание процесса работы среды

Взаимодействие пользователя со средой делится на 3 этапа:

- этап конфигурации;
- этап симуляции;
- этап анализа.

На этапе конфигурации пользователь передает в python-скрипт необходимые параметры. Также на этапе конфигурации пользователь передает среде информацию о необходимых для запуска симуляции исходных файлах сети и вычислительных узлов/их моделей.

На этапе симуляции средой выполняется RTL-симуляция NoC с подключенными к ней вычислительными узлами/их моделями. Результатом

данного этапа является набор лог-файлов, содержащих информацию о поведении сети в процессе симуляции.

На этапе анализа пользователю предоставляется возможность воспользоваться Matlab-классом, реализующим извлечение и анализ данных, полученных на этапе симуляции.

На рис. 1 представлена схема процесса работы среды. Прямоугольниками со скругленными краями обозначены компоненты среды. Обычными прямоугольниками обозначены файлы или же действия пользователя.

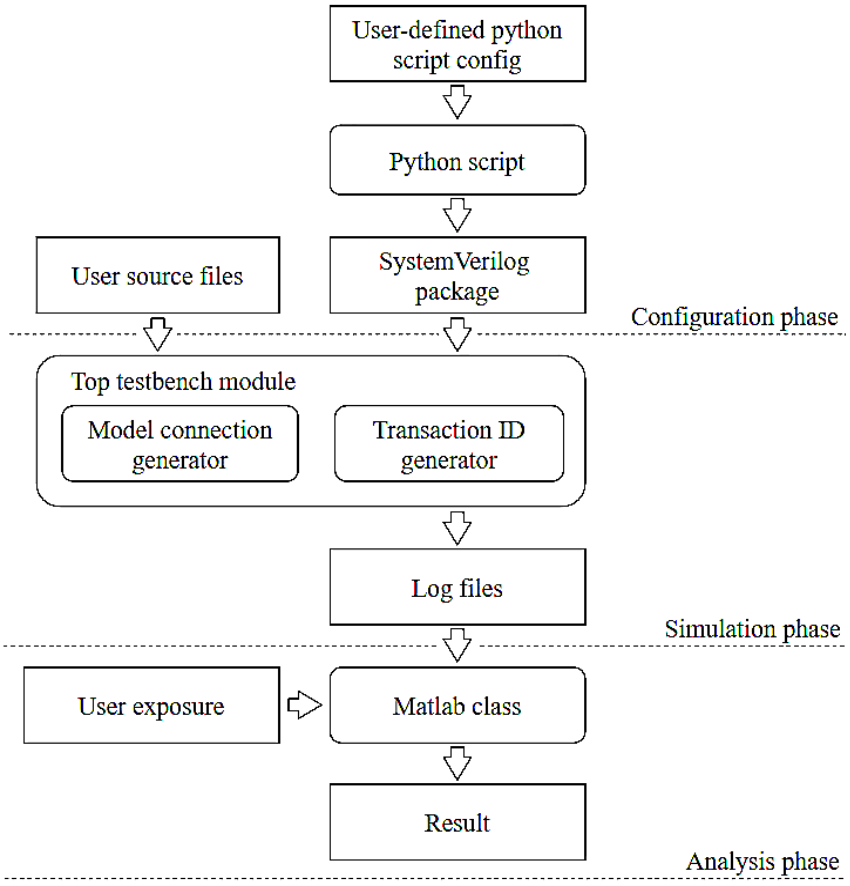

Рис.1. – Схема процесса работы среды

Структура и описание основных компонентов фазы симуляции

На рис. 2 приведена структура главного тестового модуля, являющегося основным компонентом в фазе симуляции. Прямоугольниками

со скругленными краями обозначены компоненты среды, обычными прямоугольниками обозначены пользовательские экземпляры модулей. Стрелками обозначены линии связи между компонентами/экземплярами.

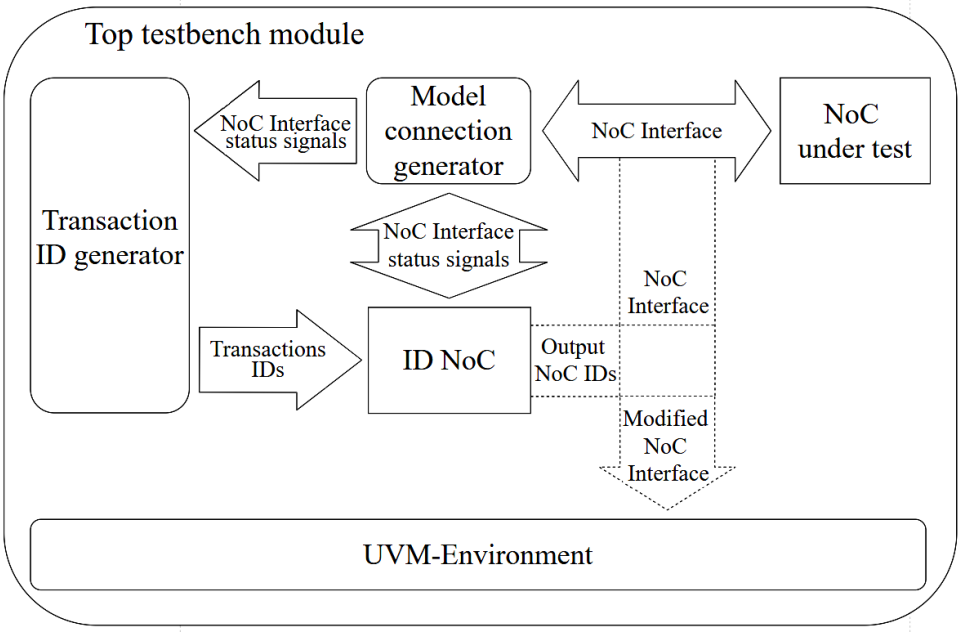

Рис.2. – Структурная схема главного тестового модуля

NoC under test – это экземпляр модуля сети, анализ поведения которой проводится средой.

ID NoC – это второй экземпляр модуля сети, использующийся для передачи уникальных идентификаторов транзакций в сети.

Model connection generator – модуль, выполняющий функцию подключения вычислительных узлов/их моделей к сети. В зависимости от параметра, определяющего положение вычислительных узлов/их моделей в сети, их интерфейсы связываются с конкретными узлами NoC.

Transaction ID generator – модуль, генерирующий уникальные идентификаторы транзакций, поступающих в сеть. Алгоритм работы зависит от статусных сигналов интерфейсов связи узлов сети и вычислительных узлов СнК.

UVM Environment – это верификационное UVM-окружение для логирования потоков данных и функциональной проверки сети. На рис.3 приведена его структура.

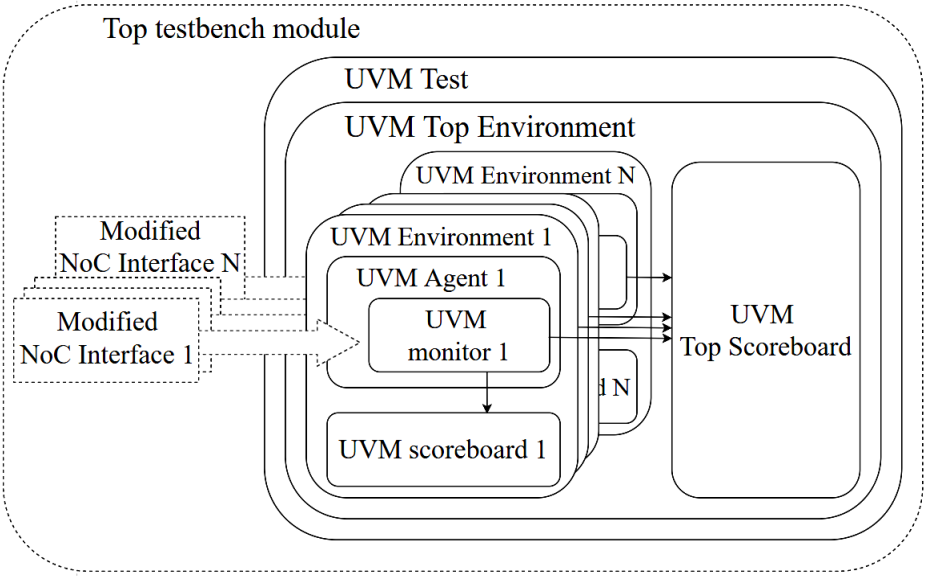

Рис.3. – Структура верификационного UVM-окружения среды

В верификационном UVM-окружении определен класс UVM Top Environment, содержащий класс UVM Top Scoreboard, необходимый для логирования потоков данных в сети, и массив классов UVM Environment. Каждый из классов массива содержит UVM Monitor и UVM Scoreboard для логирования потоков данных конкретного узла. Для каждого UVM Monitor определен свой интерфейс узла, с которым он может взаимодействовать.

Описание основных компонентов фазы анализа

В фазе анализа основным компонентом среды является Matlab-класс, реализующий извлечение информации из лог-файлов и методы ее обработки. При создании экземпляра класса в качестве аргумента конструктору передается строка, содержащая путь до лог-файла.

В классе определены методы, реализующие запросы на получение характеристик производительности NoC.

Характеристики производительности, которые могут быть получены при помощи Matlab-класса:

- максимальное/минимальное значение задержки передачи данных от узла A к узлу B (без учета задержки сети на принятие данных) в течение времени симуляции;
- максимальное/минимальное значение задержки передачи данных от узла A к узлу B (с учетом задержки сети на принятие данных) в течение времени симуляции;
- количество транзакций, переданных от узла A к узлу B в течение времени симуляции;
- количество транзакций, переданных от узла A к узлу B в течение времени симуляции, относительно их общего количества, переданного сетью за время симуляции;
- относительная пропускная способность сети для пары узлов A и B;
- получение характеристик из вышестоящих пунктов в виде матрицы, содержащей характеристики каждого узла;
- относительная пропускная способность сети.

Отслеживание и логирование потоков информации в тестируемой сети

В данном параграфе будут рассмотрены основные концепции, применяемые в среде в фазе симуляции для отслеживания и логирования потоков данных в NoC.

В верификационном окружении каждой паре узел NoC – вычислительный узел СнК выделен свой UVM Monitor, отслеживающий поток транзакций текущей пары. Отслеживая сигналы интерфейса, связывающего узел NoC и вычислительный узел, UVM Monitor собирает необходимую информацию о конкретной транзакции и отправляет данные в UVM Scoreboard.

UVM Scoreboard получают информацию о факте испускания или получения транзакции узлом. На основе полученных данных, генерируются лог-файлы потока транзакций для каждого узла. Лог-файл испущенных транзакций каждого узла содержит все транзакции, испущенные узлом (в том числе те, которые по какой либо причине не были приняты и перенаправлены сетью) в течение симуляции. Лог файл принятых транзакций содержит все транзакции, принятые узлом в течение симуляции. Создание отдельного лог-файла для каждого узла позволяет пользователю при необходимости максимально быстро получить список всех транзакций (испущенных или принятых) конкретного узла, а также убедиться в отсутствии во входном потоке некорректных транзакций (к примеру транзакций с несуществующим адресом назначения и т.п.).

UVM Top Scoreboard также получает информацию о факте испускания или получения транзакций узлами. Дополнительно этот класс получает информацию о времени инициации узлами передачи транзакций, времени попадания транзакций в сеть и времени выхода транзакций из сети (т.е. времени получения транзакций узлами). На основании полученных данных UVM Top Scoreboard формирует основной лог-файл, содержащий информацию о перемещении транзакций в сети. Формат основного лог-файла представлен на рис.4.

На рис.4 source - адрес источника данных в сети, destination - адрес приемника данных в сети, data - данные, init time - время инициации передачи транзакции, from time - время попадания транзакции в сеть, to time - время получения транзакции приемником.

source	destination	data	init time	from time	to time
[1 1]	[1 0]	c000004590c44447	1665000	1665000	1725000
[1 1]	[1 0]	c00000400d1e14ca	1555000	1555000	1735000
[1 1]	[1 1]	c0000047ba536e7c	1705000	1705000	1755000
[1 1]	[1 0]	c0000048a4e48dce	1725000	1735000	1755000
[1 0]	[1 0]	800000460c000138	1605000	1605000	1815000

Рис.4. – Формат основного лог-файла

Стоит отметить, что UVM-окружение не имеет прямого доступа ко внутренним линиям сети, т.е. не может получить информацию о текущем расположении транзакций в сети. В связи с этим, при определенных условиях может возникнуть конфликт значений полезных данных в транзакциях, что приведет к невозможности оценить временные характеристики транзакций.

Условием для возникновения конфликта значений полезных данных является нахождение в сети более одной транзакции из узла A в узел B с одинаковыми значениями полезных данных. Рассмотрим пример возникновения конфликта.

На рис.5 представлена временная диаграмма передачи транзакций между двумя вычислительными устройствами.

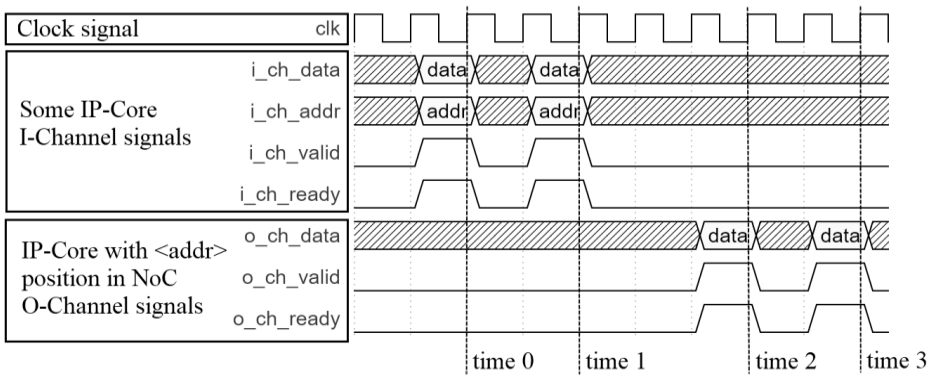

Рис.5. – Временная диаграмма условия возникновения конфликта в UVM окружении.

В момент времени time 0 и time 1 от определенного устройства в сеть попадают транзакции с одинаковыми полезными данными data. Далее в течение некоторого времени данные передаются в сети к узлу с адресом addr. В моменты времени time 2 и time 3 транзакции передаются на устройство, подключенное к узлу с адресом addr в сети. Из-за эквивалентности полезных данных в транзакциях, а также эквивалентности адреса отправителя транзакции, возникает ситуация, в которой невозможно точно определить время нахождения транзакции в сети. К примеру, для транзакции,

переданной в момент времени time 0, время пребывания транзакции в сети может быть равно time 2 - time 0 или же time 3 - time 0.

Для решения данной проблемы в среде каждой транзакции присваивается уникальный идентификатор. Для их генерации используется Transaction ID generator. Связь данного компонента с остальными компонентами среды представлена на рис. 2. Генератор подключается ко второму экземпляру NoC, который используется для передачи идентификаторов в сети (ID NoC). Введение ID NoC необходимо, так как передача уникальных идентификаторов через основной экземпляр NoC приведет к модификации полезных данных, что в свою очередь может нарушить протокол обмена данными между вычислительными узлами СнК.

В процессе симуляции поток полезных данных идет через основной экземпляр, а поток уникальных идентификаторов через дополнительный. Сигналы управления обоих экземпляров связаны с Model connection generator, что гарантирует одинаковое поведение обоих экземпляров в течение симуляции. Выходные шины данных с обоих сетей объединяются и модифицированные интерфейсы передаются в верификационное UVM-окружение, гарантируя уникальность каждой транзакции.

Интеграция сети в среду

По умолчанию в среде для связи вычислительных узлов/их моделей и узлов NoC используется интерфейс, передача данных по которому осуществляется при помощи так называемых “рукопожатий” на линиях сигналов valid и ready (см. рис. 6). Данный интерфейс используется в Hoplite NoC [9]. Если интерфейс связи пользователя отличается от интерфейса связи в среде по умолчанию, то, для интеграции своего интерфейса в среду, пользователю достаточно заменить стандартный интерфейс на пользовательский.

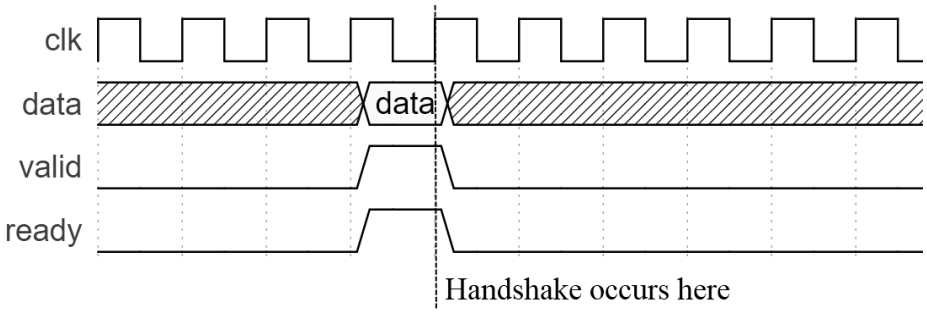

Рис.6. – “Рукопожатие”

Данный подход реализуем, так как в верификационном окружении взаимодействие с интерфейсами реализуется при помощи абстрактных классов [10]. На рис.7 представлена схема замены интерфейса по умолчанию на пользовательский. В базовом классе интерфейса определен минимальный набор методов для взаимодействия с ним. При интеграции пользователю необходимо переопределить виртуальные методы базового класса для взаимодействия с пользовательским интерфейсом.

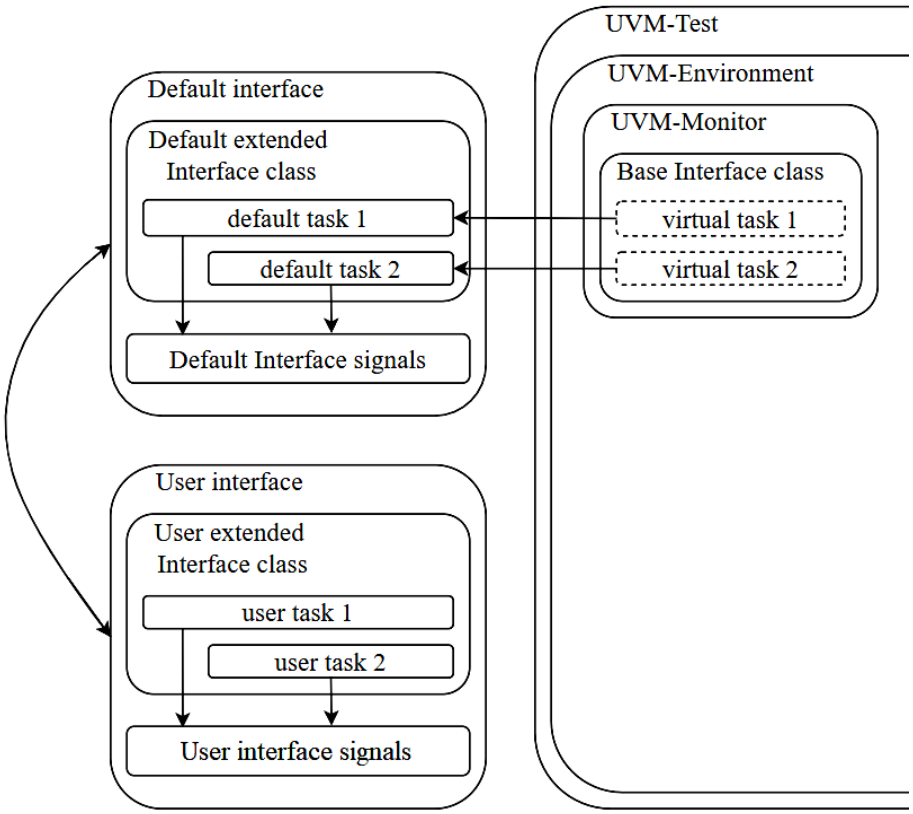

Рис.7. – Замена стандартного интерфейса пользовательским

Пример применения среды для анализа производительности NoC

В данном параграфе будет рассмотрен пример оценки производительности Hoplite NoC размерностью 4x3. В данной сети используется топология Torus, алгоритмы маршрутизации Dimension Order Routing (DOR) и Deflection Routing.

Для упрощения примера в связке с сетью будут использоваться модели вычислительных узлов, генерирующих псевдослучайный трафик. То есть, узлы с определенной вероятностью генерируют полезные данные и адрес, а после инициируют отправку, а также с определенной вероятностью принимают данные.

На рис. 8 в главном экземпляре сети выделены параметры, которые необходимо определить при подключении.

```
drt_noc #(
    .X_SIZE        ( X_SIZE        ),
    .Y_SIZE        ( Y_SIZE        ),
    .DATA_WIDTH    ( DATA_WIDTH    ),
    .X_ADDR_WIDTH  ( X_ADDR_WIDTH  ),
    .Y_ADDR_WIDTH  ( Y_ADDR_WIDTH  )
) DUT (
    .clk_i         ( /* Clock signal */ ),
    .aresetn_i     ( /* Reset signal */ ),
    .i_data_i      ( /* Connection 1 */ ),
    .i_x_addr_i    ( /* Connection 2 */ ),
    .i_y_addr_i    ( /* Connection 3 */ ),
    .i_valid_i     ( /* Connection 4 */ ),
    .o_ready_i     ( /* Connection 5 */ ),
    .i_ready_o     ( /* Connection 6 */ ),
    .o_data_o      ( /* Connection 7 */ ),
    .o_valid_o     ( /* Connection 8 */ )
);
```

Рис.8. – Параметры экземпляра HDL-дизайна сети

На этапе конфигурации данные параметры указываются в файле конфигурации Python-скрипта. Также в файле конфигурации указываются названия моделей узлов (имена модулей), которые будут подключены к сети на кристалле. Содержание файла конфигурации Python-скрипта показано на рис. 9.

```
DRT_NOC_MODELS = [
    "simple_node_model"
]

DRT_NOC_PARAMETERS = [
    "X_SIZE",
    "Y_SIZE",
    "DATA_WIDTH",
    "X_ADDR_WIDTH",
    "Y_ADDR_WIDTH"
]
```

Рис.9. – Пример файла конфигурации Python-скрипта

В процессе RTL-симуляции окружением генерируются лог-файлы, обработка которых будет производиться в фазе анализа.

На рис. 10 приведен пример Matlab-скрипта, реализующего создание экземпляра Matlab-класса и взаимодействие с ним.

```
%% Create class instance
t = TraceAnalyzer('trace_log.txt');

%% Execute methods
% Get NoC throughputs from 0 to 10000 ns
thpts = t.get_throughputs(0, 10000);

%% Methods data processing
% Display throughputs to node [0,0]
disp('Throughputs to node [0,0]:');
disp(thpts(:,:,1,1)');
```

Рис.10. – Пример использования Matlab-класса

В данном примере вызывается метод получения относительной пропускной способности для всех пар вычислительных устройств, подключенных к сети. Затем выполняется вывод матрицы относительной пропускной способности для всех пар узлов, где получателем является узел с адресом (0; 0) в сети. Результат выполнения представлен на рис. 11.

```
Throughputs to node [0,0]:
    0.5667    0.5301    0.5271    0.5731
    0.5291    0.5395    0.5505    0.5301
    0.5915    0.5595    0.5331    0.5771
```

Рис.11. – Результат использования Matlab-класса

Заключение

В данной статье была представлена тестовая среда для оценки производительности RTL-дизайна сети на кристалле в составе СнК в условиях тестирования, максимально приближенных к реальным условиям использования устройства.

Средой выполняется функциональная RTL-симуляция сети в составе СнК с использованием реального HDL-описания сети. Роль вычислительных узлов СнК выполняют HDL-модули с синтезируемым дизайном или же их несинтезируемые модели.

Взаимодействие со средой реализовано при помощи Python-скрипта, отвечающего за определение необходимых параметров и положений вычислительных узлов в сети, а также Matlab-класса, отвечающего за извлечение и анализ данных лог-файлов, генерируемых UVM-окружением среды в процессе симуляции.

Работа выполнена при финансовой поддержке Фонда содействия инновациям (договор № 39ГУРЭС14/72784 от 26.12.2021).

Литература (References)

1. Jones Andrew, Ryan Stuart Enabling High Performance SoCs Through Multi-Die Re-use. URL: design-reuse.com/articles/29013/multi-die-re-use.html (Date accessed: 09.11.22).
2. Dally W., Towles B. Principles and Practices of Interconnection Networks. San Francisco, Morgan Kaufmann Publ., 2003, pp. 1-2.
3. Umit Y. Ogras, Radu Marculescu Modeling, Analysis and Optimization of Network-on-Chip Communication Architectures. New York, Springer, 2013, pp. 1-3.
4. Jiang N, Becker DU, Michelogiannakis G, Balfour J, Towles B, Shaw DE, Kim J, Dally WJ. A detailed and flexible cycle-accurate Network-on-Chip

simulator // IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 21-23 April 2013, pp. 86-96.

5. Catania, V., Mineo, A., Monteleone, S., Palesi, M. and Patti, D.,. Noxim: An open, extensible and cycle-accurate network on chip simulator // 2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 21-23 April 2015, pp. 162-163.
6. Van Laer, A., Jones, T. and Watts, P.M.. Full system simulation of optically interconnected chip multiprocessors using gem5 // 2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 17-21 March 2013, pp. OTh1A-2.
7. Universal Verification Methodology (UVM) 1.2 User's Guide. URL: accellera.org/images/downloads/standards/uvm/uvm_users_guide_1.2.pdf (Date accessed: 16.11.22).
8. Standard Universal Verification Methodology. URL: accellera.org/downloads/standards/uvm (Date accessed: 16.11.22).
9. Kapre N, Gray J.. Hoplite: Building Austere Overlay NoCs. In 2015 25th international conference on field programmable logic and applications (FPL) 2015 Sep 2. Pp. 1-8.
10. How to Access a Parameterized SystemVerilog Interface from UVM. URL: doulos.com/knowhow/systemverilog/uvm/easier-uvm/easier-uvm-deeper-explanations/parameterized-interface-example/ (Date accessed: 09.11.22).