УДК 004.41 DOI: 10.31114/2078-7707-2020-4-45-50

Адаптация RTL-описания системы на кристалле для распределенной системы эмуляции

А.Н. Петров[1], С.В. Юрлин[1,2]

[1]АО «МЦСТ», [2]ПАО «ИНЭУМ им И.С. Брука»,

turingmachinerepair@yandex.ru

***Аннотация* — Рассмотрена задача адаптации RTL-описания системы на кристалле для распределенной системы эмуляции на ПЛИС. Обозначены основные проблемы процесса перехода к функциональной верификации системы на кристалле на базе системы аппаратной эмуляции на основе ПЛИС для разработчика микросхемы и предложен метод их решения на примере системы КУБ-ПРО, применяемой в АО «МЦСТ».**

***Ключевые слова* — эмуляция, ПЛИС, микропроцессор, системы эмуляции.**

I. Введение

Прототипы на основе ПЛИС уже десятилетиями применяются для аппаратной верификации систем на кристалле. Эксплуатация данных устройств зачастую даёт более реалистичные результаты с большей производительностью, нежели программная симуляция [1]. Такие прототипы являются специализированными изделиями, применение которых за пределами верификации отдельного устройства ограничено, что делает эксплуатацию таких прототипов экономически невыгодной, так как с ростом сложности растет количество нужных ПЛИС, а недостаточная гибкость конфигурации для новых проектов заставляет каждый раз заново делать новые устройства.

С целью снижения затрат на проведение функциональной верификации гораздо эффективнее применять аппаратные системы эмуляции на основе ПЛИС. Системы аппаратной эмуляции состоят из множества ПЛИС, соединенных между собой в соответствии со структурой устройства СнК. Данные системы можно конфигурировать в зависимости от верифицируемой микросхемы и масштабировать без принципиальных ограничений. Такой подход позволяет расширить спектр применения системы верификации, однако ставит перед разработчиками дополнительную задачу – необходимость адаптировать описание системы на кристалле для системы аппаратной эмуляции. В задачу адаптации входят:

1) Разделение модели СнК на блоки и их расположение на нескольких ПЛИС.
2) Конфигурация соединений между ПЛИС в соответствии со связями между блоками логики.
3) Создание встраиваемого ПО для ПЛИС.

Сложность и трудоемкость данной задачи растет с ростом количества элементов в системе эмуляции и фактически для СнК свыше 100 тысяч стандартных ячеек ручное разделение неэффективно [2].

В АО «МЦСТ» уже на протяжении 12 лет применяются прототипы на основе ПЛИС для верификации микропроцессоров, последние 2 года из которых - на базе системы аппаратной эмуляции КУБ-ПРО [3]. В данной системе исходное RTL-описание (Register Transfer Level) микропроцессора разбивается на блоки логики, соединенные между собой логическими связями. Каждый блок логики размещается на одной ПЛИС. Каждая микросхема располагается на модуле, который позволяет подключить до 8 других аналогичных модулей посредством интерфейсов двух типов.

Для решения вышеописанных задач в рамках эксплуатации распределенной системы эмуляции «КУБ-ПРО» в АО «МЦСТ» был разработан метод адаптации описания системы на кристалле для системы аппаратной эмуляции. В соответствии с этим методом была создана САПР, которая выполняет данные действия в автоматическом режиме.

В данной статье рассматривается метод адаптации описания микропроцессора для распределённой системы эмуляции на основе ПЛИС.

II. Постановка задачи

К моменту начала эксплуатации системы эмуляции описание СнК должно пройти модульное и интеграционное тестирование. Это позволяет точно оценить потребляемые ресурсы отдельными модулями СнК.

Для проведения функциональной верификации как следующего этапа тестирования необходимо это описание разместить в ПЛИС системы эмуляции. Соответственно между окончанием автономной верификации и началом функциональной появляется задача: адаптировать исходное описание системы на кристалле для эксплуатации на системе эмуляции и создать встроенное программное обеспечение для вычислительных узлов. Прежде всего определим входные данные для задачи адаптации.

Входными данными являются:

1) RTL-описание исходной системы на кристалле.

2) Набор файлов настройки, определяющих зависимые от программной среды параметры и задающие разделение исходной СнК.

3) Библиотека элементов применяемой системы эмуляции.

4) Базовые проекты ПЛИС применяемой системы эмуляции.

Результатами решения задач адаптации являются:

1) Адаптированные блоки СнК – блоки логики, на которые разделяется RTL-описание исходной системы на кристалле.

2) Модель распределенной системы эмуляции.

3) Результирующие проекты ПЛИС используемой системы эмуляции для последующей компиляции и получения встроенного ПО.

При реализации метода адаптации для решения этой задачи необходимо из исходного RTL-описания микросхемы путем последовательных преобразований распределить модули СнК на ПЛИС без нарушения функциональной целостности, соединить их между собой посредством доступных на модулях системы эмуляции интерфейсов и создать проекты ПЛИС для получения встроенного ПО.

III. Маршрут работы

Для решения описанной ранее задачи был разработан метод адаптации, позволяющий осуществить необходимое разделение.

Важной особенностью метода является тот факт, что само разделение задает разработчик СнК, а реализация метода формирует встраиваемое ПО. Автоматизация вывода разделения сводится к задаче кластеризации графа [4]. Имеет смысл её реализовывать в том случае, когда в процессе разработки микросхемы процедуру нужно выполнять много раз. Однако в случае наличия формально верифицированного описания микропроцессора изменение наполнения логического описания происходит крайне редко. Иначе говоря, стратегия разделения задается разработчиком и в случае формально верифицированного описания она меняется редко. Если необходимо повторное разделение, например в случае изменения интерфейса между блоками логики, и нужно снова сгенерировать ПО, то затраты на повторное решение задачи разделения (а также её реализацию и отладку) превышают затраты на ручное определение разделения.

В рамках метода предусмотрено последовательное выполнение пяти этапов:

1) Настройка среды.
2) Разделение и сборка.
3) Создание модели.
4) Пост-обработка.
5) Генерация проектов.

Полный маршрут представлен на рис. 1.

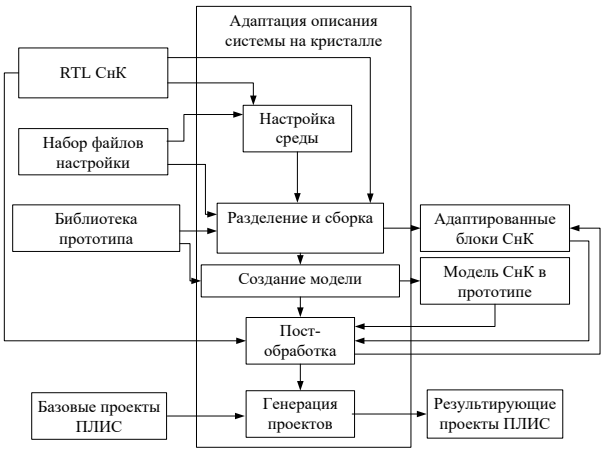

Рис. 1. Маршрут метода адаптации

A. *Настройка среды*

Первый этап выполняет подготовку исходного описания к процессу разделения. На нём создаётся копия части описания микросхемы, которая будет участвовать в разделении. Этап выполняет следующие действия:

1) Установка переменных среды (производится в соответствии со списком значений, предоставляемым в конфигурационных файлах).

2) Проверка иерархии директорий по сравнению с заданной в конфигурационном файле. При отсутствии нужных директорий процесс их создаёт.

3) Проверка корректности конфигурационных файлов для разделения – в том числе списков наполнения микросхем конфигурируемой логики.

4) Копирование необходимых файлов проекта. Создаются копии описания исходной микросхемы, при этом определённые модули очищаются от внутренней логики. Данные модули ограничивают глубину вложенности иерархии исходного описания и задают совокупность сущностей, подвергающихся группировке. Копируемые модули указываются в конфигурационных файлах в списках модулей с логическим наполнением и без него.

В результате выполнения данного этапа в операционной системе ЭВМ заданы нужные переменные среды, создана или исправлена иерархия директорий и подготовлена локальная копия описания системы на кристалле с учетом требований процесса разделения.

B. *Разделение и сборка*

Второй этап выполняет разделение исходного описания микросхемы согласно конфигурации и создаёт связанные между собой блоки логики и модули верхнего уровня для ПЛИС системы эмуляции.

Входными данными являются: описание исходной микросхемы, библиотека системы эмуляции и конфигурационные файлы. Пример структуры абстрактной СнК приведен на рис. 2.

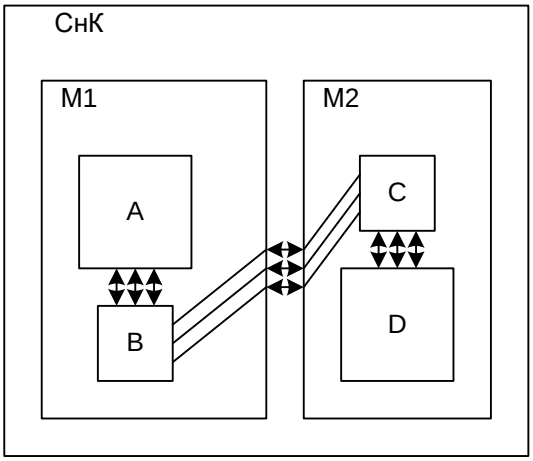

Рис. 2. Исходная структура абстрактной СнК

Этап разделяется на три последовательные части: выравнивание иерархии исходной СнК, группировка модулей СнК в блоки логики, создание файлов языка описания аппаратуры, описывающих модули верхнего уровня для каждой ПЛИС. Данные процессы происходят в рамках компилятора языка описания аппаратуры.

Выравнивание иерархии расформировывает логические модули до тех пор, пока модули на верхнем уровне не имеют вложенной логики. Пример структуры СнК после выравнивания приведен на рис. 3.

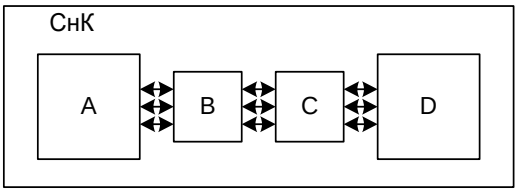

Рис. 3. Система на кристалле после расформировывания

Процесс группировки происходит следующим образом:

1) создаются модули логики, соответствующие каждой группе;

2) в модулях размещаются необходимые модули исходной иерархии;

3) создаются порты модулей, соответствующие связям между группами;

4) модули групп соединяются между собой согласно связям между модулями внутри групп.

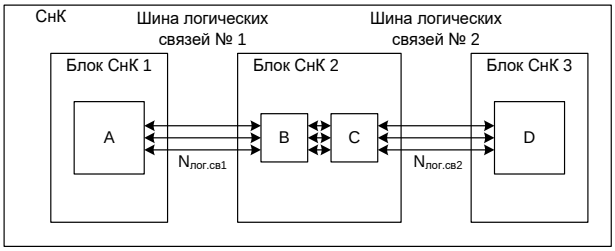

Рис. 4. Система на кристалле, разделенная на блоки логики

В результате верхний уровень микросхемы состоит из блоков логики, связанных между собой и содержащих модули исходного описания. Данный процесс вплоть до последнего шага можно выполнять параллельно для каждой группы. Пример структуры СнК после группировки приведен на рис. 4.

Процесс создания модулей верхнего уровня ПЛИС происходит следующим образом:

1) создаются пустые модули верхнего уровня ПЛИС из модулей верхнего уровня соответствующих базовых проектов;

2) в каждый модуль размещается соответствующий модуль логики;

3) модуль логики подключается к интерфейсам межмодульной связи в соответствии с конфигурацией, полученной в результате процесса разделения. Пример структуры СнК после подключения интерфейсов приведен на рис. 5;

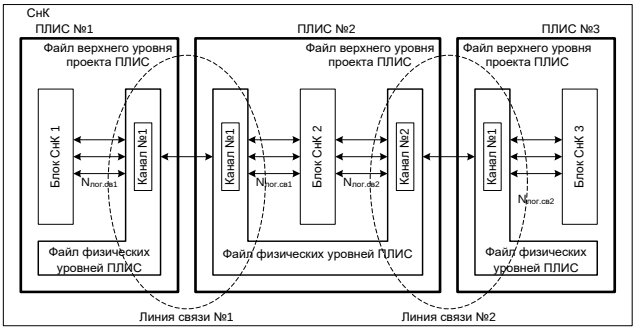

Рис. 5. Система на кристалле, размещенная в системе эмуляции

4) модуль логики подключается к выводам модуля верхнего уровня и другим модулям на том же уровне иерархии.

Данный процесс может выполняться параллельно для каждого отдельного устройства системы. Результатом действия является RTL-описание модулей логики и модулей верхнего уровня ПЛИС. Пример структуры СнК после этого действия приведен на рис. 6.

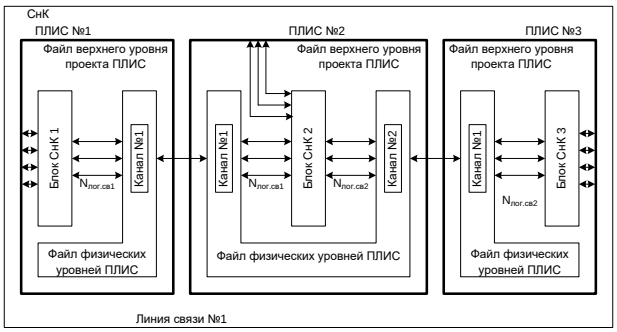

Рис. 6. СнК в системе эмуляции после этапа разделения и сборки

C. Создание модели

Третий этап собирает RTL-представление сконфигурированной системы аппаратной эмуляции.

Путем симуляции данной модели можно убедиться, что прототип функционально эквивалентен верифицируемому устройству.

Алгоритм данного этапа представлен на рис. 7.

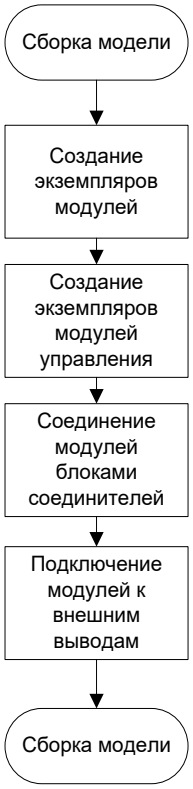

Рис. 7. Процесс сборки модели прототипа

Данный этап дополнительно создает модели IP-блоков для дальнейшей симуляции в Synopsys VCS. Полученные модели учитывают физические параметры IP-блоков и позволят максимально реально симулировать физические уровни интерфейсов, однако для этого требуются большое количество ресурсов.

Для того чтобы модель можно было собрать и запустить в симуляторе, необходимо составить конфигурационный файл, содержащий в себе следующие данные:

1) Содержимое конфигурационного файла тестового окружения системы эмуляции.
2) Список путей к заменяемым модулям.
3) Содержимое основного конфигурационного файла тестового окружения верифицируемой СнК.
4) Список путей к моделям модулей симуляции и модулю модели прототипа.
5) Явное указание модуля верхнего уровня.

D. Пост-обработка

Этап пост-обработки приводит полученные файлы к удобному для восприятия разработчиком виду и для редактирования характеристик распределённой системы эмуляции. На данном этапе могут выполняться следующие процессы: параметризация объявлений модулей, расстановка отступов и переносов строк, проведение эквивалентных замен синтаксических конструкций для поддержки более старых или новых стандартов применяемого языка описания аппаратуры.

С точки зрения САПР параметры не изменяются в рамках одного дизайна, поэтому в процессе создания внутреннего представления модуля параметры добавляются в название типа модуля, что отражается и в выводе САПР. Например, если у нас имелся экземпляр модуля со следующим объявлением:

```
test_module #( .PAR(5) ) inst(),
```

то в Verilog-модуле, сгенерированным компилятором Design Compiler фирмы Synopsys, этот же модуль инстанциируется в следующем виде:

```
test_module_PAR5 inst().
```

Разумеется, при компиляции проекта с данным модулем будет выдана ошибка, так как модуля с таким названием не существует. Как следствие, необходимо преобразовать данные модули к исходному виду.

Данная операция происходит путем поиска экземпляров модулей и обработки с помощью соответствующего регулярного выражения.

С точки зрения САПР интерфейс является набором сигналов и внутреннее представление преобразует модульные порты в обычные, название которых состоит из конкатенации названия модульного порта, символа точки и названии сигнала. При попытке компиляции проекта с таким инстанциированием модуля САПР фирмы Xilinx и Intel выдадут ошибку. Следовательно, необходимо восстановление модульных портов и корректное их переподключение. Блок-схема алгоритма восстановления интерфейсов представлена на рис. 8.

Результатом этапа является RTL-описание модулей, подвергнувшихся преобразованиям.

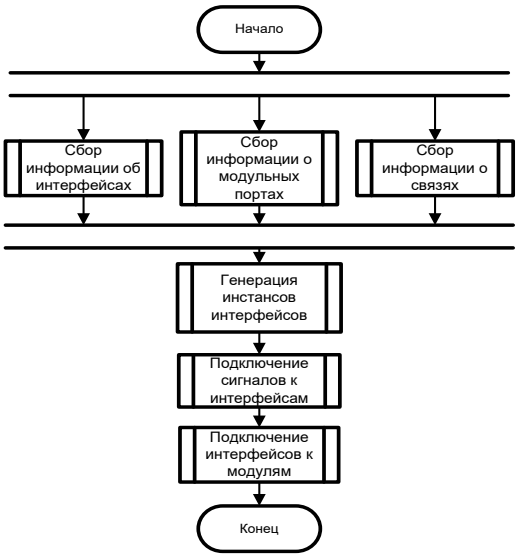

Рис. 8. Алгоритм восстановления интерфейсов

E. Генерация проектов

Данный этап создает результирующие проекты ПЛИС на основе предоставленных базовых.

Базовый проект – проект для микросхемы ПЛИС FPGA, который содержит описание всех интерфейсов, подключенных к микросхеме на модуле эмуляции. Введение базовых проектов позволяет проще автоматизировать процесс назначения выводов для тестируемых интерфейсов.

Базовые проекты содержат в себе шаблоны модуля верхнего уровня с модулями интерфейсов связи. Результирующие проекты адаптированы для системы эмуляции посредством ряда модификаций базового проекта, в которые могут входить: замена IP-блоков сторонних производителей на IP-блоки применяемой ПЛИС, явное применение специальных блоков памяти вместо регистровых файлов, замена блоков фазовой автоподстройки частоты. Модификация базовых проектов необходима, так как сторонние IP-блоки могут не поддерживаться целевой ПЛИС. Помимо этого данные замены зачастую выгодны в целях увеличения производительности целевой платформы. Генерацию проектов можно проводить параллельно для всех проектов САПР в проекте разделения, так как они независимы друг от друга и не содержат общей информации.

Под проектом в данном случае подразумевается совокупность файлов настройки, ограничений и файл логического анализатора.

Результирующий проект формируется следующей последовательностью действий:

1) Копируются файлы проекта.
2) В файле .qsf:
   a. заменяется название модуля верхнего уровня базового проекта на модуль верхнего соответствующего результирующего проекта;
   b. в конец файла добавляются преобразованные в формат qsf пути до директорий и файлов из конфигурационного файла симуляции СнК;
   c. добавляются макросы компиляции из конфигурационного файла симуляции СнК.
3) В файле .stp заменяется название модуля верхнего уровня.

Результатом данного этапа являются результирующие проекты Quartus, которые можно запускать для компиляции.

IV. Заключение

В статье рассмотрена основная проблема процесса перехода к функциональной верификации системы на кристалле на базе системы аппаратной эмуляции. Это переход от описания системы на кристалле к описанию вычислительных узлов системы, на базе которой происходит функциональная верификация. Данная проблема ставит перед разработчиком задачу адаптации исходного описания.

Рассмотрена задача адаптации RTL-описания системы на кристалле для распределенной системы эмуляции на ПЛИС. В качестве решения задачи представлен универсальный метод адаптации, применяемый АО «МЦСТ» для работы с системой аппаратной эмуляции «КУБ-ПРО».

В данном методе выполняются пять этапов: настройка среды, разделение и сборка, создание модели, пост-обработка и генерация проектов.

Настройка среды подготавливает программную среду, в которой будет исполняться реализация метода. Также на этом этапе происходит ограничение глубины иерархии микросхемы, так как разделение редко требует большой глубины.

Разделение и сборка – основной этап, который расформировывает иерархию микросхемы до максимальной глубины, ограниченной в предыдущем этапе, группирует модули в связанные между собой блоки логики, размещает их в модулях верхнего уровня применяемых ПЛИС и подключает в данных модулях межмодульные связи и периферийные интерфейсы. В результате пользователь получает адаптированное описание СнК и модули верхнего уровня ПЛИС, используемых в системе эмуляции.

Этап создания модели формирует модель системы аппаратной эмуляции путем сборки последней из библиотеки моделей аппаратуры согласно заданной пользователем конфигурации и результатом предыдущего этапа. В результате пользователь получает полноценную модель адаптированной СнК и может провести моделирование.

Этап пост-обработки приводит сгенерированные файлы в удобный для восприятия вид. Это необходимо для пользователей, которые планируют в дальнейшем дорабатывать адаптированное представление в процессе верификации.

Этап генерации проектов создаёт проекты САПР используемых ПЛИС для последующей компиляции и получения встраиваемого ПО.

Реализация данного метода позволила достичь ряда положительных результатов:

1) исключен человеческий фактор из процесса создания встраиваемого ПО для модулей системы эмуляции;

2) ускорен процесс адаптации даже для небольшого количества ПЛИС;

3) расширена универсальность – метод позволяет выполнять адаптацию любого описания СнК для любой платформы любой конфигурации;

4) простая конфигурация позволяет адаптировать СнК разработчикам с минимальным знанием особенностей платформы эмуляции.

Литература

[1] Юрлин С.В., Воробьев А.С. Масштабируемая система эмлуяции сложных вычислительных систем // mcst.ru. 2018. URL: http://www.mcst.ru/files/5c276c/340cd8/509238/000004/yurlin_s.v_vorobev_a.s_masshtabiruemaya_sistema_emulyatsii_slozhnyh_vychislitelnyh_sistem.pdf (дата обращения: 01.04.2020).

[2] Vaibbhav T. Logic Synthesis and SoC Prototyping. Пуна, Махараштра, Индия, Springer Nature Singapore Pte Ltd., 2020. 251 с.

[3] Юрлин С. В. Универсальный подход к построению масштабируемых прототипов многоядерных микропроцессоров (КУБ-ПРО) // Вопросы радиоэлектроники. 2018. № 2. С. 93-98.

[4] Gupta N., Garg D., Gupta S. Genetic Algorithms based Partitioning of VLSI Circuit Systems // ijcaonline.org - International Journal of Computer Applications. 2012. URL: https://research.ijcaonline.org/ncfaaiia/number2/ncfaaiia1013.pdf (дата обращения: 02.04.2020).

Adaptation Process of System-on-a-Chip RTL-Description for Distributed Emulation System

A.N. Petrov[1], S.V. Yurlin[1,2]

[1]JCT «MCST», Moscow

[2]PJSC «Institute of Electronic Control Computers named after I.S. Bruk»,
turingmachinerepair@yandex.ru

***Abstract* — Multi-FPGA distributed emulation systems are commonly used for functional verification of System-on-a-Chip. However, using such systems poses a problem of adaptation RTL-description for use in multi-FPGA system. This article outlines common issues for transition process towards functional verification of System-on-a-Chip using multi-FPGA distributed emulation system from the developer point of view. Article also introduces a method of adaptation, currently used in JCT "MCST". Adaptation process presents following problems: distributing SoC description across multiple FPGAs, configuring inter-FPGA connections according to RTL logic, generating FPGA firmware. Proposed method solves these problems. It consists of five stages. The first stage: preparation - setting up runtime for implementation and limited flattening of SoC hierarchy. The second stage - partition and assembly – creating interconnected logic modules containing original RTL, FPGA wrappers and top-level modules. Model generation – generating RTL model of distributed emulation system. Post-processing – correcting of CAD artifacts and making generated RTL more human-readable for further modification. FPGA project generation – creating FPGA projects from project templates for further firmware generation.**

***Keywords* — emulation, FPGA, microprocessor, emulation systems.**

References

[1] Yurlin S.V., Vorobiev A.S. Scalable emulation system for complex computing systems. Available at: http://www.mcst.ru/files/5c276c/340cd8/509238/000004/yurlin_s.v_vorobev_a.s_masshtabiruemaya_sistema_emulyatsii_slozhnyh_vychislitelnyh_sistem.pdf (accessed 01.04.2020). (In Russian).

[2] Vaibbhav T. Logic Synthesis and SoC Prototyping, Pune, Maharashtra, India, Springer Nature Singapore Pte Ltd., 2020. P. 203-205. 251p.

[3] Yurlin S.V. A Universal Approach to Building Scalable Prototypes of Multi-Core Microprocessors (KUB-PRO). Available at: http://www.mcst.ru/files/5a9eba/800cd8/50f31d/000000/yurlin_s._v._universalnyy_podhod_k_postroeniyu_masshtabiruemyh_prototipov_mnogoyadernyh.pdf (accessed 02.04.2020). (In Russian).

[4] Gupta N., Garg D., Gupta S. Genetic Algorithms based Partitioning of VLSI Circuit Systems. Available at: https://research.ijcaonline.org/ncfaaiia/number2/ncfaaiia1013.pdf (accessed 02.04.2020).