DOI: 10.15514/ISPRAS-2026-38(3)-3

Статический анализ модели подсистемы питания цифровой аппаратуры

[1,2] А.В. Егиазарян, ORCID: 0009-0003-2563-1669 <yeghiazaryan.arman@ispras.ru>
[1] Я.А. Чуркин, ORCID: 0009-0000-5044-4249 <yan@ispras.ru>
[1,3] И.И. Чернявских, ORCID: 0009-0004-8414-9079 <chernyavskih@ispras.ru>
[1] А.М. Коцыняк, ORCID: 0000-0003-3499-4368 <kotsynyak@ispras.ru>
[2] К.Н. Китаев, ORCID: 0009-0004-9469-8100 <kitaev.kn@phystech.edu>
[1] Р.А. Бучацкий, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
[1,2,4,5] А.С. Камкин, ORCID: 0000-0001-6374-8575 <kamkin@ispras.ru>
[6] А.В. Коршунов, ORCID: 0000-0002-2470-5836 <a.korshunov@istc-miet.ru>
[6] А.Л. Переверзев, ORCID: 0000-0002-5834-5138 <a.pereverzev@istc-miet.ru>

[1] Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

[2] Московский физико-технический институт,
141701, Московская область, г. Долгопрудный, Институтский переулок, д. 9.

[3] Национальный исследовательский университет «Высшая школа экономики»
Россия, 101000, г. Москва, ул. Мясницкая, 20.

[4] Московский государственный университет имени М.В. Ломоносова,
Россия, 119991, Москва, Ленинские горы, д. 1.

[5] Российский экономический университет имени Г.В. Плеханова,
117997, Россия, г. Москва, Стремянный переулок, д. 36.

[6] Акционерное общество «Международный научно-технологический центр МИЭТ»
Россия, 124527, Москва, Зеленоград, Солнечная аллея, д. 10, стр. 1.

Аннотация. Традиционные языки описания аппаратуры Verilog/SystemVerilog и VHDL не предоставляют средства для эффективного описания структуры и режимов работы подсистемы питания. Для решения этой проблемы в стандарте IEEE 1801 был представлен формат Unified Power Format (UPF), позволяющий формализовать структуру и правила управления питанием в цифровых системах. Однако, большинство современных САПР, поддерживающих интерпретацию UPF-описаний, являются коммерческими и нередко имеют отклонения от стандарта. При этом некоммерческие инструменты значительно отстают от коммерческих аналогов в степени поддержки конструкций формата и редко включают средства анализа корректности описания подсистемы питания. В работе представлен инструмент для интерпретации UPF-описания и статического анализа модели подсистемы питания, который обеспечивает полную поддержку стандарта IEEE 1801-2018.

Ключевые слова: подсистема питания; формат описания подсистемы питания UPF; командный язык инструментов Tcl; статический анализ; язык описания аппаратуры; язык описания аппаратуры Verilog; язык описания аппаратуры SystemVerilog; интегральная схема; система автоматизации проектирования (САПР); абстрактное синтаксическое дерево (АСД); система статического анализа языка описания аппаратуры SVAN.

Для цитирования: Егиазарян А.В., Чуркин Я.А., Чернявских И.И., Коцыняк А.М., Китаев К.Н., Бучацкий Р.А., Камкин А.С., Коршунов А.В., Переверзев А.Л. Статический анализ модели подсистемы питания цифровой аппаратуры. Труды ИСП РАН, том 38, вып. 3, часть 1, 2026 г., стр. 45–70. DOI: 10.15514/ISPRAS–2026–38(3)–3.

Благодарности: Проект поддерживается Минпромторгом России в рамках ОКР «Разработка системы статического анализа для языка описания аппаратуры», шифр «САПР-Анализ» (в составе ОКР «САПР микроэлектроника», головной исполнитель – АО «МНТЦ МИЭТ»).

Static analysis of power intent in integrated circuits

[1,2] A.V. Yeghiazaryan, ORCID: 0009-0003-2563-1669 <yeghiazaryan.arman@ispras.ru>
[1] Y.A. Churkin, ORCID: 0009-0000-5044-4249 < yan@ispras.ru>
[1,3] I.I. Chernyavskih, ORCID: 0009-0004-8414-9079 <chernyavskih@ispras.ru>
[1] A.M. Kotsynyak, ORCID: 0000-0003-3499-4368 <kotsynyak@ispras.ru>
[2] K.N. Kitaev, ORCID: 0009-0004-9469-8100 <kitaev.kn@phystech.edu>
[1] R.A. Buchatskiy, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
[1,2,4,5] A.S. Kamkin, ORCID: 0000-0001-6374-8575 <kamkin@ispras.ru>
[6] A.V. Korshunov, ORCID: 0000-0002-2470-5836 <a.korshunov@istc-miet.ru>
[6] A.L. Pereverzev, ORCID: 0000-0002-5834-5138 <a.pereverzev@istc-miet.ru>

[1] Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] Moscow Institute of Physics and Technology,
9, Institutsky Lane, Dolgoprudny, Moscow Region, 141701, Russia.
[3] HSE University
20 Myasnitskaya Ulitsa, Moscow 101000, Russia
[4] Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
[5] Plekhanov Russian University of Economics,
36 Stremyanny lane, Moscow, 117997, Russia.
[6] JSC "International Scientific and Technological Center of MIET"
10, Solnechnaya Alley, Moscow, 124498, Russia.

Abstract. With the increasing number of electrical components and integrated circuits' physical size reduction, organizing the energy-efficient operation of digital hardware has become significantly more complex. Traditional hardware description languages Verilog/SystemVerilog and VHDL lack the means for effectively defining and optimizing the description of the power intent. To address this issue, the IEEE 1801 standard introduced the Unified Power Format (UPF), which allows for the formalization of the power management structure and rules in digital systems. However, most modern EDA tools that support the interpretation of UPF descriptions are commercial and often exhibit deviations from the standard, whereas non-commercial tools significantly lag behind their commercial counterparts in terms of language command support and rarely include power intent analysis capabilities. This paper presents a tool for interpreting UPF descriptions and performing static analysis of the power intent model that fully supports the IEEE 1801-2018 standard.

Keywords: power intent; Unified Power Format; UPF; Tcl; static analysis; hardware description language; Verilog; SystemVerilog; integrated circuit; EDA; AST; SVAN.

For citation: Yeghiazaryan A.V., Churkin Y.A., Chernyavskih I.I., Kotsynyak A.M., Kitaev K.N., Buchatskiy R.A., Kamkin A.S., Korshunov A.V., Pereverzev A.L. Static analysis of power intent in integrated circuits. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 3, part 1, 2026, pp. 45-70 (in Russian). DOI: 10.15514/ISPRAS-2026-38(3)-3.

Acknowledgements. The project is supported by the Ministry of Industry and Trade of the Russian Federation within the R&D project "Development of a static analysis system for a hardware description language", code "SAPR-Analiz" (as part of the R&D project "SAPR mikroelektronika", the main contractor is JSC "ISTC MIET").

1. Введение

Проектирование цифровых интегральных схем – это последовательный процесс из нескольких этапов, начиная от спецификации и разработки модели на языке описания аппаратуры (HDL, Hardware Description Language), до логического и физического синтеза микросхемы. Построение модели регистровых передач (RTL, Register Transfer Level) является первым этапом на этом пути. При этом в настоящее время одной из ключевых проблем стало не только увеличение скорости работы и уменьшение занимаемого пространства на кристалле, но и уменьшение энергопотребления.

Для повышения энергоэффективности были разработаны такие методы снижения энергопотребления, как управление (стробирование) цепями синхросигнала, локальное отключение компонентов, реструктуризация логики и другие [1-3]. Но со временем уменьшение технологических норм и одновременное увеличение числа логических вентилей в схемах усложнило применение методов энергоэффективной разработки, организацию системы отвода тепла, а также привело к росту токов утечки [4]. Все эти факторы привели к значительным сложностям при разработке энергоэффективных решений. При этом множество современных цифровых устройств получают энергию от аккумулятора, что также повышает требования к эффективности работы микросхемы.

Для решения этой проблемы были разработаны методы энергосбережения, основанные на разделении интегральной схемы на отдельные части или домены. Такое разделение позволяет для каждого из доменов определить различные параметры распределения питания. Совокупность описаний питания для всех доменов представляет собой описание подсистемы питания (power intent). Использование описания подсистемы питания с доменами дает возможность реализовать такие методы энергосбережения, как:

- использование отключаемых доменов питания;
- динамическое изменение уровня напряжения питания домена или его отдельных частей;
- распределение памяти на блоки, расположенные в разных доменах питания, с возможностью отключать или переводить отдельные блоки в режим сохранения состояния.

Доменный подход разработки распределения питания в схеме дает возможность значительно уменьшить причины рассеяния энергии, вплоть до уменьшения токов утечки в 50 раз [4]. Однако языки описания аппаратуры Verilog/SystemVerilog и VHDL не предоставляют возможностей для описания подсистемы питания, так как изначально не предусмотрены для решения этой задачи.

Для этого был разработан формат описания подсистемы питания UPF (Unified Power Format). Он дает возможность внедрить методы снижения энергопотребления в самом начале разработки на уровне RTL-описания.

Впервые UPF 1.0 [5] был представлен организацией Accellera Systems Initiative в 2007 г. С 2009 года организация IEEE начала работу над дополнением и улучшением стандарта [6-9]. На данный момент последней является версия UPF 4.0 (IEEE 1801-2024) [10].

Подсистема питания внедряется в процесс проектирования интегральной схемы вместе с RTL-описанием, а это значит, что поиск и исправление ошибок на поздних этапах разработки требует значительных временных и экономических затрат. При этом, возможные ошибки в описании подсистемы питания могут привести к ухудшению производительности, некорректной работе схемы или выходу устройства из строя. В связи с этим важным этапом разработки становится анализ описания подсистемы питания.

Существует ограниченное подмножество команд UPF, необходимое и достаточное для описания подсистемы питания. Это позволяет разрабатывать инструменты, ориентированные на поддержку лишь данного подмножества без реализации полной поддержки стандарта. В

результате большинство современных САПР (систем автоматизации проектирования) обеспечивают лишь частичную поддержку UPF. Коммерческие проприетарные инструменты, разрабатываемые компаниями Cadence, Synopsys и Siemens, позволяют обрабатывать и анализировать UPF-описания, однако степень соответствия стандарту существенно различается как по набору поддерживаемых команд, так и по особенностям их синтаксической интерпретации. Это приводит к необходимости адаптации UPF-описаний под конкретный инструмент, что снижает переносимость и воспроизводимость проектов. Кроме того, получение лицензий на использование зарубежных коммерческих решений в настоящее время затруднено. В то же время инструменты с открытым исходным кодом существенно уступают коммерческим аналогам по уровню поддержки UPF и, как правило, не предоставляют средств анализа подсистемы питания. В совокупности указанные факторы обуславливают необходимость разработки собственного инструмента, обеспечивающего интерпретацию UPF-описаний в соответствии со стандартом и выполняющего анализ построенной в результате интерпретации модели подсистемы питания.

Для решения этой проблемы в рамках выполнения ОКР по созданию системы статического анализа для языка описания аппаратуры SystemVerilog SVAN [11], разрабатываемого в Институте системного программирования РАН, был разработан и реализован интерпретатор и статический анализатор модели подсистемы питания UPF. При обработке RTL-описания проекта схемы SVAN строит абстрактное синтаксическое дерево (AST, Abstract Syntax Tree), которое используется в дальнейшем для статического анализа. Это представление и UPF-описание подсистемы питания используются как входные данные для построения модели подсистемы питания. Анализ модели подсистемы питания позволяет определить ошибки и несоответствия в описании доменов питания и отдельных элементов модели, включая стратегии изоляции, состояния портов и цепей питания и т.п. В процессе анализа построенной модели можно оценить энергетическую эффективность и стабильность подсистемы питания схемы на ранних этапах разработки.

Детальное рассмотрение структуры и основных команд формата UPF приведено в разделе 2. В разделе 3 представлен обзор существующих инструментов с примерами отличия от стандарта IEEE 1801. Разделы 4 и 5 посвящены разбору реализованного инструмента для анализа модели подсистемы питания. Результаты тестирования инструмента представлены в разделе 6.

2. Формат описания подсистемы питания UPF

Формат описания UPF – это специальный формат для поведенческого описания подсистемы питания электронного компонента или системы в целом. Он позволяет разработчику формально описать систему питания проекта, включая сети питания, домены питания, отключающие элементы, ячейки изоляции, регистры сохранения состояния, а также другие задаваемые элементы. Ключевая особенность UPF заключается в возможности связать эти специальные элементы с логикой, описанной на языках проектирования VHDL или Verilog/SystemVerilog. Например, при определении доменов питания используются имена экземпляров модулей из описания проекта, а в роли сигналов контроля для отключающих элементов могут быть использованы логические сигналы из соответствующих модулей в HDL-описании. При этом стандарт UPF не предназначен для полного описания схемотехнических особенностей цепей питания, необходимых для аналоговой симуляции. Вместо этого он задает структуру питания на поведенческом уровне. Ячейки, реализующие элементы данной структуры, могут быть определены в самом UPF-описании и в отдельных файлах в формате Liberty [12], на которые может ссылаться UPF-описание. Подсистема питания в формате UPF совместно с HDL-описанием и описанием ячеек Liberty служит входными данными для симуляции, верификации и синтеза проекта. В процессе синтеза абстрактные компоненты модели питания отображаются на соответствующие библиотечные ячейки [13].

Синтаксически команды UPF представляют собой расширение скриптового языка **Tcl** (Tool Command Language) [14]. Команды, записанные в файл с расширением .upf, последовательно описывают всю архитектуру подсистемы питания проекта. Важным свойством UPF является его независимость от логических операций в HDL-описании, единственным связующим звеном является иерархия объектов: модулей, экземпляров, логических портов и сигналов и так далее. С другой стороны, логическое описание по-прежнему можно верифицировать отдельно от описания структуры электропитания. Кроме того, для одного и того же HDL-описания или их набора может существовать несколько различных UPF-описаний, предназначенных для разных сценариев использования.

Синтаксически команды UPF представляют собой расширение скриптового языка **Tcl** (Tool Command Language) [14]. Команды, записанные в файл с расширением. upf, последовательно описывают всю архитектуру подсистемы питания проекта. Важным свойством UPF является его независимость от логических операций в HDL-описании, единственным связующим звеном является иерархия объектов: модулей, экземпляров, логических портов и сигналов и т.д. С другой стороны, логическое описание по-прежнему можно верифицировать отдельно от описания структуры электропитания. Кроме того, для одного и того же HDL-описания или их набора может существовать несколько различных UPF-описаний, предназначенных для разных сценариев использования.

Стандарт UPF активно развивается, и, с течением времени, от версии к версии множество команд было изменено, удалено или добавлено. В табл. 1 представлена эволюция стандартной структуры описания подсистемы питания от версии UPF 1.0 до 3.1. В ней описаны основные компоненты подсистемы питания и соответствующие им UPF команды и их аналоги в зависимости от представленной версии. О каждом из представленных компонентов и о его роли в энергосбережении схемы будет подробно рассказано в следующем разделе.

Табл. 1. Стандартная структура UPF-описания для разных версий формата.
Table 1. Standard structure of a UPF description for different versions of the format.

Описание	UPF 1.0 / 2.0	UPF 2.1	UPF 3.0 / 3.1
Домены питания	`create_power_domain` `set_domain_supply_net`	`create_power_domain` `associate_supply_set`	`create_power_domain` `associate_supply_set`

Порты и сети питания	`create_supply_port`
	`create_supply_net`
	`connect_supply_net`
		`create_supply_port`
		`create_supply_net`
		`connect_supply_net`
			`create_supply_port`
			`create_supply_net`
			`connect_supply_net`

Состояния объектов	`add_port_state`
	`create_pst`
	`add_pst_state`
		`add_power_state`	`add_power_state`
			`add_supply_state`
			`add_state_transition`

Переключатели питания	`create_power_switch` `map_power_switch`	`create_power_switch` `map_power_switch`	`create_power_switch` `map_power_switch`

Блоки изоляции	`set_isolation`
	`set_isolation_control`
	`map_isolation_cell`
		`set_isolation`
		`use_interface_cell`
			`set_isolation`
			`use_interface_cell`

Преобразователи уровня напряжения	`set_level_shifter` `map_level_shifter_cell`	`set_level_shifter` `use_interface_cell`	`set_level_shifter` `use_interface_cell`

Регистры сохранения состояния	`set_retention`
	`set_retention_control`
	`map_retention_cell`
		`set_retention`
		`map_retention_cell`
			`set_retention`
			`map_retention_cell`

2.1 Основные компоненты подсистемы питания

В этом и последующих разделах приведены примеры команд формата UPF 3.1.

Представление RTL-описания в виде иерархии экземпляров модулей, портов и других объектов дает необходимую информацию для разработки описания подсистемы питания.

В подсистеме питания экземпляры модулей из логического описания группируются в **домены питания (power domain)**. Таким образом у разработчиков есть возможность объединить модули с одинаковыми параметрами распределения питания в один домен. Дальнейшая детализация распределения питания в системе производится в контексте доменов и их взаимодействия. Домены питания могут быть вложенными. Для каждого домена определена главная сеть питания, но могут существовать и дополнительные для отдельных частей домена. Это позволяет определять уникальные правила распределения питания и уровни напряжения, необходимые для корректной работы каждого компонента системы. Домены питания задаются с помощью команды `create_power_domain` (листинги 1, 2).

```
create_power_domain PD_TOP -elements { . }

add_power_state -supply PD_TOP.primary -update -state {OFF -illegal}
```

Листинг 1. Пример описания основного набора сетей питания домена напрямую.
Listing 1. An example of describing the primary supply set of domain power directly.

Распределение питания между отдельными доменами, модулями и внешними источниками питания обеспечивается через **порты (supply port) и сети питания (supply net)**. Режимы работы отдельных компонентов определяются состояниями сетей питания, подключенных к ним. При этом, UPF дает возможность задать не только состояние сетей и портов "включен"/"выключен", но и конкретные уровни напряжения для разных состояний. Важно упомянуть также **наборы сетей питания (supply set)**. В каждый набор входят от 2 до 6 сетей питания, каждая из которых играет отдельную роль в распределении напряжения в схеме. Особый интерес представляют 2 обязательные сети питания, отвечающие за соединение с источником напряжения и с землей. Для создания портов, сетей и наборов сетей питания используются команды `create_supply_port`, `create_supply_net` и `create_supply_set` (листинги 2 и 3) соответственно. Состояния и переходы между ними задаются командами `add_power_state` (листинг 1) и `add_state_transition` (листинг 3).

Введем понятие **основного набора сетей питания (primary supply set)**. Основной набор сетей питания является главным в домене и создается автоматически при объявлении домена. По умолчанию, основные элементы в домене питания используют его в качестве источника напряжения питания и земли. Рассмотрим примеры работы с основным набором сетей питания. В листинге 2 представлен способ описания, при котором после создания домена питания его основной набор питания описывается напрямую путем определения состояний сетей питания. В данном примере определяется состояние `OFF` как некорректное с помощью опции `-illegal`. Если в процессе верификации компонент подсистемы питания перешел в состояние, которое в описании объявлено некорректным с помощью этой опции, то инструмент верификации должен выдать соответствующее сообщение об ошибке.

С другой стороны, в листинге 3 представлен пример, в котором на основной набор подключается уже существующий набор сетей питания. Для этого в опцию `-supply` команды `create_power_domain` передаются два аргумента, первым из которых является дескриптор (`handle`) набора сетей питания `primary`, а вторым – имя существующего в текущем контексте набора `pwr_top_ss`. Подключение наборов сетей питания подразумевает соединение между собой каждой пары сетей питания, которые отвечают за одну и ту же функцию в двух наборах.

```
create_supply_net vss_n
create_supply_net vdd_n

create_supply_set pwr_top_ss \
    -function { power vdd_n } \
    -function { ground vss_n }

create_power_domain PD_TOP \
    -elements { . } \
    -supply { primary pwr_top_ss }
```

Листинг 2. Пример использования объявленного ранее набора сетей питания в качестве основного набора домена питания.
Listing 2. An example of using a previously declared supply set as the primary supply set of a power domain.

```
create_supply_port VN1 -direction inout

create_supply_net local_vdd_3 -resolve one_hot

create_supply_set relative_always_on_ss \
   -function {power vdd} \
   -function {ground vss}

set_retention pd_sw_ret -domain PD_sw \
   -save_signal {w_ret_save posedge} \
   -restore_signal {w_ret_restore posedge}

create_power_switch sw_2 -domain PD_sw \
   -input_supply_port {SW_IN pwr_2_ss.power} \
   -output_supply_port {SW_OUT sw_pwr_2_ss.power} \
   -control_port {SW_DIS w_d1_sw_disable} \
   -on_state {ON_STATE SW_IN {!SW_DIS}} \
   -off_state {OFF_STATE {SW_DIS}}

add_state_transition -domain PDA \
   -transition {turn_on  -from OFF_MODE -to NORMAL_MODE} \
   -transition {suspend  -from NORMAL_MODE -to SLEEP_MODE} \
   -transition {resume   -from SLEEP_MODE -to NORMAL_MODE} \
   -transition {turn_off -from NORMAL_MODE -to OFF_MODE}
```

Листинг 3. Примеры использования некоторых UPF-команд в соответствии со стандартом IEEE 1801-2018.
Listing 3. Examples of using some UPF commands according to IEEE 1801-2018.

Отдельное внимание стоит уделить понятию эквивалентности портов, сетей питания и наборов сетей питания. Выделяют два типа эквивалентности: электрическую и функциональную. Наличие электрической эквивалентности означает также функциональную эквивалентность, в то время как обратное неверно. Неразрешенной сетью питания называется сеть, имеющая один источник питания. Соответственно, разрешенной сетью питания называется та сеть питания, которая имеет несколько источников питания и набор правил, по которым используется питание от того или иного источника. Вводятся следующие правила:

1. Порт P (port) или сеть N (net) питания электрически эквивалентны сами себе.
2. Если неразрешенная сеть питания N и порт питания P соединены, то N и P электрически эквивалентны.
3. Если разрешенная сеть питания N и порт питания P соединены и P – двунаправленный порт или является выходным сигналом для сети N, то N и P электрически эквивалентны.

4. Если все функции, имеющиеся у двух наборов сетей питания, являются электрически/функционально эквивалентными, то эти наборы сетей питания являются электрически/функционально эквивалентными.
5. Электрическая эквивалентность является транзитивной.
6. Если порты (или сети) реализуют функции в одном наборе сетей питания, то они являются электрически эквивалентными.
7. Если объекты одинакового типа (порты, сети или наборы сетей питания) явно объявлены электрически/функционально эквивалентными с помощью команды `set_equivalent`, то эти объекты электрически/функционально эквивалентны.

В зависимости от типа эквивалентности может меняться множество портов, удовлетворяющих тем или иным фильтрам, в частности, для стратегий изоляции и преобразования уровня напряжения. Детальный разбор фильтров этих стратегий представлен в разделе 5.1.

Корректное взаимодействие компонентов с разными уровнями напряжения и режимами питания обеспечивается с помощью **блоков изоляции (isolation cell)** и **преобразователей уровня напряжения (level shifter)**.

Отключение отдельных частей схемы является одним из основных методов энергосбережения. Блоки изоляции играют ключевую роль в корректной реализации этого метода. Они вставляются между двумя объектами системы, один из которых может перейти в режим “выключен”, тогда как другой продолжает работу. Это предотвращает возможные ошибки из-за неопределенного состояния выходных сигналов отключенного объекта, заменяя их на известное значение (логические 0 и 1 или захваченное значение). Благодаря этому выполняется безопасное отключение отдельных компонент системы, что уменьшает общее энергопотребление.

Другим методом экономии питания является определение разных уровней напряжения для частей схемы. Таким образом отдельные части схемы потребляют меньше питания. Преобразователи уровня напряжения, в свою очередь, обеспечивают корректную связь между двумя включенными объектами, режимы работы которых заданы с разными уровнями напряжения.

Некоторые компоненты системы, в частности блоки памяти, часто переводятся в отключенный режим в целях энергосбережения. При этом важно не потерять значение, сохраненное в этих компонентах. Для этого используются **регистры сохранения состояния (retention state register)**. Они подключаются к отдельным источникам питания, отличным от основного питания домена, и поддерживают минимальное напряжение для сохранения значений в ячейках памяти внутри отключенного домена. Это ускоряет инициализацию модуля при включении питания.

Стратегии работы блоков изоляции, преобразователей уровня напряжения и регистров сохранения состояния задаются командами `set_isolation` (листинг 4), `set_level_shifter` (листинг 4) и `set_retention` (листинг 3) соответственно.

Еще одним важным элементом в обеспечении корректной работы доменов и их внутренних компонент и увеличения энергосбережения являются **переключатели или ячейки отключения питания (power switch cell)**. Для создания таких переключателей используется команда `create_power_switch` (листинг 4).

В листингах 1, 2, 3 и 4 представлены примеры использования перечисленных выше команд. В частности, в листинге 4 представлен простейший пример HDL-описания схемы на языке SystemVerilog и описания подсистемы питания в формате UPF.

Описание проекта на SystemVerilog (слева) состоит из трех модулей: одного родительского модуля top с одноименным экземпляром и двух дочерних модулей `power_domain1` и `power_domain2` с экземплярами `pd1` и `pd2` соответственно. В UPF-описании подсистемы

питания (справа) экземпляры модулей распределены на три домена питания (`PD_TOP`, `PD1` и `PD2`). Для основных сетей питания каждого из доменов определены состояния “включен” и “выключен”, причем уровень напряжения включенного состояния отличается у всех трех (`1`, `0.7` и `0.8` соответственно). На границах между родительским модулем и дочерними определены стратегии изоляции (`ISO_PD1` и `ISO_PD2`) и стратегии преобразования уровня напряжения (`LS_PD1` и `LS_PD2`).

```
module top (
  input logic clk,
  input logic reset,
  input logic data_in,
  output logic data_out
);
  logic data_intermediate;
  power_domain1 pd1 (
   .clk(clk),
   .reset(reset),
   .data_in(data_in),
   .data_out(data_intermediate)
  );

  power_domain2 pd2 (
   .clk(clk),
   .reset(reset),
   .data_in(data_intermediate),
   .data_out(data_out)
  );
endmodule

module power_domain1 (
  input logic clk,
  input logic reset,
  input logic data_in,
  output logic data_out
);
endmodule

module power_domain2 (
  input logic clk,
  input logic reset,
  input logic data_in,
  output logic data_out
);
Endmodule
```

```
set_design_top top
set_scope .

create_power_domain PD_TOP \
   -elements top
create_power_domain PD1 \
   -elements pd1
create_power_domain PD2 \
   -elements pd2

add_power_state -supply PD_TOP.primary \
  -state {ON -supply_expr \
{power == { 1 } && ground == { 0 }}} \
  -state {OFF -supply_expr \
{power == { OFF } && ground == { 0 }}}

add_power_state -supply PD1.primary \
  -state {ON -supply_expr \
{power == { 0.7 } && ground == { 0 }}} \
  -state {OFF -supply_expr \
{power == { OFF } && ground == { 0 }}}

add_power_state -supply PD2.primary \
  -state {ON -supply_expr \
{power == { 0.8 } && ground == { 0 }}} \
  -state {OFF -supply_expr \
{power == { OFF } && ground == { 0 }}}

set_isolation ISO_PD1 -domain PD1 \
   -elements pd1/data_in
set_isolation ISO_PD2 -domain PD2 \
   -elements pd2/data_in

set_level_shifter LS_PD1 -domain PD1 \
   -applies_to inputs
set_level_shifter LS_PD2 -domain PD2 \
   -applies_to inputs
```

Листинг 4. Пример HDL-описания схемы на языке SystemVerilog (слева) и описания соответствующей подсистемы питания в формате UPF (справа).
Listing 4. An example of a circuit design in SystemVerilog (left) and a description of the corresponding power subsystem in UPF format (right).

3. Обзор существующих решений

Для формирования методологии анализа было проведено исследование существующих инструментов, способных интерпретировать и анализировать UPF-описания. Целью исследования было выявление ключевых правил и критериев анализа. Особое внимание было уделено основным компонентам в описаниях подсистемы питания, представленным в разделе 2.1, и связанным с ними ошибкам.

3.1 Коммерческие инструменты

Ведущие вендоры инструментов для разработки сверхбольших интегральных схем (СБИС) предоставляют возможность работать с форматом UPF. Эти решения имеют высокую степень поддержки стандарта и возможность анализа UPF-описания, но высокая стоимость лицензии и ограничение доступности в России серьезно усложняют применение этих инструментов.

Cadence [15] – это один из ведущих вендоров передовых САПР СБИС. Компанией Cadence в 2007 году был предложен Common Power Format (CPF), который поддерживается стандартом Si2 [16]. Именно этот формат остается приоритетным в инструментах компании. Отличия между двумя форматами минимальны, в основном связаны с взаимодействием со сторонними библиотеками. При этом UPF, активно развивающийся и поддерживаемый стандартом IEEE, стал более распространен в индустрии. Большинство инструментов ориентируются на UPF, из-за чего их интеграция в процесс разработки вместе с системами Cadence вызывает сложности. Хотя схожесть CPF и UPF позволяет инструментам Cadence обрабатывать UPF-описание, отсутствие общепринятого стандарта и проблема взаимодействия с другими инструментами затрудняют разработку энергоэффективных СБИС. Стоит заметить, что, начиная с 2014 года Cadence тоже начали выводить на передний план, однако поддержка версий UPF 3.0 и далее пока отстает.

Synopsys предоставляет широкий спектр возможностей при работе с UPF. Инструмент статической верификации и анализа SpyGlass [17], помимо проверки RTL-кода, также имеет набор из свыше 200 правил, относящихся к UPF. Это как семантические и синтаксические ошибки, так и оценка функциональной эффективности описанной подсистемы питания, и предложение альтернативных применений методов экономии энергопотребления.

VCS [18] (Verilog Compiled Simulator), также разработанный компанией Synopsys, – это высокопроизводительный симулятор и инструмент для анализа описаний на языках Verilog и SystemVerilog. Он также позволяет интерпретировать UPF.

Siemens предоставляет инструмент **Questa** [19], который широко применяется на этапах верификации, тестирования и анализа энергопотребления интегральных схем. Поддержка стандарта IEEE 1801-2018 позволяет проектировать цифровую аппаратуру с учетом эффективного потребления энергии и анализировать разработанные схемы в разных режимах питания. Модуль Power Aware Simulation дает возможность учитывать состояния доменов питания и переходы между этими состояниями при моделировании работы схемы. Это позволяет провести детальную проверку семантического и функционального описания компонентов подсистемы питания и обнаруживать ошибки на ранних стадиях разработки до физического синтеза.

В виде примера для демонстрации отличий стандарта IEEE 1801 и коммерческих инструментов представлен рис. 1. Здесь команды стандарта UPF IEEE 1801-2018, используемые в описании подсистемы питания, разделены на группы в зависимости от выполняемой роли в описании. Красным выделены команды, которые отсутствуют в документации Synopsys, посвященной использованию формата UPF в ряде инструментов компании. Синим выделены команды, описание которых в документации отличается от стандарта. Отличия могут быть синтаксические, например, другая расстановка фигурных скобок для команды `add_power_state`, отсутствие некоторых опций команды и другие.

3.2 Инструменты с открытым исходным кодом

Высокая стоимость лицензий коммерческих решений ограничивает их использование для большинства малых бизнесов и научно-образовательной сферы. Этот фактор является основной движущей силой в вопросе разработки решений с открытым исходным кодом. При этом интеграция и поддержка формата UPF в таких инструментах пока сильно отстает в степени поддержки стандарта от коммерческих аналогов. Одним из наиболее развитых является проект OpenROAD.

OpenROAD [20] – это набор инструментов для автоматизации этапов разработки цифровой аппаратуры. Поддержка формата UPF была добавлена в 2022 году и продолжает развиваться. На данный момент OpenROAD позволяет описывать и обрабатывать базовые структуры и объекты подсистемы питания. При этом поддерживается только часть команд UPF [21], а сохранение информации для возможной передачи на следующие этапы разработки полностью не реализована. Также на данный момент не реализованы средства верификации и проверки корректности описания UPF.

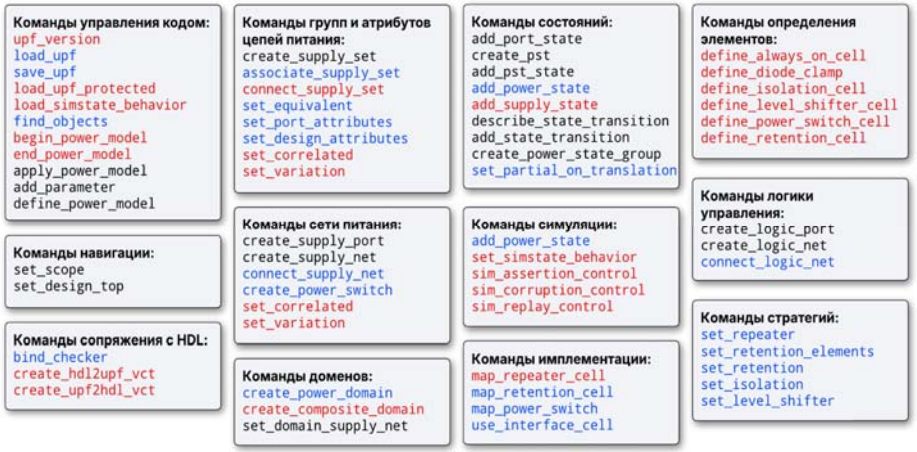

Рис. 1. UPF-команды стандарта IEEE 1801-2018.
Fig. 1. UPF-commands from IEEE 1801-2018.

3.3 Выводы

С распространением стандарта IEEE-1801 анализ UPF-описания подсистемы питания становится все более и более востребованным. Крупные компании, предоставляющие комплексные решения для разработки СБИС, уже реализуют соответствующие инструменты и модули для этой цели. Но на данный момент все предоставляют только ограниченную поддержку стандарта. Вместе с этим высокая стоимость лицензии делает затруднительным их широкое использование для малых компаний и научно-исследовательских организаций.

Инструменты с открытым исходным кодом в свою очередь сильно отстают по уровню поддержки стандарта UPF и возможностей работы с подсистемой питания. Ограниченные ресурсы сообщества замедляют развитие таких решений, из-за чего они пока не пригодны для использования в индустрии. В связи с представленным обзором можно заключить, что реализация решения по разбору и интерпретации команд UPF в соответствии со стандартом приобретает особую важность при создании отечественных инструментов проектирования цифровых СБИС.

4. Построение модели подсистемы питания

Из примеров кода в листингах 1-4 можно заметить, что для корректной интерпретации UPF-описания и построения соответствующей модели подсистемы питания необходимо иметь информацию об объектах из HDL-описания. В связи с этим в нашей разработке было решено в качестве входных данных для инструмента помимо файлов с описанием подсистемы питания использовать абстрактное синтаксическое дерево (АСД), полученное при обработке соответствующего HDL-описания (рис. 2). В этой главе представлен подход к интерпретации, реализованный в разработанном программном обеспечении.

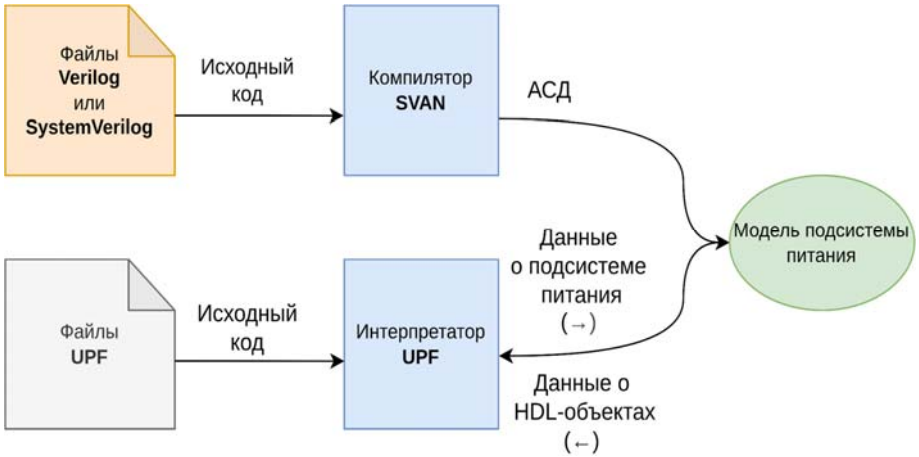

Рис. 2. Схема построения полной информационной модели из HDL-описания и UPF-описания.
Fig. 2. Scheme for constructing a complete information model from an HDL description and a UPF description.

4.1 Обработка HDL-описания и библиотек ячеек Liberty

Прежде чем переходить к интерпретации UPF-описания, надо обработать HDL-описание схемы и получить все необходимые данные из файлов с описанием стандартных ячеек в формате Liberty, поставляемых в PDK-пакетах заводом изготовителем схем. Liberty-описание ячейки содержит, в том числе, представление логической функции и значения физических характеристик определенного компонента схемы. Это требование обусловлено тем, что HDL-описание предоставляет иерархию экземпляров модулей, портов и других объектов, которая необходима для корректного определения компонент подсистемы питания, а библиотечные ячейки указываются в таких командах, как `use_interface_cell` и `map_power_switch`, того, чтобы задать какая именно ячейка должна реализовать данный компонент на схеме.

Обработка HDL-описания выполняется соответствующим модулем из системы статического анализа **SVAN**, разрабатываемой в ИСП РАН. SVAN основан на открытой библиотеке **slang** [22] для анализа и компиляции описаний на языке SystemVerilog. Поддержка стандарта SystemVerilog IEEE 1800 [23], а также создание АСД на основе входного кода делают slang лучшим кандидатом для использования в разработке собственной системы статического анализа. Крайне полезным является также интерфейс обхода АСД, который позволяет писать пользовательские методы прохода АСД для поиска нужной информации в RTL-модели.

Еще одним инструментом, необходимым для полной и корректной интерпретации UPF, является открытая библиотека **ReadCells** [24]. Она также разрабатывается в ИСП РАН и предназначена для парсинга и обработки языка описания библиотек стандартных ячеек Liberty.

Инструмент ReadCells хранит описание ячеек во внутреннем представлении, что позволяет задействовать их на различных стадиях проектирования. Эта возможность критична для интеграции библиотечных элементов, описанных в Liberty, со стратегиями управления энергопотреблением, определенными в формате UPF.

Библиотека ReadCells была интегрирована в инструмент SVAN для возможности проведения анализа описаний схемы, требующего информацию о характеристиках Liberty-ячеек. В рамках интеграции ReadCells в SVAN был разработан транслятор Liberty в Verilog для

возможности межмодульного анализа, чтобы экземпляры Liberty ячеек не представлялись в модели, как “черные ящики”.

4.2 Интерпретация UPF-описания

Как уже упоминалось, Unified Power Format является надстройкой над Tcl. Начиная от версии UPF 2.1, в стандарте IEEE 1801 явно указывается, что для корректной обработки UPF файлов требуется интерпретатор Tcl версии 8.4 или выше.

Также, благодаря последовательному выполнению команд как в UPF, так и в Tcl, в описании подсистемы питания допускается использование Tcl-процедур [25], библиотек и скриптов с использованием Tcl-команд общего назначения `foreach`, `set` и т.д. При этом доступ к объектам HDL-описания производится с помощью только одной команды `find_objects`.

Из листинга 1 можно заметить, что команды UPF по структуре схожи с командами языка командной оболочки (например, такого, как `bash`): ключевые слова, набор аргументов, опций и флагов. Исходя из этого сходства было принято решение использовать библиотеку **CLI11** [26] для парсинга аргументов. CLI11 предназначена для работы с командной строкой и предоставляет все инструменты для обработки команд, аргументов и опций.

Поддержка UPF в разработанном инструменте реализована с помощью Tcl C API. В стандарте IEEE 1801-2018 описано 63 команды, для каждой из которых в разработанном программном обеспечении реализован собственный обработчик. Команда `Tcl_CreateCommand` регистрирует UPF-команды в интерпретаторе и добавляет соответствующие им обработчики в Tcl. В процессе интерпретации UPF-описания, встретив зарегистрированную команду, инструмент вызывает нужный обработчик, который, закончив работу, возвращает необходимые данные и код состояния `TCL_OK` или `TCL_ERROR` при корректном завершении работы или ошибке соответственно.

На листинге 5 представлен пример обработчика команды UPF `set_correlated`. При вызове обработчика как аргумент передается объект `UPFReader`. Он управляет процессом и хранит всю информацию о текущем состоянии интерпретации UPF-файла.

Эта информация хранится в виде объектной модели. Для каждого домена питания, порта и сети питания, стратегий и других элементов, описанных соответствующей UPF-командой подсистемы питания, создается объект определенного класса с параметрами, заданными в опциях команды. Этот объект присоединяется с остальными и может изменяться или дополняться в будущем при обработке другой команды. Построение полноценной модели подсистемы питания является главной целью этапа интерпретации UPF-описания.

Через объект `UPFReader` также можно получить доступ к `CLI::App`. Это парсер обрабатываемой команды, который инициализируется только один раз при первой встрече этой команды. CLI позволяет в процессе парсинга провести первоначальную обработку аргументов, конвертируя строки в списки или числа. Также CLI предоставляет интерфейс для определения некоторых характеристик для опций и аргументов, таких как количество или тип аргументов, возможность задавать некоторые опции в одной команде или запрет установления определенного набора опций. Таким образом парсер выполняет роль первого уровня проверки корректности UPF-описания.

После успешного парсинга и извлечения аргументов запускается выполнение логики команды. В зависимости от команды этот этап может включать в себя создание новых объектов в модели UPF, изменение характеристик или установку связей между объектами, поиск объектов в RTL-модели и другие действия. При успешном выполнении логики изменения сохраняются в модели внутри объекта `UPFReader` и возвращается код статуса `TCL_OK`. В противном случае управление передается в блок `catch`, который возвращает строку ошибки и код `TCL_ERROR`. Такая реализация позволяет останавливать интерпретацию в любой момент.


```
int UPF_set_correlated(ClientData data, Tcl_Interp* interp,
                       int argc, const char* argv[]) {
    auto* reader = static_cast<UPFReader*>(data);
    if (reader->flags().has(UPFContextFlag::Block))
        return reader->save_cmd(argc, argv);
    auto [cli, exists] = reader->getCLI(argv[0]);

    if (!exists) {
        // Регистрация опций команды
        auto* group = cli->add_option_group("")->require_option();
        group->add_option("-nets")->expected(1);
        group->add_option("-sets")->expected(1)->excludes("-nets");
    }
    try {
        // Парсинг
        cli->parse(argc, argv);

        // Получение значений опций
        CLI::results_t nets_str, sets_str;
        cli->get_option("-nets")->results(nets_str);
        cli->get_option("-sets")->results(sets_str);
        TclList<TclList<std::string>> nets(interp, nets_str,
                                           SkipFirstBrackets());
        TclList<TclList<std::string>> sets(interp, nets_str,
                                           SkipFirstBrackets());

        // Реализация логики команды
        ...
    }
    catch (const std::runtime_error& e) {
       Tcl_SetResult(interp, const_cast<char*>(e.what()), TCL_VOLATILE);
       return TCL_ERROR;
    }
    return TCL_OK;
}
```

Листинг 5. Обработчик UPF-команды set_correlated из стандарта IEEE 1801-2018.
Listing 5. The handler for UPF command set_correlated from the IEEE 1801-2018 standard.

5. Статический анализ модели подсистемы питания

Анализ UPF-описания направлен на проверку корректности построенной модели подсистемы питания. Свойства, связанные с проверкой корректности, можно разделить на два типа:

1) корректность конкретных языковых конструкций в описании подсистемы питания, например:
   a) использование неподдерживаемой стандартом UPF команды либо опции команды;
   b) вызов объекта (порт питания, сеть питания, домен и т.д.) до его создания или вне области видимости, в которой она была создана;
   c) некорректное использование опций команд или передача некорректных аргументов;
2) корректность полученной модели, например:
   a) проверка отсутствующих или некорректно заданных объектов, в частности стратегий изоляции или преобразования уровня напряжения;
   b) оценка энергоэффективности заданной подсистемы питания.

Интерпретатор вместе с обработкой файла и построением модели выполняет проверку свойств первого типа. Для проверки свойств второго типа реализован статический анализатор модели подсистемы питания. В рамках данной работы разрабатывались методы статического анализа двух компонентов из перечисленных в разделе 2.1: блоки изоляции и преобразователи уровня напряжения.

Блоки изоляции и преобразователи уровня напряжения на практике задаются двумя частями: стратегия изоляции (преобразования уровня напряжения) и описание ячейки, которая будет выполнять функцию изоляции (преобразователя уровня напряжения). Команды `set_isolation` и `set_level_shifter` задают стратегию блока изоляции и преобразователя уровня соответственно. Описание ячейки может быть задано тремя способами:

1. в виде отдельного файла в библиотеке стандартных ячеек на языке Liberty;
2. внутри UPF-описания с помощью команд `define_isolation_cell` и `define_level_shifter_cell`;
3. вместе с HDL-описанием.

Далее UPF-команда `use_interface_cell` связывает соответствующую стратегию с ячейкой.

В следующем подразделе представлено описание реализации проверок существования и корректности стратегий изоляции и преобразования уровня напряжения.

5.1 Проверка стратегий изоляции и преобразования уровня напряжения

Прежде чем перейти к проверке стратегий, необходимо определить области их применения, способы задания, а также возможные последствия нарушения их корректности. Как уже упоминалось в разделе 2.1, блоки изоляции и преобразователи уровня напряжения обеспечивают корректную и безопасную связь между доменами питания, у которых отличаются режимы работы.

Домены питания представляют собой группы экземпляров модулей (или ячеек) из HDL-описания схемы. В каждом домене есть так называемые граничные экземпляры – экземпляры, порты которых выходят наружу, соединяя их с такими же граничными экземплярами соседних доменов. При описании стратегий изоляции и преобразования уровня напряжения через опции соответствующих команд задаются фильтры, которые определяют, для каких портов задана данная стратегия. Таким образом, проверка наличия и корректности стратегий сводится к обходу всех портов граничных экземпляров и определению для каждого из них соответствующей стратегии путем сопоставления параметров порта с условиями заданных фильтров.

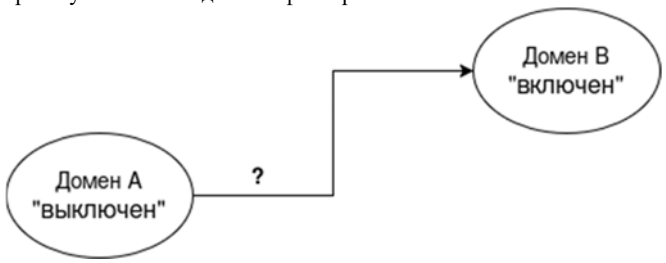

Рис. 3. Пример схемы с ошибкой отсутствия необходимого блока изоляции между доменами A и B.
Fig. 3. An example of a circuit with an error due to missing required isolation cell between power domains A and B.

Если между двумя соседствующими доменами нет необходимой стратегии или для некоторого порта на границе этих доменов не выполняются условия ни одной из стратегий, определенных для этих доменов, это указывает на нарушение требования существования и корректности описания стратегий. В случае отсутствия корректной стратегии изоляции это приведет к ситуации, где из выключенного домена во включенный передается неопределенное значение (рис. 3), что может привести к некорректной работе домена. Отсутствие стратегии преобразования уровня напряжения, приведет к передаче сигнала между двумя доменами с разным уровнем напряжения, что может привести как к ошибке работы доменов, так и выходу из строя всей схемы.

Необходимость применения стратегии изоляции на границе двух доменов питания определяется на основе анализа возможных режимов работы этих доменов. Если существует состояние схемы, при котором один из доменов находится в состоянии “выключен” или же основное питание домена (`primary supply set`) находится в состоянии “выключено”, в то время как другой домен “включен” либо его основное питание находится в состоянии “включено” и при этом не зависит от основного питания первого домена, то на границе необходима стратегия изоляции.

На рис. 4 представлен пример части схемы, где стратегия изоляции необходима, вместе с примером реализации ячейки изоляции через логическое “И”. В этом примере, когда домен A находится в состоянии “выключен”, значение сигнала на его выходе неопределенное. Для предотвращения распространения такого сигнала выход домена A подается на логический элемент «И» совместно с постоянным значением 0, что обеспечивает фиксированное значение на выходе изоляционной ячейки. При включении домена A постоянный вход в ячейку изоляции переводится в значение 1. Таким образом ячейка изоляции будет передавать в домен B корректный сигнал в результате работы домена A.

Если существует состояние схемы, при котором оба домена включены, но при этом их основное питание подключено к источникам с разным уровнем напряжения, то между ними должен присутствовать преобразователь уровня напряжения. На рис. 5 представлен пример схемы с ячейкой преобразования уровня напряжения. Реализация преобразования уровня может быть выполнена с помощью двух ячеек, по одной в каждую сторону перехода. В таком случае опции `-rule` команды `set_level_shifter` для каждой из стратегий должно быть передано соответствующее значение `low_to_high`/`high_to_low`. Альтернативное описание стратегии преобразования уровня напряжения может быть выполнено с использованием одной команды `set_level_shifter`: одна стратегия со значением опции `-rule both`.

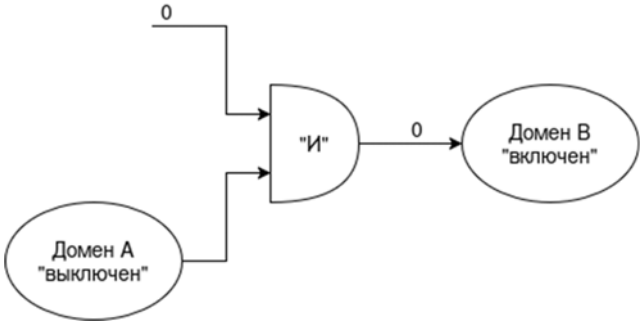

Рис. 4. Пример схемы с реализацией ячейки изоляции через логическое “И”.
Fig. 4. An example of a circuit implementing an isolation cell using a logical “AND”.

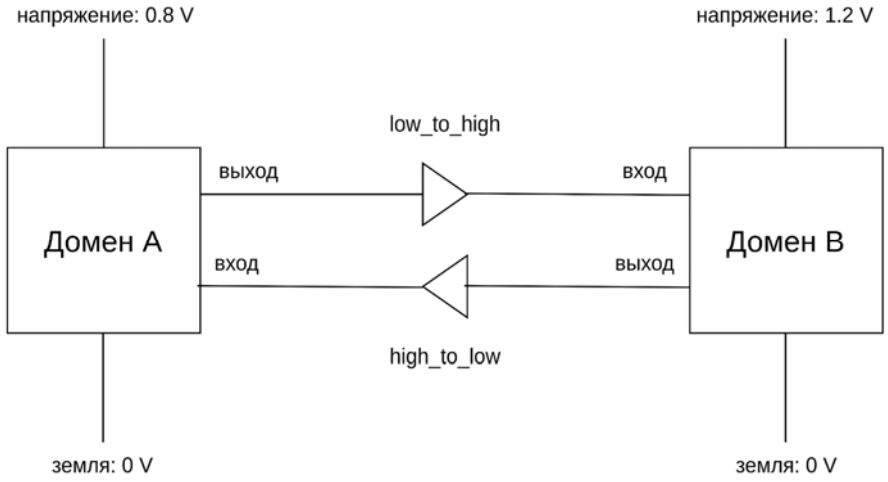

Рис. 5. Пример схемы с использованием стратегии преобразования уровня напряжения.
Fig. 5. An example circuit using voltage level shifter strategy.

В стандарте IEEE 1801-2018 представлены фильтры стратегий **преобразования уровня напряжения** и **изоляции**, которые определяют для каких портов задана данная стратегия.

Фильтры стратегии **изоляции** (рис. 6):

- Виды сети питания получателя и источника сигнала определяются аналогично тому, как это происходит для стратегии преобразования уровня напряжения. Также идентичны фильтры для местонахождения ячейки на схеме, проверок границы домена и направления порта.
- Значение при включении стратегии: при отключении основного питания, ячейка, реализующая стратегию изоляции, должна поддерживать уровень напряжения, соответствующий значению `UPF_clamp_value`, указанному в качестве атрибута порта. Допустимо одно из следующих значений: `0`, `1`, `Z`, фиксация последнего значения порта, любое, а также пользовательское. Фильтр позволяет установить стратегию изоляции на порты, которые будут иметь указанное значение при отключении питания. Задается через опцию `-applies_to_clamp`.
- Значение при выключении источника сигнала: в случае, если состояние сети питания, в которую передается сигнал, изменяется на любое, кроме состояния для режима симуляции, помеченного как активный, порт должен принять значение, указанное с помощью атрибута порта `UPF_sink_off_clamp_value`. Фильтр позволяет установить стратегию изоляции на порты, которые будут иметь указанное значение при указанном событии. Задается через опцию `-applies_to_sink_off_clamp`.
- Значение при выключении источника сигнала: в случае, если состояние сети питания, из которой передается сигнал, изменяется на любое, кроме состояния для режима симуляции, помеченного как активный, порт должен принять значение, указанное с помощью атрибута порта `UPF_source_off_clamp_value`. Фильтр позволяет установить стратегию изоляции на порты, которые будут иметь указанное значение при указанном событии. Задается через опцию `-applies_to_source_off_clamp`.

- Единая сеть питания: фильтр определяет, являются ли сети питания между двумя портами эквивалентными. По умолчанию, ячейки изоляции ставятся в тех случаях, когда основные сети питания доменов не эквивалентны друг другу. Можно указать, что стратегия изоляции является применимой в случаях, когда это условие нарушено, с помощью опции `-diff_supply_only`, передав аргумент `FALSE`/`TRUE` в зависимости от того, требуется ли отсутствие или наличие эквивалентности. Опция `-use_functional_equivalence`, получающая те же аргументы, позволяет указать, проверяется ли функциональная эквивалентность между сетями питания.

```
set_isolation strategy_name
  -domain domain_name
  [-elements element_list]
  [-exclude_elements exclude_list]
  [-source <source_domain_name | source_supply_ref>]
  [-sink <sink_domain_name | sink_supply_ref>]
  [-diff_supply_only [<TRUE | FALSE>]]
  [-use_functional_equivalence [<TRUE | FALSE>]]
  [-applies_to <inputs | outputs | both>]
  [-applies_to_boundary <lower | upper | both>]
  [-applies_to_clamp <0 | 1 | any | Z | latch | value>]
  [-applies_to_sink_off_clamp <0 | 1 | any | Z | latch | value>]
  [-applies_to_source_off_clamp <0 | 1 | any | Z | latch | value>]
  [-no_isolation]
  [-force_isolation]
  [-location <self | other | parent | fanout>]
  [-clamp_value <0 | 1 | Z | latch | value | {<0 | 1 | Z | latch | value>*}>]
  [-isolation_signal signal_list [-isolation_sense <high | low | {<high | low>*}>]]
  [-isolation_supply supply_set_list]
  [-name_prefix pattern] [-name_suffix pattern]
  [-instance {{instance_name port_name}*}]
  [-update
  [-isolation_power_net net_name] [-isolation_ground_net net_name]
  [-use_equivalence [<TRUE | FALSE>]]
```

Рис. 6. Описание опций команды set_isolation из стандарта IEEE 1801-2018.
Fig. 6. Set isolation command options description from IEEE 1801-2018.

Фильтры стратегии **преобразования уровня напряжения** (рис. 7):

- Граница домена: иерархическая структура подсистемы питания означает, что у каждого домена питания может быть верхняя и нижняя границы. Фильтр показывает, для портов какой именно границы задана данная стратегия (lower/upper/both). Значение задается через опцию `-applies_to_boundary`.
- Направление порта: фильтр показывает, для портов с каким направлением задана данная стратегия (`inputs`/`outputs`/`both`). Значение задается через опцию `-applies_to`.
- Указание сети питания получателя сигнала: стратегия задана только для тех портов, для которых сеть питания на стороне получателя сигнала имеет определенные характеристики. Значение задается через опцию `-sink`.

- Указание сети питания источника сигнала: стратегия задана только для тех портов, для которых сеть питания на стороне источника сигнала имеет определенные характеристики. Значение задается через опцию `-source`.
- Направление изменения напряжения: фильтр ограничивает распространение стратегии для тех портов, при переходе через которые уровень напряжения меняется в заданном направлении (`low_to_high/high_to_low/both`). По умолчанию значение фильтра `both`. Значение задается через опцию `-rule`.
- Минимальная разница напряжений: фильтр определяет минимальную разницу между уровнями напряжения на двух концах порта, для которой задана стратегия. Значение по умолчанию равно **0**. Значение задается через опцию `-threshold`.
- Местонахождение ячейки в схеме, которая будет реализовывать эту стратегию по отношению к домену, определяющего стратегию (`self/parent/fanout/other`). Значение по умолчанию `self`. Значение задается через опцию `-location`.

```
set_level_shifter strategy_name
    -domain domain_name
    [-elements element_list]
    [-exclude_elements exclude_list]
    [-source <source_domain_name | source_supply_ref>]
    [-sink <sink_domain_name | sink_supply_ref>]
    [-use_functional_equivalence [<TRUE | FALSE>]]
    [-applies_to <inputs | outputs | both>]
    [-applies_to_boundary <lower | upper | both>]
    [-rule <low_to_high | high_to_low | both>]
    [-threshold <value>]
    [-no_shift] [-force_shift]
    [-location <self | other | parent | fanout>]
    [-input_supply supply_set_ref] [-output_supply supply_set_ref]
    [-internal_supply supply_set_ref]
    [-name_prefix pattern] [-name_suffix pattern]
    [-instance {{instance_name port_name}*}]
    [-update]
    [-use_equivalence [<TRUE | FALSE>]]
```

Рис. 7. Описание опций команды set_level_shifter из стандарта IEEE 1801-2018.
Fig. 7. Set level shifter command options description from IEEE 1801-2018.

Перед выполнением проверки проводится обход каждого домена в модели подсистемы питания и сбор списка структур, состоящих из пар соседствующих доменов и списка портов между граничными экземплярами этих доменов. Для каждой структуры далее проводится проверка корректности стратегий. Объекты доменов питания хранят внутри все стратегии, определенные в соответствующих доменах.

Первым этапом проверки является определение необходимости стратегии между данными доменами в соответствии с приведенными выше критериями для каждой из стратегий. В случае подтверждения необходимости выполняется проверка наличия соответствующих стратегий хотя бы в одном из соседствующих доменов. Если стратегий нет, то выдается сообщение об отсутствии необходимой стратегии (листинг 6, строки 1-2 и 4-5).

Если список стратегий на доменах не пустой, то для каждого порта на границе и для каждой стратегии данных доменов проверяется соответствие порта фильтрам стратегии. Если в

результате обхода всех стратегий не нашлось хотя бы одной, всем фильтрам которой порт бы соответствовал, то выдается сообщение об ошибке с указанием порта, номера строки с определением стратегии (одной из стратегий) и фильтра (одного из фильтров), которым не удовлетворяет данный порт (листинг 6, строки 7-8 и 10-11).

В листинге 6 строки 1-2 и 7-8 соответствуют ошибкам стратегии изоляции, а строки 4-5 и 10-11 – стратегии преобразования уровня напряжения. Строки 4-5 соответствуют ошибке фильтра `-applies_to_boundary`, строки 10-11 – фильтру `-threshold`.

```
1  notset-iso.upf:0: fatal error: IsolationNotSet: isolation strategy was not set to
2  the port '/topt/u_blink_ctrl/clk', but isolation is required
3
4  notset.upf:0: fatal error: LevelShifterNotSet: level shifter strategy was not set
5  to the port '/topt/chk1/p1', but level shifter is required
6
7  bound-iso.upf:152: fatal error: InvalidStrategyBoundary: port named
8  '/topt/u_blink_ctrl/clk' has no isolation strategy, boundary filter failed
9
10 thres.upf:39: fatal error: InvalidLevelShifterThreshold: port named
11 '/topt/chk1/p1' has no required level shifter strategy, threshold filter failed
```

Листинг 6. Примеры сообщений ошибок при отсутствии (строки 1-2 и 4-5) и некорректном определении стратегий (строки 7-9 и 11-13).
Listing 6. Examples of error messages for missing (lines 1-2 and 4-5), and incorrectly defined strategies (lines 7-9 and 11-13).

5.2 Интеграция UPF в статический анализатор SVAN

Для корректного анализа описания подсистемы питания в инструмент анализа SVAN была добавлена опция `--upf`. В листинге 7 приведен пример запуска инструмента с для проверки проекта вместе с описанием подсистемы питания. В файле `design.sv` передается HDL-описание проекта, а опция `--upf` в виде аргумента принимает путь к UPF-файлу с соответствующим описанием (`power.upf`). Если описание подсистемы питания разделено на несколько файлов, то инструменту передается только один файл, в котором определяется домен питания, соответствующий главному модулю HDL-описания. Переход к остальным файлам должен быть корректно задан в описании подсистемы питания через команду `load_upf` и инструмент при обработке этой команды также перейдет к соответствующим файлам. Анализатор после построения АСД проекта схемы запускает интерпретацию UPF-файла. При успешном завершении этапа интерпретации построенная модель подсистемы питания передается в модуль статического анализа UPF-описания.

```
svan design.sv --upf power.upf
```

Листинг 7. Пример команды запуска инструмента.
Listing 7. Example command to run the tool.

При обнаружении синтаксической или семантической ошибки инструмент передает в стандартный поток вывода сообщение с именем файла, номером строки и описанием обнаруженной ошибки (листинг 8). Ошибка `FindFail` означает, что в описании используется имя объекта `"/sw_2/SW_OUT"`, который не был объявлен заранее. Такая ошибка может возникнуть при отсутствии необходимого объявления или при опечатке в написании имени объекта при использовании. Сообщения ошибок `InvalidLevelShifterRule` и `InvalidStrategyDirection` указывают на

некорректное описание стратегий. В первом случае фильтр `-rule` команды `set_level_shifter`, описанный в разделе 5.1, задан противоположно направлению порта `"/topt/chk3/p"`, который находится на границе между двумя доменами, вследствие чего этот порт остается непокрытым стратегией преобразования уровня напряжения, что является ошибкой. Вторая ошибка указывает на схожее нарушение, связанное с фильтром `-applies_to`. Так как этот фильтр присутствует в стратегии изоляции и в стратегии преобразования уровня напряжения, текст сообщения указывает в какой именно команде найдена ошибка. В данном случае указана стратегия преобразования уровня напряжения `–level shifter strategy`.

```
dut.upf:138: fatal error: FindFail: Object "/sw_2/SW_OUT" was not found in name map

rule.upf:39: fatal error: InvalidLevelShifterRule: port named '/topt/chk3/p' has no
required level shifter strategy, rule filter failed

dir.upf:39: fatal error: InvalidStrategyDirection: port named '/topt/chk3/p' has no
required level shifter strategy, port direction filter failed
```

Листинг 8. Примеры сообщений ошибок.
Listing 8. Tool error messages examples.

Все команды UPF из стандарта IEEE 1801-2018 поддерживаются. Дополнительно, все команды из стандартов IEEE 1801-2024 и более старых поддерживаются на уровне чтения спецификации для поддержки обратной совместимости. В дальнейшем планируется реализация полной поддержки и всех версий формата.

6. Тестирование на открытых проектах

Для оценки качества интерпретатора и модуля статического анализа были подготовлены синтетические примеры, как соответствующие стандарту UPF и проверяемым свойствам элементов подсистемы питания, так и нарушающие их. Всего было подготовлено порядка 830 синтетических тестов для проверки языковых конструкций и 20 полноценных проектов для проверки построенной модели, в том числе 2 позитивных, 17 негативных и один контрольный проект из стандарта IEEE 1801-2018 раздел E с разработанным для него HDL-кодом. Также было проведено тестирование на проектах с открытым исходным кодом. Ввиду малого количества открытых проектов с описанием подсистемы питания в формате UPF, в тестировании были использованы проекты openC910 [27], x-heep [28], 8b10b [29] и тестовый экземпляр, представленный в проекте OpenROAD [30].

В openC910 были обнаружены незначительные отличия от стандарта, такие как объявление комментариев в формате C/C++ вместо Tcl и передача команде `set_design_top` имени экземпляра модуля в качестве аргумента вместо имени самого модуля, которые могут быть следствием особенностей конкретного инструмента, используемого авторами в процессе разработки. В проектах x-heep и 8b10b были обнаружены несоответствия стандарту: отсутствие обязательной опции `-update` в команде `add_power_state` [31] и недокументированная опция `-control_ports` в команде `create_power_swtich` [32] соответственно. В экземпляре из проекта OpenROAD была обнаружена ошибка использования необъявленного объекта [33]. Эта ошибка была исправлена разработчиками [34].

Сравнительное тестирование проектов из листинга 4 и из стандарта UPF на разработанном инструменте, VCS и SpyGlass указало на главный недостаток проприетарных инструментов. Как VCS, так и SpyGlass имеют ограниченную поддержку стандарта формата UPF. Если первый инструмент способен поддерживать обработку UPF с небольшими ограничениями,

то в случае последнего наблюдается значительное отличие в синтаксисе опций и степени поддержки UPF-команд. Отсутствие поддержки некоторых команд и опций, как `set_variation`, `set_correlated`, `-model` для команды `add_power_state`, и синтаксические отличия сильно усложняют процесс разработки и тестирования подсистемы питания, требуя адаптацию описания подсистемы питания и соответствующего HDL-кода под конкретный инструмент. В отличие от них, разработанная нами система статического анализа поддерживает все команды стандартов UPF, включая версию 3.1. Поддержка стандарта IEEE 1801-2024 (версия 4.0) находится в процессе разработки.

7. Заключение

В рамках данной работы представлен инструмент для интерпретации и построения объектной модели описания подсистемы питания интегральных схем в формате UPF, а также статического анализа полученной модели на предмет корректности описания основных элементов данной подсистемы. В процессе разработки изучен стандарт формата UPF IEEE 1801, а также типичная структура и основные компоненты описания подсистемы питания с использованием этого формата. Рассмотрены существующие коммерческие и открытые инструменты, представлены их основные недостатки: высокая стоимость лицензии, разногласия со стандартом, отсутствие возможности проанализировать описание подсистемы питание. На основе этих исследований разработан подход для реализации отечественного инструмента.

Разработанный инструмент предоставляет полную поддержку формата UPF в соответствии со стандартом IEEE 1801-2018 и поддержку команд из других версий стандарта на уровне чтения спецификации. Реализована проверка синтаксических и семантических нарушений, ошибок корректности языковых конструкций, а также анализ стратегий изоляции и преобразования уровня напряжения. Инструмент был протестирован на синтетических тестах и открытых проектах.

В дальнейшей работе планируется расширение поддерживаемых команд до полной поддержки всех версий UPF и реализация большего количества проверок, включая стратегию сохранения состояния и блоки отключения питания, а также разработка методов анализа энергоэффективности построенной модели.

Также планируется сравнительное тестирование с коммерческими инструментами, предоставляющими возможность такого анализа, и продолжение взаимодействия с предприятиями-потребителями для получения обратной связи и внесения исправлений и дополнений.

Список литературы / References

[1]. Gibbons A. et al. Low power methodology manual, pp. 208-211. 2007.
[2]. Gourisetty V. et al. Low power design flow based on Unified Power Format and Synopsys tool chain. 2013 3rd Interdisciplinary Engineering Design Education Conference. IEEE, 2013, pp. 28-31.
[3]. Haripriya K. R., Somkuwar A., Kumre L. Low power checks in multi voltage designs. Wseas Transactions on Electronics. 2020, vol. 11, pp. 105-111.
[4]. A Practical Guide to Low-Power Design. User Experience with CPF [Electronic resource], 2009. Available at: http://www.si2.org/events_dir/2009/PowerForward/LowPowerGuide09232009/pfi_lpg_chapters/lpg_sect1_06052009.pdf, accessed: 19.03.2015.
[5]. Accellera Standards. Unified Power Format (UPF) Standard. Version 1.0. February 22, 2007.
[6]. IEEE Standard for Design and Verification of Low Power Integrated Circuits: Tech. Rep. 1801-2009. IEEE: Institute of Electrical and Electronics Engineers, 2009, Mar. Also known as Unified Power Format (UPF) standard.
[7]. IEEE Standard for Design and Verification of Low-Power Integrated Circuits: Standard 1801-2013. IEEE: Institute of Electrical and Electronics Engineers, 2013, May. Revision of IEEE Std 1801-2009.

[8]. IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems: Standard 1801-2015. IEEE: Institute of Electrical and Electronics Engineers, 2016, Mar. Revision of IEEE Std 1801-2013.

[9]. IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems: Standard 1801-2018 / IEEE: Institute of Electrical and Electronics Engineers, 2019, Mar. Revision of IEEE Std 1801-2015.

[10]. IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems: Standard 1801-2024. IEEE: Institute of Electrical and Electronics Engineers, 2025, Mar. Revision of IEEE Std 1801-2018.

[11]. Чуркин Я.А., Бучацкий Р.А., Китаев К.Н., Волохов А.Г., Долгодворов Е.В., Камкин А.С., Коцыняк А.М., Самоваров Д.О. Система статического анализа для языка описания аппаратуры SystemVerilog. Труды Института системного программирования РАН. 2025;37(1):7-40.

[12]. IEEE Standard for Liberty Variation Format (LVF), IEEE Std 1603-2023. IEEE, 2023.

[13]. Шашков А. С. Проектирование цифровых систем с пониженным энергопотреблением с применением технологии UPF-описания подсистемы питания. Информатика, 2016, №. 3, с. 90-104.

[14]. Tcl community. Tcl Tool Command Language, 2025. Available at: https://www.tcl-lang.org, accessed 01.11.2025.

[15]. Cadence Documentation. Available at: https://www.cadence.com/en_US/home/resources/documentation.html, accessed 01.11.2025.

[16]. Silicon Integration Initiative (Si2). Common Power Format Specification, Version 1.0. Austin, TX: Si2, Inc., March 2007.

[17]. SpyGlass: Early Design Analysis Tools for SoCs. Available at: https://www.synopsys.com/verification/static-and-formal-verification/spyglass.html, accessed 01.11.2025.

[18]. VCS: Functional Verification Solution. Available at: https://www.synopsys.com/verification/simulation/vcs.html, accessed 01.11.2025.

[19]. Questa One smart verification solution. Available at: https://eda.sw.siemens.com/en-US/ic/questa-one/, accessed 01.11.2025.

[20]. The OpenROAD Project. OpenROAD's unified application implementing an RTL-to-GDS. Available at: https://github.com/The-OpenROAD-Project/OpenROAD, accessed 01.11.2025.

[21]. OpenROAD. Available at: https://github.com/The-OpenROAD-Project/OpenROAD/issues/5617, accessed 01.11.2025.

[22]. Mike Popoloski, slang – SystemVerilog Language Services. Available at: https://github.com/MikePopoloski/slang, accessed 01.11.2025

[23]. IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language. IEEE Std 1800-2017. 1315 p. DOI: 10.1109/IEEESTD.2018.8299595.

[24]. РАН, Институт системного программирования. ReadCells компилятор Liberty, 2025. Доступно по ссылке: https://gitlab.ispras.ru/mvg/readcells, дата обращения: 16.06.2025.

[25]. Tcl community. Tcl C API, 2025. Available at: https://www.tcl-lang.org/man/tcl8.7/TclLib/, accessed 01.11.2025.

[26]. Schreiner, Henry. CLI11 Command Line Parser. Available at: https://github.com/CLIUtils/CLI11, accessed 01.11.2025.

[27]. OpenXuantie – OpenC910 Core, Available at: https://github.com/XUANTIE-RV/openc910, accessed 01.11.2025.

[28]. EPFL, ESL. x-heep: eXtendable Heterogeneous Energy-Efficient Platform. Available at: https://github.com/esl-epfl/x-heep, accessed 01.11.2025.

[29]. insularity1337. 8b10b encoder/decoder implementations. Available at: https://github.com/insularity1337/8b10b, accessed 01.11.2025.

[30]. OpenROAD UPF test example. Available at: https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/test/upf, accessed 01.11.2025.

[31]. x-heep issue: несоответствие языковым стандартам. Available at: https://github.com/esl-epfl/x-heep/issues/687, accessed 01.11.2025.

[32]. 8b10b issue: несоответствие языковым стандартам. Available at: https://github.com/insularity1337/8b10b/issues/2, accessed 01.11.2025.

[33]. OpenROAD issue. Available at: https://github.com/The-OpenROAD-Project/OpenROAD/issues/7363, accessed 01.11.2025.

[34]. OpenROAD pull request. Available at: https://github.com/The-OpenROAD-Project/OpenROAD/pull/7407, accessed 01.11.2025.

Информация об авторах / Information about authors

Арман Вагаршакович ЕГИАЗАРЯН – аспирант МФТИ, стажер-исследователь отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Arman Vagharshakovich YEGHIAZARYAN – postgraduate student at MIPT, research assistant at Compiler Technology department of ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Ян Андреевич ЧУРКИН – младший научный сотрудник отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации, языки описания аппаратуры.

Yan Andreevich CHURKIN – researcher at Compiler Technology department of ISP RAS. Research interests: static analysis, compiler technologies, optimizations, hardware description languages.

Илья Игоревич ЧЕРНЯВСКИХ – студент бакалавриата НИУ ВШЭ, лаборант отдела технологий программирования ИСП РАН. Научные интересы: статический анализ программ, верификация, проектирование цифровых устройств.

Ilya Igorevich CHERNYAVSKIKH – undergraduate student of at the NRU HSE, laboratory assistant at Programming Technologies department of ISP RAS. Research interests: static analysis, verification, digital hardware design.

Артем Михайлович КОЦЫНЯК – научный сотрудник отдела технологий программирования ИСП РАН. Научные интересы: формальные методы, компиляторные технологии, модели вычислений, языки программирования.

Artem Mikhailovich KOTSYNYAK – researcher at Software Engineering department of ISP RAS. Research interests: formal methods, compiler technologies, models of computation, programming languages.

Константин Николаевич КИТАЕВ – студент магистратуры МФТИ. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Konstantin Nikolaevich KITAEV – master's student at MIPT. Research interests: static analysis, compiler technologies, optimizations.

Рубен Артурович БУЧАЦКИЙ – кандидат технических наук, научный сотрудник отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Ruben Arturovich BUCHATSKIY – Cand. Sci. (Tech.), researcher at Compiler Technology department of ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Александр Сергеевич КАМКИН – кандидат физико-математических наук, ведущий научный сотрудник отдела технологий программирования ИСП РАН, ведущий научный сотрудник РЭУ им. Г.В. Плеханова. Научные интересы: формальные методы, синтез и верификация цифровой аппаратуры, гетерогенные компьютерные системы.

Alexander Sergeevich KAMKIN – Cand. Sci. (Phys.-Math.), leading researcher at Software Engineering department of ISP RAS, leading researcher at Plekhanov Russian University of Economics. Research interests: formal methods, synthesis and verification of digital hardware, and heterogeneous computer systems.

Андрей Владимирович КОРШУНОВ – кандидат технических наук, доцент, руководитель проектов АО «МНТЦ МИЭТ». Области научных интересов: методы и средства автоматизированного проектирования изделий микроэлектроники, включая сверхбольшие интегральные схемы (СБИС) и системы на кристалле (СнК). Ключевые направления исследований последних лет – проектирование сетей питания СБИС и СНК, энергоэффективное проектирование, разработка средств проектирования для производства фотошаблонов и технологии автоматизированного проектирования для перспективной ЭКБ.

Andrey Vladimirovich KORSHUNOV – Cand. Sci. (Tech.), projects lead, JSC "ISTC MIET". Research interests include methods and tools for automated design of microelectronics products, including very large-scale integrated circuits (VLSI) and systems-on-chip (SoC). Key research areas in recent years include the design of VLSI and SoC power networks, energy-efficient design, the development of design tools for photomask production, and automated design technologies for advanced electronic components.

Алексей Леонидович ПЕРЕВЕРЗЕВ – доктор технических наук, доцент, первый заместитель генерального директора АО «МНТЦ МИЭТ». Основное направление научной деятельности: проектирование, анализ и экспериментального исследования информационно-управляющих систем с целью улучшения их технических характеристик, включая исследование и разработку сложно-функциональных блоков, структурных и архитектурных решений, алгоритмов цифровой обработки сигналов и способов их технической реализации.

Alexey Leonidovich PEREVERZEV – Dr. Sci. (Tech.), First Deputy General Director, JSC "ISTC MIET". The main area of scientific activity: design, analysis and experimental research of information and control systems with the aim of improving their technical characteristics, including the research and development of complex functional blocks, structural and architectural solutions, digital signal processing algorithms and methods for their technical implementation.