НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
март–апрель 2026 Том 26 № 2 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
March–April 2026 Vol. 26 No 2 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2026-26-2-378-384
УДК 004.436.2

Анализ эффективности оптимизации поведенческих описаний аппаратуры в логических синтезаторах для FPGA

Леонид Вадимович Андрейченко[1], Александр Александрович Антонов[2], Павел Валерьевич Кустарев[3]✉

[1,2,3] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] buffer404@itmo.ru, https://orcid.org/0009-0002-6884-1765
[2] antonov@itmo.ru, https://orcid.org/0000-0002-4596-9275
[3] kustarev@itmo.ru✉, https://orcid.org/0000-0001-9326-0837

Аннотация

Введение. Рассмотрена проблема выбора оптимального уровня описания при проектировании цифровых схем на языках описания аппаратуры (Hardware Description Language, HDL). Актуальность исследования обусловлена тем, что ручная оптимизация дизайна для улучшения его характеристик часто вступает в противоречие с сохранением читабельности, конфигурируемости и сжатыми сроками на разработку. При этом конструкции, идиомы поведенческого описания аппаратуры, предлагаемые современными HDL, поддержаны оптимизирующими логическими синтезаторами в современных системах автоматизированного проектирования (САПР) программируемых вентильных матриц (Field-Programmable Gate Array, FPGA) с разным уровнем качества. Существующие наборы оценочных тестов (бенчмарков) зачастую фокусируются на интегральных показателях производительности, не позволяя детально оценить качество конкретных механизмов трансформации кода. В работе выполнен сравнительный анализ современных САПР FPGA и создание набора рекомендаций для эффективного использования HDL без ущерба качеству синтезированных решений. **Метод.** Исследование проводится в несколько этапов. На первом этапе проводится классификация известных методов оптимизации, применяемых при трансформации проекта на языке Verilog/SystemVerilog в структурное представление. На базе полученной классификации разрабатываются синтетические тесты, ориентированные на проверку оптимизаций, относящихся к конкретным классам. Данные тесты представляют собой пары эквивалентных по поведению описаний цифровых схем, одно из которых оптимизировано вручную, а во второе введена избыточность посредством использования конкретных конструкций языка Verilog/SystemVerilog и/или идиом поведенческого описания. Разница в характеристиках данных реализаций позволяет сделать вывод об уровне эффективности САПР в применении соответствующих оптимизаций. **Основные результаты.** Предложена трехуровневая классификация оптимизаций поведенческих описаний аппаратуры. В рамках данной классификации разработан тестовый набор из 19 тестов, выборочно направленных на оценку оптимизаций, принадлежащих разным уровням предложенной классификации. Данные тесты применены для ряда современных САПР FPGA (Vivado, Quartus, Yosys). Демонстрируется последовательное снижение эффективности применения идиом кодирования при их приближении к поведенческому уровню описания, причем с повышением уровня усиливаются различия между разными САПР. Существенное падение качества результата наблюдается при оценке оптимизации поведения на уровне множественных тактов синхросигнала. На основе полученных результатов сформулированы практические рекомендации для разработчиков цифровой аппаратуры по стилю написания HDL-кода, позволяющие максимально эффективно использовать возможности конкретных синтезаторов. **Обсуждение.** Результаты работы позволяют выделить общее «ядро» HDL и идиом описания логики без ущерба качеству синтезируемой аппаратуры, а также определить перспективные направления дальнейшего совершенствования синтезаторов и HDL.

Ключевые слова

САПР, FPGA, HDL, Verilog, SystemVerilog, логический синтез, поведенческий синтез

Ссылка для цитирования: Андрейченко Л.В., Антонов А.А., Кустарев П.В. Анализ эффективности оптимизации поведенческих описаний аппаратуры в логических синтезаторах для FPGA // Научно-технический вестник информационных технологий, механики и оптики. 2026. Т. 26, № 2. С. 378–384. doi: 10.17586/2226-1494-2026-26-2-378-384

Analysis of the effectiveness of optimizing behavioral descriptions of hardware in logic synthesizers for FPGA

Leonid V. Andreichenko[1], Alexander A. Antonov[2], Pavel V. Kustarev[3]✉

[1,2,3] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] buffer404@itmo.ru, https://orcid.org/0009-0002-6884-1765
[2] antonov@itmo.ru, https://orcid.org/0000-0002-4596-9275
[3] kustarev@itmo.ru✉, https://orcid.org/0000-0001-9326-0837

Abstrac

This article discusses the problem of choosing the optimal description level when designing digital circuits in Hardware Description Language (HDL). The relevance of this research is due to the fact that manual optimization of design to improve its characteristics often conflicts with maintaining readability, configurability, and tight development deadlines. At the same time, the constructs, idioms, and practices of abstract (behavioral) hardware description offered by modern HDLs are supported by optimizing logic synthesizers in modern CAD systems for Field-Programmable Gate Array (FPGA) with varying levels of quality. Existing sets of evaluation tests (benchmarks) often focus on integrated performance indicators, preventing a detailed assessment of the effectiveness of specific code transformation mechanisms. The main goal of this work is to conduct a comparative analysis of modern FPGA CAD systems and create a set of recommendations for the effective use of HDL without compromising the quality of synthesized solutions. The research is conducted in several stages. The first stage involves classification of known optimization methods used in the transformation of Verilog/SystemVerilog-designs into a structural representation. Based on the resulting classification, synthetic tests are developed to verify optimizations, related to specific classes. These tests consist of pairs of behaviorally equivalent designs, one of which is optimized manually, while the other has redundancy due to the use of abstract or inflated Verilog/SystemVerilog language constructs and/or behavior description patterns. The “gap” in the characteristics of these implementations allows us to draw conclusions about the level of CAD efficiency in the application of specific optimizations. A three-level classification of optimizations of behavioral descriptions of hardware is proposed. Within the framework of this classification, a test package of 19 tests has been developed, selectively aimed at evaluating optimizations belonging to different levels of the proposed classification. These tests have been applied to a number of modern FPGA CAD systems (Vivado, Quartus, Yosys). A consistent decrease in the effectiveness of coding patterns is demonstrated as they approach the behavioral level of description, with the differences between different CAD systems increasing as the level increases. A significant drop in quality of results is observed when evaluating behavior optimization at the level of multiple clock cycles. Based on the results obtained, practical recommendations are formulated for developers of digital equipment on the style of writing HDL code, allowing the most effective use of the capabilities of specific synthesizers. The results of the work allow us to identify the common “core” of HDL and logic description patterns without compromising the quality of the synthesized equipment as well as to determine promising directions for further improvement of synthesizers and HDL.

Keywords

CAD, FPGA, HDL, Verilog, SystemVerilog, logic synthesis, behavioral synthesis

For citation: Andreichenko L.V., Antonov A.A., Kustarev P.V. Analysis of the effectiveness of optimizing behavioral descriptions of hardware in logic synthesizers for FPGA. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2026, vol. 26, no. 2, pp. 378–384 (in Russian). doi: 10.17586/2226-1494-2026-26-2-378-384

Введение

Логический синтез в современных маршрутах проектирования цифровых схем остается, как правило, де-факто первым формальным этапом, на котором намерения разработчика, выраженные в описании на уровне регистровых передач (Register Transfer Level, RTL), переводятся в конкретную структурную реализацию. При этом исходный текст на языке описания аппаратуры (Hardware Description Language, HDL), являясь полностью синтезируемым и корректно описывающим поведение цифровой схемы, не всегда представляет эффективную, оптимизированную аппаратную реализацию (структуру). Проблемой исчерпывающей оптимизации в рамках средств логического синтеза является сложность задачи: оптимизация логики и последовательностного поведения на уровне RTL трудная NP-задача (Nondeterministic Polynomial time), для нетривиальных схем полный перебор трансформаций невозможен, поэтому промышленные инструменты вынуждены опираться на набор эвристик и ограниченных по глубине и области видимости преобразований. С другой стороны, оптимизация непосредственно на уровне исходного RTL-кода может входить в конфликт с требованиями по читаемости, конфигурируемости, простоте автоматизированной генерации, срокам разработки и другим причинам. Как следствие, для эффективного использования актуальных маршрутов проектирования оказывается необходимо иметь представление о том, какие именно классы оптимизаций встроены в соответствующие инструменты, с какой интенсивностью они применяются и, как следствие, какие идиомы HDL-кодирования являются допустимыми без существенных потерь качества аппаратной реализации. Анализ и рекомендации к применению этих практик является целью настоящей работы.

Методы оценки логического синтеза

Оценка эффективности методов и инструментов логического синтеза является актуальным и широко разрабатываемым вопросом. Первой решенной задачей были тесты для проверки работоспособности двухуровневой комбинационной логики устройств типа про-

граммируемые логические матрицы [1]. Следующим шагом стали наборы тестов MCNC и LGSynth'91 [2], в рамках которых к программируемым логическим матрицам добавились многоуровневые BLIF-нетлисты и последовательностные схемы, ориентированные на полноценный логический синтез с регистрами и несколькими уровнями логики. В начале 90-х прошлого века появились наборы тестов IWLS'91 и IWLS'93 [3]: во втором случае те же схемы были дополнительно синтезированы через проприетарную систему автоматизированного проектирования (САПР) и преобразованы в формат EDIF, что породило избыточные нетлисты, хорошо выступающие в качестве стресс-теста для оптимизаторов. Параллельно в тестовую практику синтеза вошли тестовые наборы ISCAS'85/'89 [4, 5]. Они использовались для оценки синтеза на реальных и относительно компактных схемах.

Далее увеличились размеры и сложность проектов и появились программируемые вентильные матрицы (Field-Programmable Gate Array, FPGA). Набор тестов IWLS 2005 [6] объединил крупные модули из каталога открытых ядер OpenCores и другие фрагменты процессорных и периферийных блоков, использующиеся в промышленности. Для FPGA отдельную роль сыграли QUIP-наборы и примеры из документации на FPGA Altera/Intel[1]: HDL-проекты, через которые можно измерять поведение конкретного промышленного синтезатора на типичных пользовательских конструкциях.

Затем исследования были направлены на увеличение размера и количества тестовых наборов. Набор комбинационных тестов EPFL [7] предназначен для оценки качества оптимизации больших комбинационных схем. Современные исследования расширяют применение методов машинного обучения [8–10] в задачах синтеза — от использования крупных HLS-наборов [11] для предсказания производительности, энергопотребления, площади кристалла и автоматического подбора оптимизаций — до интеллектуальной настройки маршрутов САПР и параметров синтеза. Также развивают направление приблизительного логического синтеза [12, 13], ориентированного на упрощение схем при допустимом снижении точности.

Стоит отметить, что на данный момент нет единого стандартного набора тестов для сравнения синтезаторов. Все перечисленные наборы направлены в большей степени на валидацию синтезаторов и позволяют произвести интегральную сравнительную оценку качества результатов без привязки к конкретным разновидностям цифровой логики (арифметические последовательности, управление конечными автоматами, контроль потока и пр.) и удобным для их описания идиомам HDL. Как следствие, остается неопределенным, в каких именно случаях возможно приоритезировать читабельность и конфигурируемость HDL-описаний, а в каких работу по оптимизации необходимо выполнять предварительно, силами разработчика. Для получения ответов на эти вопросы в настоящей работе предлагается следующий подход:

1) разработать тестовый набор микробенчмарков, каждый из которых представляет собой пару модулей: эталонный (Gold) и экспериментальный (Challenge): Gold-вариант — вручную оптимизированная реализация; Challenge-вариант — функционально эквивалентное, но заведомо избыточное описание той же функции, полученное путем применения конкретного типа конструкций HDL и/или приема описания;
2) выполнить логический синтез обоих модулей в нескольких современных инструментах при разных режимах загрузки и стратегиях оптимизации;
3) измерить и сопоставить разницу по утилизации ресурсов между Gold и Challenge, тем самым эмпирически оценив «глубину» реализованных оптимизаций на уровне RTL.

Разработка селективных тестов для оценки синтезаторов

Под поведенческим описанием в настоящей работе понимается высокоуровневое описание, абстрагированное от особенностей аппаратного базиса, реализующих поведение устройства.

Поведенческое описание системы фокусируется на поведении системы, а не на стиле описания, и предполагает возможность описания поведения системы с возможностью свободного комбинирования синтезируемых конструкций HDL, включая элементарные операции, установку структурных связей и объединение операций в процедурные блоки с произвольным потоком управления. Поведенческое описание дает широкий простор для оптимизации в рамках синтеза аппаратуры.

Соответственно, функции оптимизации предлагается распределить по следующим уровням описания аппаратуры: уровень тракта данных; уровни операций и потока управления; уровень многотактовых вычислений.

На уровне тракта данных подход направлен на логическую (булевую) оптимизацию. Данная оптимизация, как правило, выполняется на ранних этапах и мало зависит от выбранной технологии реализации аппаратуры.

На уровне операций синтезатор манипулирует арифметическими блоками как цельными и неделимыми и учитывает специфику целевой технологии, например, возможности конкретной архитектуры FPGA по реализации сумматоров или цепочек переноса.

Оптимизация потоков управления связывает синтезатор с классическим программным компилятором, например, при удалении недостижимых ветвей логики и минимизации таблиц переходов. На стыке управления и последовательностной логики находятся оптимизации конечных автоматов (Finite State Machine, FSM), где критически важным является выбор эффективного кодирования состояний, синтез логики переходов и выходов.

Многотактовые трансформации принадлежат к следующей по сложности категории, так как предусматривают оптимизацию поведения в течение множествен-

[1] Altera. Advanced Synthesis Cookbook. 2011 [Электронный ресурс]. Режим доступа: https://fpgacpu.ca/multiport/stx_cookbook.pdf, свободный. Яз. англ. (дата обращения: 23.01.2026).

ных циклов тактового сигнала (например, конвейерных трактов данных). В данном случае требуется строгое формальное подтверждение эквивалентности при перераспределении вычислений во времени.

Перечисленные уровни описания аппаратуры в целом покрывают все синтезируемое подмножество языков HDL, являющееся, как правило, относительно небольшим по сравнению с конструкциями для описания тестовых окружений и виртуальных прототипов [14].

Для простоты рассмотрим только синхронные схемы, к которым относится большинство разрабатываемой аппаратной логики на сегодняшний день.

В табл. 1 представлены описания тестов разработанного набора, охватывающих указанные уровни абстракции. Каждый тест реализует специфическую стратегию описания на языке HDL, что обеспечивает селективность оценки исследуемых методов оптимизации. Для каждого тестового уровня определены ключевые ресурсы, на которые влияет тест.

Результаты экспериментов

Для оценки эффективности оптимизаций были выбраны три распространенных САПР FPGA, представляющие доминирующие проприетарные платформы (Intel Quartus, AMD/Xilinx Vivado) и наиболее развитый маршрут проектирования с открытым исходным кодом (Yosys). Во всех случаях использовались версии 2025 года.

Эксперименты выполнялись для двух современных и широко используемых семейств FPGA. Для Vivado, а также для синтеза на базе Yosys, в качестве целевого устройства была выбрана FPGA AMD/Xilinx Artix-7 XC7A100T-CSG324-1. Для Quartus в качестве целевого устройства использовалась FPGA Intel Cyclone V 5CGXFC7C7F23C8.

Поскольку основной метрикой выступает минимизация аппаратных затрат, во всех трех САПР применялись настройки, ориентированные на максимальную оптимизацию по площади и аппаратным ресурсам. Результаты экспериментов приведены в табл. 2. Для каждого теста в ячейке указана количественная утилизация контрольного ресурса, согласно табл. 1.

Анализ качества оптимизаций

Отношение эталонного (минимального) количества затраченных ресурсов в Gold-варианте к Challenge-варианту визуализировано на рисунке. Высота диаграммы характеризует то, насколько Challenge прибли-

Таблица 1. Разработанные тесты

Table 1. Developed tests

Тест	Паттерн оптимизации	Контрольный ресурс
Уровень тракта данных		
DP1	Усложненные последовательности булевых операций	LUT + CARRY
DP2	Многоразрядные сигналы, где только часть битов используется для вычисления результата	LUT + CARRY
DP3	Мертвая логика, которая не влияет на выходные данные	LUT + CARRY
DP4	Последовательная обработка набора битов	LUT + CARRY
DP5	Распознавание константных битов в индексе декодера	FF
Уровень операций		
OL1	Усложненные последовательности арифметических операций	LUT + CARRY
OL2	Возведение в степень двойки	LUT + CARRY
OL3	Наличие повторяющихся подвыражений	LUT + CARRY
OL4	Инкремент на каждом такте	LUT + CARRY
OL5	Операции выполняются над константами, которые не зависят от входов	LUT + CARRY
CF1	Ветки с эквивалентным телом	LUT + CARRY
CF2	Использование варианта по умолчанию (default) при полном переборе всех его состояний	LUT + CARRY
CF3	Использование оператора выбора с маскированием (casex/z)	LUT + CARRY
FSM1	Наличие недостижимого состояния	LUT + CARRY
FSM2	Наличие состояний с эквивалентной логикой	LUT + CARRY
Уровень многотактовых вычислений		
FSM3	Наличие константной зависимости между состояниями	LUT + CARRY
MC1	Константные условия	LUT + CARRY
MC2	Многотактовая функциональная избыточность	LUT + CARRY
MC3	Сохранение переменных вместо их результата	FF

Примечание: DP — уровень тракта данных; OL — уровень операций; CF — уровень потока управления; MC — уровень многотактовых вычислений; LUT + CARRY — таблица преобразований и цепь переноса; FF — триггер.

Таблица 2. Результаты тестирования — количество утилизированных ячеек FPGA, ед.
Table 2. Test results — the number of utilized FPGA-cells

Тест	Платформы					
	Vivado		Yosys		Quartus	
	Микробенчмарки					
	Gold	Challenge	Gold	Challenge	Gold	Challenge
Уровень тракта данных						
DP1	2 061	2 086	3 981	3 981	4 112	4 112
DP2	2 096	2 096	2 981	2 993	2 122	2 122
DP3	8 205	8 205	8 207	8 207	8 336	8 336
DP4	897	929	1 162	1 111	1 292	1 292
DP5	132	132	2 053	16 389	2 053	2 053
Уровень операций						
OL1	4 292	56 620	6 906	38 190	4 240	8 595
OL2	2 023	2 023	1 817	1 817	3 920	3 920
OL3	14 714	14 714	12 791	12 793	16 527	16 527
OL4	140	141	46	8 146	4 112	12 560
OL5	4 560	8 034	2 125	4 078	4 179	4 174
CF1	4 173	4 173	2 992	3 090	4 240	4 240
CF2	12 456	12 456	10 063	10 063	9 956	9 956
CF3	14 534	14 534	8 265	8 298	16 443	16 443
FSM1	10 360	10 023	4 175	4 174	12 433	12 434
FSM2	10 290	10 301	4 177	4 178	14 481	12 467
Уровень многотактовых вычислений						
FSM3	77	139	82	328	81	81
MC1	9 746	9 862	6 189	6 221	12 368	12 560
MC2	125	142	94	113	80	144
MC3	508	755	4 224	8 256	4 224	8 256

Примечание: оранжевый цвет ячеек — эталонное значение теста; зеленый — тест пройден; красный — тест не пройден.

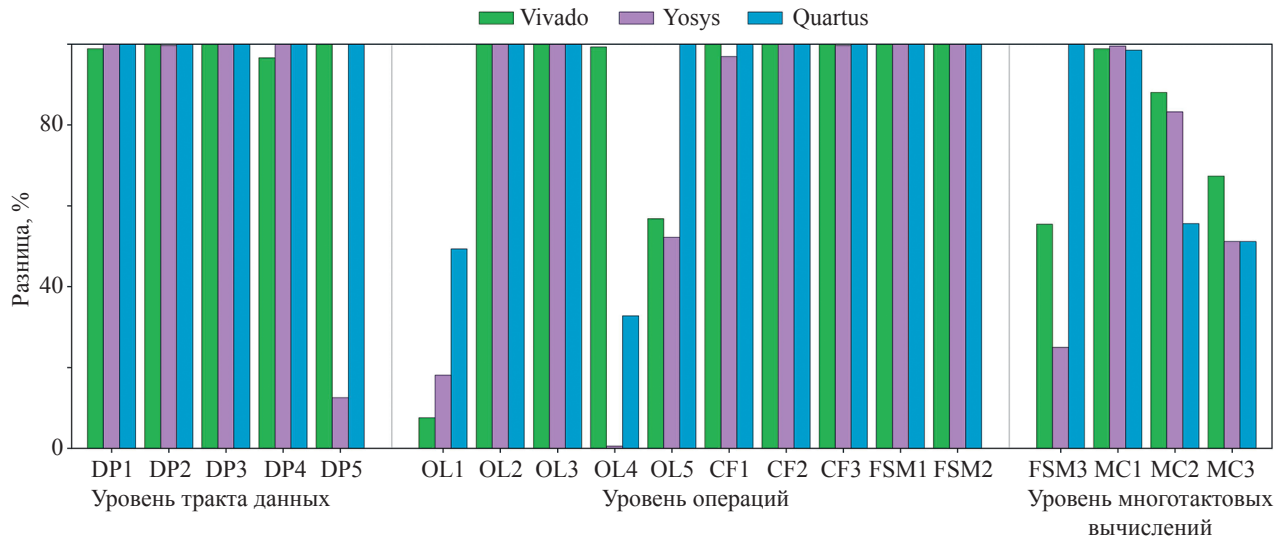

Рисунок. Разница утилизации ресурсов в тестах
Figure. The difference in resource utilization in tests

зился по ресурсам к Gold и достигает 100 % в случае совпадения, т. е. полностью успешного применения оптимизации.

Наилучший результат наблюдается в тестах, где оптимизация сводится к локальным преобразованиям в пределах комбинаторного конуса: булева алгебра, устранение мертвого кода, упрощение ветвлений и явная минимизация таблиц case (таблиц ветвлений). В таких случаях синтезатору не нужно анализировать зависимости дизайна во времени — достаточно классических проходов упрощения логики. По этой причине при тестировании оптимизаций на уровнях тракта данных и потока управления большинство Challenge-версий после синтеза практически неотличимы от Gold-версий — инструменты уверенно находят эквивалентные ветки, неиспользуемые сигналы и удаляют лишнюю логику.

Наиболее выразительная и устойчивая разница проявляется для оптимизаций уровней многотактовых вычислений и частично уровня операций, что закономерно. На уровне операций синтезатор работает уже не только с булевыми примитивами, но и с крупными узлами (сумматоры, каскады сложений/вычитаний, выражения с общими подвыражениями). В этих случаях результат в большей степени зависит от того, насколько агрессивно инструмент выполняет алгебраические преобразования, поиск общих подвыражений, переиспользование аппаратных блоков. Именно поэтому в тесте «уровень операций» результаты менее однозначны: одни паттерны «схлопываются» практически всегда, другие — дают существенную остаточную разницу. Такой профиль обычно означает, что часть оптимизаций инструмент выполняет на уровне простых упрощений, но не делает глубокой нормализации арифметики или ограничен в переиспользовании из-за внутренних эвристик и ограничений по временным характеристикам или структуре межсоединений элементов, особенно это видно в тесте OL1.

Для уровня многотактовых вычислений разница особенно показательна: здесь оптимизация требует не только комбинаторной эквивалентности, а фактически перераспределения вычислений во времени. Это также требует доказательства, что манипуляции над промежуточными значениями не приводят к изменениям в поведении, наблюдаемом извне цифровой схемы.

Обсуждение

Полученные результаты позволяют сформулировать практический вывод: качество оптимизации в логическом синтезе определяется не только выбранными опциями САПР, но и тем, насколько исходный HDL-код выражен в узнаваемых для синтезатора идиомах. В тех классах тестов, где Challenge-описание строилось из стандартных, широко поддерживаемых конструкций, синтезаторы устойчиво приводят схему к форме, близкой к Gold. Напротив, когда поведенческое описание выполнено на высоком уровне, то это заметно усложнило работу синтезатора и оптимизация требовала доказательства свойств системы во времени или глубокой нормализации арифметики, разница между Gold и Challenge чаще сохранялась и сильнее зависела от конкретного инструмента. Этот вывод наглядно виден на рисунке. Чем больший уровень абстракции поведенческого описания используется, тем труднее его оптимизировать, приблизив к Gold.

Перечислим идиомы, которые часто ухудшают оптимизацию:

— скрытая/косвенная недостижимость, когда избыточность ветки выполнения следует не из логики комбинационной схемы на текущем такте, а из возможной предыстории поведения в течение множества тактов;
— интеграция логики, избыточность которой следует из порядка прохождения транзакций через аппаратную структуру в течение множества тактов синхросигнала;
— неоптимальное планирование вычислений на протяжении множества тактов синхросигнала, ведущее к удлинению жизненного цикла данных и появлению избыточных регистров в дизайне;
— глубокие арифметико-логические вычисления, выполняемые в пределах одного тактового интервала, относятся к числу наиболее сложных объектов для оптимизации;
— «анти-идиоматический» код, который затрудняет распознавание типовых шаблонов: избыточные промежуточные регистры, искусственные зависимости, сложные комбинации блокирующих/неблокирующих присваиваний.

Таким образом, использование данных практик нежелательно при применении актуальных САПР FPGA, однако их поддержка может стать целью при создании перспективных САПР либо HDL, в более явном виде отражающих намерения разработчика при конструировании сложной аппаратной микроархитектуры.

Заключение

Экспериментальное исследование показало, что свободное комбинирование всех конструкций языков описания аппаратуры, даже при условии их синтезируемости, не обеспечивает высокого качества оптимизации.

Современные системы автоматизированного проектирования позволяют стабильно качественно оптимизировать поведенческие описания на уровне тракта данных и частично уровня операций. Если приоритетом являются высокие характеристики цифровой схемы, целесообразно самостоятельно проводить выборочную оптимизацию на уровне операций (последовательности арифметических операций) и все оптимизации на уровне многотактовых вычислений. Данные классы оптимизации можно считать перспективными при разработке новых алгоритмов.

Стабильность полученных результатов позволяет рекомендовать разработанный набор тестов для выбора и оценки качества систем автоматизированного проектирования.

Литература

1. Wei R.-S., Sangiovanni-Vincentelli A. PLATYPUS: A PLA test pattern generation tool // Proc. of the 22nd ACM/IEEE Design Automation Conference. 1985. P. 197–203. https://doi.org/10.1109/dac.1985.1585935
2. Krakow W.T. The MCNC design initiative program // Proc. of the 8th University/Government/Industry Microelectronics Symposium. 1989. P. 35–39. https://doi.org/10.1109/UGIM.1989.37294
3. McElvain K. IWLS'93 benchmark set: version 4.0 // Distributed as part of the MCNC International Workshop on Logic Synthesis' 93 benchmark distribution. 1993.
4. Hansen M.C., Yalcin H., Hayes J.P. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering // Proc. of the IEEE Design & Test of Computers. 1999. V. 16. N 3. P. 72–80. https://doi.org/10.1109/54.785838
5. Brglez F., Bryan D., Kozminski K. Notes on the ISCAS'89 Benchmark Circuits. Technical report // Microelectronics Center of North Carolina. 1989. P. 1929–1934.
6. Albrecht C. IWLS 2005 benchmarks // International Workshop for Logic Synthesis (IWLS). 2005. V. 9.
7. Amarú L., Gaillardon P.E., De Micheli G. The EPFL combinational benchmark suite // Hypotenuse. 2015. V. 256. N 128. P. 214335.
8. Abi-Karam S., Sarkar R., Seigler A., Lowe S., Wei Z., Chen H., et al. HLSFactory: a framework empowering high-level synthesis datasets for machine learning and beyond // Proc. of the ACM/IEEE 6th Symposium on Machine Learning for CAD (MLCAD). 2024. P. 1–9. https://doi.org/10.1109/MLCAD62225.2024.10740213
9. Goswami P., Shahshahani M., Bhatia D. MLSBench: A synthesizable dataset of HLS designs to support ML based design flows // Proc. of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2020. P. 312. https://doi.org/10.1145/3373087.3375378
10. Wanna A., Chen H., Hao C. ForgeBench: a machine learning benchmark suite and auto-generation framework for next-generation HLS tools // arXiv. 2025. arXiv:2504.15185. https://doi.org/10.48550/arXiv.2504.15185
11. Bai Y., Cong J., Hu Z., Qin Z., Sohrabizadeh A., Sun Y. Towards a comprehensive benchmark for high-level synthesis targeted to FPGAs // Proc. of the 37th Conference on Neural Information Processing Systems. 2023. V. 36. P. 45288–45299. https://doi.org/10.52202/075280-1962
12. Scarabottolo I., Ansaloni G., Constantinides G.A., Pozzi L., Reda S. Approximate logic synthesis: a survey // Proceedings of the IEEE. 2020. V. 108. N 12. P. 2195–2213. https://doi.org/10.1109/JPROC.2020.3014430
13. Hu H., Cai S. OPTDTALS: approximate logic synthesis via optimal decision trees approach // arXiv. 2024. arXiv:2408.12304. https://doi.org/10.48550/arXiv.2408.12304
14. Lööw A. The simulation semantics of synthesisable Verilog // Proc. of the ACM on Programming Languages. 2025. V. 9. N OOPSLA1. P. 1295–1320. https://doi.org/10.1145/3720484

References

1. Wei R.-S., Sangiovanni-Vincentelli A. PLATYPUS: A PLA test pattern generation tool. *Proc. of the 22nd ACM/IEEE Design Automation Conference*, 1985, pp. 197–203. https://doi.org/10.1109/dac.1985.1585935
2. Krakow W.T. The MCNC design initiative program. *Proc. of the 8th University/Government/Industry Microelectronics Symposium*, 1989, pp. 35–39. https://doi.org/10.1109/UGIM.1989.37294
3. McElvain K. IWLS'93 benchmark set: version 4.0. *Distributed as part of the MCNC International Workshop on Logic Synthesis' 93 benchmark distribution*, 1993.
4. Hansen M.C., Yalcin H., Hayes J.P. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *Proc. of the IEEE Design & Test of Computers*, 1999, vol. 16, no. 3, pp. 72–80. https://doi.org/10.1109/54.785838
5. Brglez F., Bryan D., Kozminski K. Notes on the ISCAS'89 Benchmark Circuits. Technical report. *Microelectronics Center of North Carolina*, 1989, pp. 1929–1934.
6. Albrecht C. IWLS 2005 benchmarks. *International Workshop for Logic Synthesis (IWLS)*, 2005, vol. 9.
7. Amarú L., Gaillardon P.E., De Micheli G. The EPFL combinational benchmark suite. *Hypotenuse*, 2015, vol. 256, no. 128, pp. 214335.
8. Abi-Karam S., Sarkar R., Seigler A., Lowe S., Wei Z., Chen H., et al. HLSFactory: a framework empowering high-level synthesis datasets for machine learning and beyond. *Proc. of the ACM/IEEE 6th Symposium on Machine Learning for CAD (MLCAD)*, 2024, pp. 1–9. https://doi.org/10.1109/MLCAD62225.2024.10740213
9. Goswami P., Shahshahani M., Bhatia D. MLSBench: A synthesizable dataset of HLS designs to support ML based design flows. *Proc. of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020, pp. 312. https://doi.org/10.1145/3373087.3375378
10. Wanna A., Chen H., Hao C. ForgeBench: a machine learning benchmark suite and auto-generation framework for next-generation HLS tools. *arXiv*, 2025. arXiv:2504.15185. https://doi.org/10.48550/arXiv.2504.15185
11. Bai Y., Cong J., Hu Z., Qin Z., Sohrabizadeh A., Sun Y. Towards a comprehensive benchmark for high-level synthesis targeted to FPGAs. *Proc. of the 37th Conference on Neural Information Processing Systems*, 2023, vol. 36, pp. 45288–45299. https://doi.org/10.52202/075280-1962
12. Scarabottolo I., Ansaloni G., Constantinides G.A., Pozzi L., Reda S. Approximate logic synthesis: a survey. *Proceedings of the IEEE*, 2020, vol. 108, no. 12, pp. 2195–2213. https://doi.org/10.1109/JPROC.2020.3014430
13. Hu H., Cai S. OPTDTALS: approximate logic synthesis via optimal decision trees approach. *arXiv*, 2024. arXiv:2408.12304. https://doi.org/10.48550/arXiv.2408.12304
14. Lööw A. The simulation semantics of synthesisable Verilog. *Proc. of the ACM on Programming Languages*, 2025, vol. 9, no. OOPSLA1, pp. 1295–1320. https://doi.org/10.1145/3720484

Авторы

Андрейченко Леонид Вадимович — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0002-6884-1765, buffer404@itmo.ru

Антонов Александр Александрович — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 56949952500, https://orcid.org/0000-0002-4596-9275, antonov@itmo.ru

Кустарев Павел Валерьевич — кандидат технических наук, декан, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 35317916600, https://orcid.org/0000-0001-9326-0837, kustarev@itmo.ru

Authors

Leonid V. Andreichenko — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0002-6884-1765, buffer404@itmo.ru

Alexander A. Antonov — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 56949952500, https://orcid.org/0000-0002-4596-9275, antonov@itmo.ru

Pavel V. Kustarev — PhD, Dean, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 35317916600, https://orcid.org/0000-0001-9326-0837, kustarev@itmo.ru

Статья поступила в редакцию 16.01.2026
Одобрена после рецензирования 26.02.2026
Принята к печати 19.03.2026

Received 16.01.2026
Approved after reviewing 26.02.2026
Accepted 19.03.2026