Семинар "Педагогика ИИ"

Дата и время: 23.07.2025, 10.00

Место проведения: НИЦ "Курчатовский институт" - НИИСИ, г. Москва, Нахимовский просп., д. 36, к.1., кабинет 1006.

Участники: Олег Вадимович Глухов (НИУ «МЭИ»), Константин Александрович Петров (НИЦ "Курчатовский институт" - НИИСИ), Александр Павлович Еремеев (НИУ «МЭИ»), Павел Юрьевич Анучин (НИУ «МЭИ»), Сергей Викторович Вишняков (НИУ «МЭИ»), Евгений Александрович Волошин (НИУ «МЭИ»), Шамиль Алиевич Оцоков (НИУ «МЭИ»), Эдуард Артурович Челышев (НИУ «МЭИ»), Вадим Игоревич Лазарев (НИУ «МЭИ»), Владимир Владимирович Чистяков (НИУ «МЭИ»), Сергей Иванович Аряшев (НИЦ "Курчатовский институт" - НИИСИ), Никита Артемович Гревцев (НИЦ "Курчатовский институт" - НИИСИ), Алексей Дмитриевич Манеркин (НИЦ "Курчатовский институт" - НИИСИ), Никита Владимирович Желудков (НИЦ "Курчатовский институт" - НИИСИ), Яков Михайлович Карандашев (НИЦ "Курчатовский институт" - НИИСИ), Борис Владимирович Крыжановский (НИЦ "Курчатовский институт" - НИИСИ), Борис Евгеньевич Евлампиев (НИЦ "Курчатовский институт" - НИИСИ), Евгений Константинович Эмин (НИЦ "Курчатовский институт" - НИИСИ), Иван Михайлович Косарев (НИЦ "Курчатовский институт" - НИИСИ).

Повестка:

1) «Применение инструментов ИИ в маршруте проектирования микросхем», три доклада:
«Машинное обучение в функциональной верификации на системном уровне»
«Машинное обучение в функциональной верификации на блочном уровне»
«Машинное обучение при оценке длины цепей на этапе размещения ячеек»;

2) Использование ИИ в образовании для подготовки кадров

Олег Вадимович Глухов:
Коллеги, рад всех приветствовать на третьем семинаре, в этот раз проводим его в НИИСИ.
Напомню наш формат: это открытый семинар, в котором участвуют представители заинтересованных организаций. Семинар проводится на регулярной основе. Ведётся аудиозапись, после чего готовится стенограмма.
Предлагаю по традиции представиться. Особенно это касается новых участников.

Меня зовут Олег Вадимович Глухов. Я ассистент кафедры Радиотехнических систем МЭИ, и являюсь организатором этого семинара. Занимался в различных областях и сферах применения ИИ. Одна из целей проведения семинара: обобщить актуальный опыт применения ИИ для решения прикладных задач и популяризировать его.
Прошу всех участников представиться — пойдём против часовой стрелки.

Константин Александрович Петров:
Меня зовут Петров Константин Александрович. Я представляю НИИ системных исследований (НИИСИ). Аккумулирую научные компетенции организации, в том числе, в области ИИ. Я являюсь доцентом кафедры Электроники НИЯУ МИФИ, поэтому меня интересуют данные технологии также с точки зрения работы со студентами.

Сергей Иванович Аряшев:
Меня зовут Сергей Иванович Аряшев. Я заместитель директора НИИСИ, руководитель дизайн центра. Мне интересно использовать ИИ в маршрутах разработки доверенных ЭКБ.

Шамиль Алиевич Оцоков:
Добрый день, коллеги. Меня зовут Шамиль Алиевич Оцоков. Я доцент кафедры Вычислительных машин, систем и сетей МЭИ. Занимался искусственным интеллектом, решал задачи, связанные с оценкой деятельности кафедры, а также другие прикладные задачи.

Александр Павлович Еремеев:
Меня зовут Александр Павлович Еремеев. Я профессор кафедры Прикладной математики и искусственного интеллекта МЭИ, лауреат премии Президента Российской Федерации в области образования за создание и внедрение учебно-методического комплекса, член Научного совета Российской ассоциации искусственного интеллекта (РАИИ).

Никита Артемович Гревцев:
Меня зовут Никита Артемович Гревцев. Я представляю на этом семинаре сектор Верификации и тестирований. С меня в НИИСИ началось применение машинного обучения в области верификации. Буду рад поделиться полученным опытом.

Алексей Дмитриевич Манеркин:
Добрый день, меня зовут Алексей Дмитриевич Манеркин. Я представляю сектор Верификации и тестирований НИИСИ, также являюсь ассистентом кафедры Радиоэлектронные системы и устройства МГТУ им. Баумана. Сфера моих интересов – это применение машинного обучения искусственного интеллекта в верификации на блочном уровне. Буду рад поделиться наработками.

Никита Владимирович Желудков:
Доброе утро, коллеги. Меня зовут Никита Желудков, я представляю НИИСИ, работаю в отделе Поиска, проектирования и синтеза микросхем. Сфера моих научных интересов также связана с применением ИИ и методов машинного обучения в маршруте проектирования.

Борис Евгеньевич Евлампиев:
Здравствуйте, коллеги. Меня зовут Борис Евгеньевич Евлампиев. Я заведующий отделом Синтеза и топологического проектирования НИИСИ РАН. Мы пытаемся использовать наработки Никиты Артемовича и его коллектива, пока что мало опыта.

Яков Михайлович Карандашев:
Добрый день, коллеги. Меня зовут Яков Михайлович Карандашев. Я представляю центр Оптико-нейронных технологий НИИСИ. Совместно с Никитой работаем над проектом по применению нейронных сетей и машинного обучения для моделирования различных этапов маршрута проектирования микроэлектроники. Помимо этого, наш центр вовлечен в различные задачи, связанные с искусственными нейросетями, мозгом.

Вадим Игоревич Лазарев:
Здравствуйте, коллеги. Меня зовут Вадим Игоревич Лазарев. Я являюсь ассистентом кафедры Вычислительных машин, систем и сетей МЭИ. На данный момент занимаюсь разработкой автоматизированной системы для настройки СВЧ устройств при помощи ИИ и машинного обучения.

Эдуард Артурович Челышев:
Здравствуйте. Меня зовут Эдуард Артурович Челышев. Я являюсь ассистентом кафедры Вычислительных машин, систем и сетей МЭИ. Меня интересуют исследования в области обработки медицинских изображений и не только при помощи ИИ. Также интересно обсудить вопросы, касающиеся обучения студентов искусственному интеллекту.

Сергей Викторович Вишняков:

Добрый день. Меня зовут Сергей Викторович Вишняков. Я являюсь директором института Информационных и вычислительных технологий МЭИ. Основной интерес – подготовка кадров с точки зрения использования и разработки ИИ.

Евгений Александрович Волошин:
Добрый день, коллеги. Меня зовут Евгений Волошин. Я являюсь представителем кафедры Релейной защиты и автоматизации энергосистем МЭИ. Основное направление работы – прикладное применение ИИ для задач проектирования, создания алгоритмов и интеллектуальных систем управления, в том числе распределенных.
Владимир Владимирович Чистяков:
Добрый день, коллеги. Меня зовут Владимир Владимирович Чистяков. Я являюсь ведущим инженером кафедры Радиотехнических систем МЭИ, разработчик ПЛИС. Также участвую в проекте по созданию ассистента разработчика ПЛИС.

Павел Юрьевич Анучин:
Добрый день, коллеги. Меня зовут Павел Юрьевич Анучин, я являюсь ассистентом кафедры Радиотехнических систем. В настоящее время занимаемся разработкой ассистентов для автоматизации и построения БИМ-моделей, ведем работу, которая позволит ускорить этот процесс.

Олег Вадимович Глухов:
Спасибо, коллеги. Сегодня планируется три содержательных доклада по тематике Проектирования микросхем. Важно, в докладе хотелось бы получить ответы на следующие вопросы: «Как появилась проблема?», «Как решалась проблема?» и «Что получилось, а что нет?». Передаю слово Никите Артемовичу Гревцеву.

Константин Александрович Петров:
Могу ли я рассказать вводный доклад перед тем, как передать слово Никите Артемовичу?

Олег Вадимович Глухов:
Вероятно, это будет полезно.

Константин Александрович Петров:
Откройте, пожалуйста, презентацию Манеркина – 4 слайд. Сейчас расскажу, о чем здесь идет речь. Есть такое понятие, как маршрут проектирования электронно-компонентной базы. В частности, для нас это проектирование сложных цифровых СБИС-микропроцессоров. На слайде представлен маршрут в нашем понимании, который используется в НИИСИ РАН. Правый столбец, который начинается с верификации на блочном уровне, далее идет создание отдельных сложно-функциональных блоков, как аналоговых, так и цифровых. Потом центральный столбец – это создание всей микросхемы, так называемого SoC, System-on-a-Chip, куда

интегрируются эти блоки со всех сторон. Левый столбец – это верификация и различные тестирования, которые существуют на всех уровнях производства, то есть мы всегда тестируем, всегда верифицируем все этапы, все представления как отдельных блоков, так и цельного микропроцессора. Если говорить про центральный столбец, то в какой-то момент, когда мы заканчиваем проектирование некой логической модели на языке описания аппаратуры микропроцессора, мы его отдаем на синтез непосредственно топологии кристалла, то есть мы должны из некого кода сделать в конечном счете рисунок, который отправится на фабрику. Фабрика по этому рисунку производит кристалл с заданными характеристиками, которые заранее в рамках системы автоматического проектирования знаем какие именно, то есть подбираем все тайминги, просадки напряжения питания, длительность разводки отдельных сигналов. На этом этапе огромное количество работы. И, соответственно, у нас в институте появилось понимание того, что на наиболее сложных этапах маршрута проектирования можно и нужно заменить человека на искусственный интеллект. Это в первую очередь, верификация как отдельных блоков левого столбца, так и верификация всего процессора. Верификация, в смысле тестирования, то есть поиск ошибок, потому что этот процесс вообще бесконечен, так как очень сложную микросхему невозможно протестировать до каждого бита, до каждого триггера. Поэтому нужно создавать такие тесты, чтобы ускорить процесс достижения некого максимума покрытия. И второй момент, в котором есть место для искусственного интеллекта и машинного обучения – физический синтез топологии этого кристалла. Там существует множество этапов, на многих из них можно также внедрить ИИ. Сегодня мы расскажем про самые сложные три места, которые у нас наиболее развиты. Передаю слово первому докладчику – Никите Артемовичу Гревцеву.

Никита Артемович Гревцев:
Еще раз здравствуйте, коллеги. Меня зовут Никита Артемович Гревцев. Тема сегодняшнего доклада – это машинное обучение в функциональной верификации микропроцессора на системном уровне. Рассмотрим вкратце, что такое верификация. Верификация в целом делится на два этапа – до и после выпуска кристалла, то есть пресиликон-верификация и постсиликон-верификация. И, в свою очередь, пресиликон-верификация делится по типам – на формальную и функциональную верификацию. В рамках формальной верификации с математической точки зрения доказывается, что какой-то блок корректен. В рамках функциональной верификации исследуется функционирование отдельного блока или процессора в сборе. Функциональная верификация делится на два типа. Это блочный уровень, в рамках которого входными воздействиями являются некие группы сигналов, выходными аналогично, то есть исследуется некий блок микропроцессора на системном уровне, в рамках которого исследуется функционирование микропроцессора в сборе, ядра микропроцессора или всего сока. Соответственно, выходными воздействиями будут являться инструкции ассемблерные, выходными воздействиями – трассы. В настоящий момент все больше крупных компаний, таких как Intel, ARM и исследовательских институтов, начинают применять различные виды машинного

обучения на всех этапах жизненного цикла разработки изделий и, собственно, на всех этапах верификации. Как на формальной верификации, так и на блочном уровне. Есть довольно много исследований по сокращению времени верификации или более точному направлению. Про это лучше расскажет следующий докладчик. На постсиликон стадии в основном работы сосредоточены на анализе очень больших объемов сырых данных, которые приходят с различных датчиков в микропроцессоре и так далее. Однако на стадии верификации на системном уровне наблюдается небольшой пробел, и большинство исследований сосредоточено на ранних стадиях верификации, например, на стадии верификации систем команд и так далее. Это представляет большой научный интерес, но с практической точки зрения это не так. Собственно, тема моего доклада – это применение машинного обучения на поздних стадиях верификации, когда необходимо найти уже самые сложные ошибки. И в рамках маршрута проектирования, соответственно, тема доклада – верификация на системном уровне. В целом, одной из основных задач при верификации на системном уровне является процесс направления тестового кода в сторону реальных задач пользователя. Так как самые ценные ошибки с точки зрения верификации это те, которые потребитель может встретить в своем продукте, то есть при запуске операционной системы. Собственно, процесс направления тестового кода в конечную область применения называется встречное тестирование. Давайте рассмотрим, как оно применяется в функциональной верификации на системном уровне. Для того, чтобы это достичь, различные блоки кода, а также некие архитектурные эвристики и метрики интегрируются в процесс верификации с помощью специализированных сценариев и макросов, которые подаются вместе с шаблоном на генератор тестов. Тест запускается одновременно на симуляторе RTL-модели и на эталонном эмуляторе. Соответственно, потом путем сравнения логов можно детектировать ошибку с точностью до одной инструкции. При этом в данном маршруте есть две ключевые проблемы. Первая проблема связана со стохастической природой данных генераторов. Так как по одному шаблону генерируется бесконечное число тестов, мы не можем гарантировать, что все тесты будут одинаково корректны и одинаково сложны. И вторая проблема – это очень сильная зависимость качества проводимой верификации от квалификации инженера-верификатора. Обе эти проблемы решаются с помощью инструментов машинного обучения. Первая проблема решается внедрением предварительной оценки сгенерированных тестов в процесс верификации. Для этого по каждому из шаблонов верификации были созданы базы тестов, которые были запущены на симуляторе со сбором покрытия. На полученной базе был обучен XGBoost регрессор, который позволяет нам оценивать предполагаемое покрытие у генерируемого теста до запуска на симуляцию, то есть до самой дорогой с точки зрения машинных ресурсов стадии. Так как генерация теста занимает порядка одной секунды, а симуляция может занимать часы. Поэтому нам наиболее выгодно отсекать нерепрезентативные тесты на ранней стадии. Внедрение этого метода позволило сократить используемые машинные ресурсы на 80% за счет сокращения требуемого количества тестов для выхода на плату покрытия данного шаблона, то есть до того момента, когда запуск нового теста уже не имеет особого

смысла. В рамках решения второй проблемы необходимо автоматизировать процесс получения и применения тестового знания, то есть исключить инженера-верификатора из данной конструкции. Для этого было предложено создать инструмент для автоматического получения экстрактов приложения пользователя – коротких и емких псевдослучайных ассемблерных тестов, которые бы отражали поведение оригинальной задачи, в которой они были построены, имели равномерно высокое покрытие и выполнялись на несколько порядков быстрее. В этом помогает инструмент машинного обучения. На слайде представлен фреймворк, разработанный генератором. Работу генератора можно разделить принципиально на две стадии. Первая стадия, отмеченная зеленым, это профилирование приложения с его сокращением времени выполнения. И вторая стадия, отмеченная синим, это максимизация покрытия в рамках каждого теста. Рассмотрим первую стадию. В рамках этой стадии нам необходимо сократить время выполнения приложения. Для этого по каждому из исследуемых пользовательских приложений строится граф переходов с описанием всех циклов и вызовов программ. В такой ситуации вершинами графа будут являться группы инструкций, а ребрами соединяющие их команды переходов. Затем происходит пропорциональное сокращение вызова всех программ и итерации циклов для сокращения времени выполнения. Для того, чтобы воспроизвести поведение оригинальной задачи, необходимо воспроизвести не только ее структуру, но и наполнение с точки зрения инструкций. Каждый блок инструкции, вершина графа, описывается с помощью функциональных метрик, то есть описываются типы инструкции, которые составляют данный блок, и различные архитектурно-независимые метрики, такие как падание или промах кэш-памяти. Для получения обратной связи и обучения модели каждый из полученных тестов можно запустить на симуляторе RTL-модели со сбором покрытия. Переходим ко второму этапу максимизации покрытия. В рамках этого этапа у нас из клона приложения (теста), полученного на предыдущем этапе, создается экстракт приложения, то есть приложение с максимизированным покрытием. Для этого агент обучения начинает менять коэффициенты в функциональных метриках, полученные на предыдущем этапе, и оценивать, насколько эти изменения были полезны, запуская тест на симуляторе Cadence. Получается, что в рамках этой задачи мы строим отображение функционального покрытия на покрытие RTL-модели. Задача агента обучения – это максимизация данной функции. Итоговым экстрактом будет являться состояние, в котором агент обучения больше не может повысить покрытие данного теста. Работа была проведена и проверена на пакете тестов производительности SPEC2000. Видно, что последняя колонка – экстракт приложения пользователя не потеряли в покрытии, но при этом удалось добиться сокращения времени выполнения приложений в среднем на два порядка. На этом всё в плане успешных применений. Однако, вы просили что-нибудь еще рассказать интересного про менее успешный опыт применения. Для решения последней задачи по автоматизации процесса получения тестового знания и генерации тестов, похожих на оригинальное приложение, изначально планировалось применять языковые модели для генерации тестов, так как оказалось, что это очень похожая задача. Если языковые модели учатся на

известных текстах и после этого могут создавать такие же тексты, внешне для наблюдателя схожие, то для порождения тестов это выглядит тоже весьма перспективным методом. Однако на данную задачу было потрачено довольно много времени без каких-то результатов. И у нас, видимо, не хватило либо квалификации, либо усидчивости, чтобы довести ее до конца. И в итоге мы свернули в сторону DQN и других способов. Но, что интересно, буквально через пару лет после того, как мы бросили данную задачу, мы находим статью о том, что компания Google сделала абсолютно точно такое же исследование, и в итоге добилась результата и сделали генератор, который работает как языковая модель. Получает на вход трассы приложений и на выходе создает ассемблерные тесты. На этом все. Спасибо большое. Я с радостью отвечу на вопросы.

Константин Александрович Петров:
У меня такой вопрос. Есть некая условно-автоматизированная система, которая берет задачи пользователя и превращает их в тесты для микропроцессора. То есть есть пользователь, который под этот микропроцессор написал какие-то программы, про эту архитектуру. А что будет, когда архитектура процессора изменится, когда изменятся эти программы? То есть есть некий автоматизированный вход, куда ты просто складываешь исходники пользовательского ПО и из них генерируешь автоматом вот эти тесты?

Никита Артемович Гревцев:
Как правило, мы сосредоточивались на известных задачах пользователя. Пакет тестов производительности SPEC2000 – это сборник типовых задач конечного пользователя. Однако, данная система позволяет работать и с другим ПО конечного пользователя. Соответственно, у нас есть потребитель, мы знаем его типаж, его задачи, которые он будет запускать.

Константин Александрович Петров:
То есть исходники его?

Никита Артемович Гревцев:
Нам даже не нужны исходники, мы используем именно трассы выполнения приложения. То есть мы можем не получать от конечного потребителя исходные коды приложений.

Константин Александрович Петров:
А брать ассемблеры, которые после компиляторов?

Никита Артемович Гревцев:
Брать бинарный файл, просто запускать его на своих симуляторах без получения возможно секретных или ограниченного доступа исходников.

Константин Александрович Петров:
Насколько я понимаю, там, где граф программы?

Никита Артемович Гревцев:
Граф потока управления, он строится по трассе приложения, по запуску.

Константин Александрович Петров:
Это ассемблерная инструкция, я правильно понимаю?

Никита Артемович Гревцев:
Это граф управления программой, то есть это визуализация трасс приложение, то есть а каждая команда перехода это вызов программы, итерация цикла и так далее каждый блок это некая группа инструкций

Константин Александрович Петров:
Хорошо. Тогда дальше вопрос: SPEC2000 – это 25 лет назад, а сейчас, может быть, он уже не актуален?

Никита Артемович Гревцев:
Сейчас периодически используется SPEC2006, но тем не менее SPEC2000 все еще остается идеальной солянкой с самыми репрезентативными задачами.

Константин Александрович Петров:
Но, видимо, какими-то несложными и универсальными максимально.

Никита Артемович Гревцев:
Максимально универсальными задачами, но которые все еще применяются. То есть, как оказалось бы, какой-нибудь Fortran, про который все уже забыли и никто не пишет, но на самом деле есть километры кодов, которые до сих пор используются.

Константин Александрович Петров:
Intel разве используют SPEC2000?

Никита Артемович Гревцев:
Intel, как правило, используют на SPEC2006. Это используется для того, чтобы можно было сравнить, у кого больше производительность, а у кого меньше. А для нас же интересно это не с точки зрения производительности, а с точки зрения типовой задачи пользователя и поиска ошибок.

Константин Александрович Петров:
Хорошо, тогда следующий вопрос. Каковы границы применимости разработанного решения, например программного обеспечения? Допустим, придут представители МЦСТ и скажут: «Можно ли адаптировать это под нас?». Что следует учитывать при

изменении архитектуры или, скажем, если вместо Cadence будет использоваться другой инструмент для оценки покрытия и аналогичных задач?

Никита Артемович Гревцев:
САПРы взаимозаменяемы, то есть можно заменить САПР от Cadence на какой-нибудь от open-source, но при этом система сильно заточена именно под ту архитектуру, на которой она была разработана, то есть под NIPS-подобную. При этом есть архитектурно независимые вещи, такие как процент попадания в кэш-память, вытеснение с кэш-памяти и так далее. Если взять другой процесс, тоже на архитектуре NIPS, то задачу можно переадаптировать. Для коллег из МЦСТ, я боюсь, это будет неприменимо.

Константин Александрович Петров:
Хорошо. Я к чему задаю эти вопросы по поводу границы применимости, по поводу актуальности теста. Было сказано, что решается задача того, что у верификатора, то есть человека, который пишет эти тесты и запускает их для модели процессора, имеет квалификацию, чтобы исключить критерии его квалификации, мы создаем автоматизированную систему. Квалификация человека повышается в процессе работы. Мы исключаем его работу. У него не будет расти квалификация. Придет студент, ему не надо будет учиться встречному тестированию и так далее.

Никита Артемович Гревцев:
Вот здесь интересный момент, особенно в контексте названия данного семинара «Педагогика ИИ». Потому что получилось, что в рамках выполнения этой работы мы поняли, что не нужно учить человека, нужно учить машину. И, собственно, здесь есть некий подводный камень с этим.

Олег Вадимович Глухов:
Тут ответ на ваш вопрос, скорее, философский. Можно фактически просто сузить число этих специалистов, которые будут уметь это делать, а все остальные будут пользоваться этим инструментом для решения задач.

Константин Александрович Петров:
Чем меньше людей работает над какой-то проблемой, тем меньше шансов, что там появится действительно умный человек. То есть это просто затормозит развитие, возможно, этой области.

Олег Вадимович Глухов:
А это нужно, если, допустим, эта задача уже будет автоматизирована решаться?

Никита Артемович Гревцев:
На самом деле абсолютно везде сейчас такое происходит. Google, Microsoft сейчас говорят, что они заменили помощников программирования. Порядка 30% кода сейчас

создается с помощью автоматизированных систем. Как правило, по последним опросам, ИИ-помощники используются как junior разработчики. При этом раньше эти задачи поручались junior разработчикам, и они на этих задачах становились middle разработчиками. Как это будет происходить теперь, не особо понятно. То есть это общая проблема, она связана не только с верификацией и так далее.

Константин Александрович Петров:
То есть решение этой проблемы пока что отсутствует, несмотря на попытки осознать полученные компетенции за последние полтора месяца.
Никита Артемович Гревцев:
Боюсь, что я именно в педагогике чуть менее силен, но меня интересует именно аспект применения и обучения аспирантов уже и так далее.

Олег Вадимович Глухов:
Встречный вопрос: В какой педагогике Вы менее сильны?

Никита Артемович Гревцев:
Как учить студентов, аспирантов в контексте постоянного применения и развития методов искусственного интеллекта. Когда студенты, аспиранты вместо того, чтобы чему-то обучаться, могут принести задачу, решенную искусственным интеллектом. Это же тоже замена процесса обучения на процесс выдачи правильного ответа. Это все общая, по-моему, проблема.

Олег Вадимович Глухов:
Понятно. Но тут как бы все комплексно. То есть на нашем семинаре мы в том числе решаем эту проблему. Это фактически тоже как задача: «Как научить ассистент», чтобы он правильно выдавал информацию. Поэтому тут все связано. Это надо разбирать. Коллеги, у кого вопросы есть?

Иван Михайлович Косарев:
Никита, вот эти вот методы машинного обучения применимы ли к фазе тестирования, к стресс-тестированию? Нужны ли какие-то квалификационные задачи для самих тестов, в первую очередь?

Никита Артемович Гревцев:
Фазинг-тестирование мы пока не исследовали. Это тоже отдельное направление. Я видел довольно много новых работ по этому тестированию и применению различных способов. Однако, для генерации стресс-тестов это максимально подходящий алгоритм, так как мы можем вместо нацеливания тестов на какое-то конечное приложение начать их нацеливать на максимальное покрытие или как-то задать эту функцию стресс-тестирования в виде целевой метрики. То есть вместо того, чтобы нацеливать на конечное положение, нацеливать на определенные условия. Для

стресс-тестирования это применимо, для фазинг-тестирования, к сожалению, не могу сказать.

Иван Михайлович Косарев:
Вы использовали квалификационные задачи для базовых тестов?

Никита Артемович Гревцев:
В данной работе нет.

Евгений Александрович Волошин:
Не совсем моя область знаний, поэтому попытаюсь сформулировать так, как смогу. Исходя из того, что представлено, в частности, на данной таблице, ключевой метрикой является степень покрытия кода теста. Ситуации бывают разные. То, что в процессе теста выполнение прошло через какой-то участок, говорит лишь о том, что он хотя бы раз был запущен. Из опыта нашей работы, мы более крупными блоками оперируем при создании алгоритма параллельной защиты. Запустить тест, в котором бы через все блоки прошло выполнение, еще не говорит о том, что это сработает во всех тех схемно-режимных ситуациях, в которых защита должна срабатывать. Соответственно, когда происходит какое-то неправильное срабатывание защиты, то есть весь код выполнился, задача выдала результат, но он функционально неверный. Собственно, к чему я веду. Функциональные проверки, они же тоже должны как-то проверяться. И генерация вот этих вот тестов, чтобы они функционально проверяли достоверность полученных результатов, тоже нужно учитывать, на мой взгляд. И второй момент, когда происходит какое-то неправильное срабатывание, это заносится в базу неправильных срабатываний, под которые пишется уже отдельный тест, который для всех последующих версий алгоритмов проверяется? Вот есть ли возможность в данной системе наполнять базу, которая накопилась на последующих итерациях?

Никита Артемович Гревцев:
Насчет первого вопроса по поводу функционирования: создается не просто тест, который похож, у нас основная метрика – схожесть теста на оригинальное приложение и его усложнение с точки зрения тестового покрытия, при этом функционирование теста продолжает проверяться, так как запускается в той же концепции общего маршрута, то есть полученный тест запускается и на RTL-модели, и на эталонном эмуляторе. Поэтому в случае возникновения ошибки она будет детектирована с точностью до одной инструкции. По второму вопросу: да, мы сохраняем все ошибки в базу, помимо наличия регрессионной базы у нас есть база лучших ошибок года, все имеющиеся ошибки используются потом в качестве сценариев и макросов для различных генераторов тестов. Они анализируются и используются в рамках обычного маршрута. При этом в рамках маршрута с применением машинного обучения это пока не используется, то есть полученные ошибки уходят на

предыдущий этап, на этап еще без машинного обучения. Это тоже отдельное направление для исследований.

Сергей Иванович Аряшев:
Еще маленький вопрос про языковые модели. Мне кажется, за пять лет языковые модели очень сильно поменялись. И вот текущие модели, и помощники, и ассистенты для программирования показывают, что, мне кажется, сейчас вообще можно еще раз зайти в языковые модели.

Никита Артемович Гревцев:
Да, вполне можно сделать еще один подход. Стало гораздо больше материала для более легкого входа во весь этот процесс. Порой не нужно быть специалистом для того, чтобы начать получать первые простые результаты и в дальнейшем совершенствовать данную модель.

Сергей Иванович Аряшев:
Единственное, наверное, будет сложность с обучением. Под нашу задачу обученных моделей не будет, а их надо как-то переучивать.

Олег Вадимович Глухов:
Коллеги, я бы дополнил. Вполне возможно, что текущей мощности вполне хватит даже без какого-то дообучения. Там все равно какой-то универсальный текстовый формат. Вопрос там адаптеров. Опять в это упираемся. На первом семинаре про это говорили. Есть предложение перейти ко второму докладчику. Никита Артемович, спасибо большое.

Алексей Дмитриевич Манеркин:
Добрый день, уважаемые коллеги. Меня зовут Манеркин Алексей Дмитриевич. Тема моего доклада – машинное обучение функциональной верификации на блочном уровне. Первое, с чего хотелось бы начать, это суммаризировать то, как разрабатываются микропроцессоры. Маршрут проектирования – это последовательный процесс, в ходе которого на основе технического задания последовательно формируются модели различного уровня абстракции, начиная с описания микропроцессора на C, далее RTL-описание, в котором работа модели представляется как преобразование цифровых сигналов, считываемых из регистров и записываемых в них. После чего RTL-описание переводится на этапе логического синтеза в Netlist, то есть в совокупность стандартных заводских ячеек, который затем переводится в топологию, после чего топология отправляется на завод и изготавливается микросхема. Каждое преобразование должно быть логически верным, то есть модели должны быть логически эквивалентны. Проверкой логической правильности преобразования моделей занимается этап верификации.

Современный микропроцессор – это устройство, которое состоит из большого числа блоков различного назначения и сложности, таких как ядра, различные системы памяти, это регистровые файлы, кэш-памяти, АЗУ, ПЗУ, контроллеры памяти, системные шины, периферийные интерфейсы. Каждый из этих блоков можно тестировать как в составе всей системы, так и по отдельности. Последний метод верификации называется блочной верификацией. Блочная верификация требует меньших временных затрат, что позволяет внедрять тестовые окружения на блочном уровне в состав CI (Continuous Integration). А независимость тестового окружения на блочном уровне от окружающих блоков компонентов микропроцессора позволяет внедрять в состав сложных функциональных блоков при продаже вот эти самые тесты на блочном уровне и тестовое окружение. Каким образом вообще тестируется микропроцессор на блочном уровне, точнее отдельные сложные функциональные блоки? У нас есть, допустим, RTL-описание блока, которое подключается к тестовому окружению, пишется набор тестов. В ходе работы тестов, тестовое окружение генерирует транзакции, которые подаются на тестируемый блок, работа которого затем симулируется в специальной программе. Возникает вопрос, а вообще до какой степени нам нужно верифицировать, потому что верификация может идти бесконечно. Для того, чтобы понимать, на каком уровне мы сейчас находимся, существует количественная метрика, которая называется тестовое покрытие. Тестовое покрытие отражает полноту тестирования, и ее значение равняется количеству проверенных элементов к общему числу элементов, элементами могут быть либо структурные единицы кода, так называемое кодовое покрытие. Это, собственно, как строки кода, так и состояние конечных автоматов, выражение, переключение сигналов и так далее, либо же функциональные требования. Такое покрытие называется функциональным. В связи с усложнением блоков процесс верификации становится все более длительный, требуется все большее число транзакций для того, чтобы достичь требуемого уровня покрытия. При этом большая часть транзакций является нерепрезентативными, не вносит значительного вклада в повышение уровня покрытия и в принципе может быть отброшена. Соответственно, возникает проблема. А как это сделать? Одним из вариантов решения этой проблемы является ускорение достижения максимально возможного уровня покрытия при помощи методов машинного обучения. На данном слайде представлено тестовое окружение с ускоренным достижением покрытия. Тестовое окружение работает в несколько этапов. Сначала генерируется обучающая тестовая выборка, на которой обучается модель, подстраиваются ее гиперпараметры, после чего в тело случайного генератора на уже обученную модель подается набор параметров транзакции, модель оценивает ее полезность для повышения уровня покрытия, и если набор параметров транзакции оказывается малополезным, то модель его отбрасывает. Оптимизированный набор параметров транзакции подается на UVM-окружение, и UVM-окружение уже преобразует это в транзакции и отправляет на симуляцию на тестируемое устройство. То есть самый вообще длительный этап в тестировании – это именно работа симулятора. Чем дольше требуется симуляция, тем дольше идет тестирование. На данном слайде представлены результаты работы тестового

окружения для различных моделей. Модели были разделены на два класса. Это одиночные модели и гибридные модели. Как мы видим, подавляющее большинство моделей смогло достичь значительного выигрыша по числу транзакций для выхода на плато по кодовому покрытию, в особенности гибридные модели. Но в процессе исследований возникли проблемы. Первое – это необходимость дообучения или повторного обучения модели при внесении изменений в блок. То есть у нас постепенно вносятся какие-то правки функционирования в RTL-описании того или иного блока, и спустя какое-то время модель может оказаться нерелевантной для повышения уровня покрытий. Тогда требуется ее повторно обучить или же дообучить.

Олег Вадимович Глухов:
Можно сразу встречный вопрос. Как часто это может происходить?
Алексей Дмитриевич Манеркин:
Это хороший вопрос. Это вообще такая общая проблема во всех статьях. Хочу отметить немножко заранее, что даже в самом худшем случае, если каждый раз потребуется обучать модель, все равно мы будем иметь выигрыш относительно просто псевдослучайной генерации.

Олег Вадимович Глухов:
Ну это как? Раз в день, раз в час, раз в секунду?

Алексей Дмитриевич Манеркин:
Раз в правку. Это самый худший случай.

Олег Вадимович Глухов:
Ну это вот какой интервал времени?

Константин Александрович Петров:
Разработчик поправил блок, допустим, он там неделю сидел и нашел у себя какую-то ошибку. После этого он добавляет блок снова в базу, и там начинается над ним работа по тестированию, чтобы потом найти следующую ошибку.

Алексей Дмитриевич Манеркин:
Так вот, это первая проблема. Вторая – это необходимость настройки тестового окружения при масштабировании на переносе на другие блоки. Это тоже требует определенных затрат времени инженера-верификатора. Обобщенный результат приведен в данной таблице, продемонстрирован выигрыш по сравнению с псевдослучайной генерацией без учета затрат на обучение и с учетом затрат на обучение. Вот как раз-таки правый столбец, это вот тот самый наихудший случай, когда при каждой правке требуется переобучать модель. Все равно мы имеем значительный выигрыш, может не такой большой, как без учета затрат на обучение, но все-таки он делает выгодным использование применения методов машинного

обучения. В результате мы имеем значительное сокращение времени на процесс верификации блоков, подобное тестовое окружение может быть внедрено в текущие тестовые обвязки для различных сложных функциональных блоков, когда требуется провести какие-нибудь быстрые тестирования. Следующий вопрос, а куда идти дальше? Было выделено несколько возможных направлений. Это, прежде всего, использование рекуррентных нейронных сетей, которые архитектурно подходят под задачу обработки временных рядов, включая LSTM, обучение с подкреплением, которое показало свою эффективность, особенно при использовании ИИ-агентов, допустим, для той же генерации кода и ускорения тестирования.

Олег Вадимович Глухов:
Это можно сравнить с той задачей, про которую Никита рассказывал? То есть, фактически, будет примерно такая же функциональная схема и тестовое окружение, на которое нужно воздействовать, если использовать RL.

Алексей Дмитриевич Манеркин:
В чем-то они похоже, но у Никиты тесты более программно-подобные. У меня больше наборы параметров транзакций по интерфейсу.

Гревцев Никита Артемович:
В моем случае было необходимо сделать тесты, похожие на оригинальные задачи, а здесь основными задачами являются сокращение времени выполнения за счет уменьшения количества транзакций и достижение тех же результатов, что и при псевдослучайной генерации, только за гораздо меньшее время. А также поиск узких ситуаций, которые очень сложно найти с помощью псевдослучайных.

Олег Вадимович Глухов:
Просто оптимизационная функция, которую нужно максимизировать или минимизировать, она другая. Если рассматривать саму концепцию, то, по сути, такая же функциональная схема будет с точки зрения применения RL.

Константин Александрович Петров:
Если у Вас функция – это похожесть на пользовательские задачи, то у Алексея функция – покрытие. Это не просто изменение одного параметра в методике, а одинаковые параметры, к которым вы стремитесь. Разве это не так?

Гревцев Никита Артемович:
Не всегда так. Я так понимаю, что в данной работе RL будет применяться не только для сокращения времени, но еще и для поиска узких сценариев. То есть среди всего множества точек покрытия, как правило, одна-две точки, которые недостижимы или почти недостижимы в рамках случайной генерации. Можно миллионы тестов прогнать и на них не наткнуться.

Алексей Дмитриевич Манеркин:
Текущие исследования больше концентрируются на ускорении тестирования. Еще очень интересной задачей является именно повышение уровня покрытия, потому что в этих последних процентах тоже могут крыться какие-то потенциальные проблемы, ошибки. Как раз для этого можно использовать RL. Никитина задача, она именно больше про получение эквивалента, а здесь оптимизация и поиск каких-то возможных сценариев для повышения покрытия.

Олег Вадимович Глухов:
Хорошо. С точки зрения функциональной схемы, какое противоречие? Ну, по сути же, одна и та же парадигма. То есть нужно создать некого агента, который работает в среде, со среды принимает состояние и как-то воздействует на эту среду. Вводим еще некую функцию, которую надо минимизировать или максимизировать.

Гревцев Никита Артемович:
В этом плане вы правы.

Олег Вадимович Глухов:
Хорошо. Вопрос про рекуррентные нейронные сети. Как предлагается это использовать? Для чего нам нужно знать временную память?

Алексей Дмитриевич Манеркин:
Так как у нас, по сути, последовательность транзакций является своего рода временным рядом, в ходе выполнения она находится на тестированном блоке последовательно, поэтому архитектурно такой класс сетей подходит под решение задачи.

Гревцев Никита Артемович:
Не все тестовые ситуации можно достичь единичной транзакцией. Для некоторых, особенно сложных тестовых ситуаций, необходимо взведение системы в некое новое состояние и действие из нее. То есть необходима последовательность транзакций за ограниченное время.

Алексей Дмитриевич Манеркин:
Самый простой пример – это перейти в какое-то состояние конечного автомата, которое находится где-то глубоко в этом графе. Туда невозможно попасть чисто за одну транзакцию, нужна какая-то последовательность.

Константин Александрович Петров:
А сейчас только одна?

Алексей Дмитриевич Манеркин:
Да, сейчас пока только одна.

Олег Вадимович Глухов:
А вот вопрос, когда вы делаете сейчас по одной транзакции, ее буквально как одну временную отметку берете? Или, допустим, вы за счет просто самой архитектуры вводите входные данные сразу несколько временных тактов? Вы так делаете?

Алексей Дмитриевич Манеркин:
На модель отправляется именно одна транзакция.
Константин Александрович Петров:
Одна транзакция – это последовательность десятков или тысяч тактов, правильно? Это не просто одно состояние поставили, дали клок и блок работает, правильно?

Алексей Дмитриевич Манеркин:
Нет, это именно пакет данных.

Константин Александрович Петров:
Пакет данных по шине AXI, допустим, процессорной шины?

Алексей Дмитриевич Манеркин:
Да, которая может быть растянута на десятки тактов.

Константин Александрович Петров:
Но при этом это одна транзакция?

Алексей Дмитриевич Манеркин:
Да, одна транзакция.

Константин Александрович Петров:
Получается, туда нельзя довольно сложные вещи добавить, для которых нужны рекуррентные сети.

Олег Вадимович Глухов:
Да, которые более длительные последовательности могут принимать. Понятно. Спасибо.

Константин Александрович Петров:
Это конец доклада?

Алексей Дмитриевич Манеркин:
Нет. Второй этап – это исследование применения машинного обучения на различных этапах тестирования RTL-моделей. То есть не только оптимизация достижения

покрытия, но и повышение уровня покрытия. Можно проводить анализ лог-файлов, генерацию тестовых сценариев. И третий этап – это исследование применения больших языковых моделей для автоматической генерации утверждений, так называемые assertions. Это тоже важный этап верификации.

Константин Александрович Петров:
Можно пример assertions?

Алексей Дмитриевич Манеркин:
Допустим, у нас есть сигнал А. Если он переводится в единичное состояние по какому-то клоку, и спустя три такта, у нас появляется сигнал В, который перейдет в состояние ноль.

Константин Александрович Петров:
Вследствие взведения сигнала А? И assertion в данном случае – это набор фраз о том, что это надо проверить?

Алексей Дмитриевич Манеркин:
Набор формальных утверждений. По сути, спецификацию мы берем и формализуем ее как раз в такие емкие assertions. Концептуально большие языковые модели могли бы подойти для этого.

Константин Александрович Петров:
То есть, когда есть некое описание блока текста, ТЗ на него, или документация, которая говорит, что эти сигналы работают определенным образом, и нужно перевести это на формальный язык описанных требований. С русского на другой, например.

Алексей Дмитриевич Манеркин:
Да. Спасибо за внимание.

Иван Михайлович Косарев:
Вопрос следующий. Насколько сложно работать с языковыми моделями? Допустим, это усиленная генерация промтов, заданий и проверки их функционирования. Это же достаточно трудоёмкая, кропотливая вещь. Много на это времени уходит?

Алексей Дмитриевич Манеркин:
На это может уйти много времени, действительно, если недостаточно грамотно составить промт и проследить за моделью. Скажем так, это не панацея. Языковая модель, допустим, при генерации какой-нибудь программы, может тоже делать

значительные ошибки, не учитывать многие моменты. То есть очень важным аспектом является именно формализация запроса на естественном языке.

Сергей Викторович Вишняков:
Можно уточнить? В этом докладе, предыдущем и, я так полагаю, в последующем мы говорим о решении обратной задачи, то есть знаем, какой должен быть ответ и хотим подобрать набор входных воздействий, которые дадут заданный спектр выходных. В этом же смысл тестирования? Мы сейчас обсуждаем варианты углубления и применения все более и более мощных методов, с точки зрения аппаратных затрат, программных затрат. А классические методы не дадут ли более простой и детерминированный вариант? Еще раз подчеркну, мы применяем очень ресурсозатратные методы, потому что под капотом находятся те же большие языковые модели, хотя задача, мне кажется, достаточно хорошо формализована. Вы знаете ответ и структуру блока. Именно классическое решение обратной задачи не будет ли лучше применить здесь? Или слишком сложно?

Алексей Дмитриевич Манеркин:
Классическое решение, во-первых, требует поиска этого решения. Во-вторых, в чем особенность тестового окружения со встроенной моделью на основе методов машинного обучения – такое тестовое окружение будет универсальным для различных блоков на том же или ином интерфейсе. Допустим, взять тот же интерфейс AXI – можно для блока А обучить модель, которая будет ускорять верификацию для него, и также перенести на блок В и обучить ее уже для блока В. В случае с классическим методом – это не очевидно.

Олег Вадимович Глухов:
Мне кажется, Сергей Викторович имел в виду первичные методы машинного обучения, которые мы имели до больших языковых моделей. Я правильно понимаю? Или буквально классические методы?

Сергей Викторович Вишняков:
Методы-то могут быть, конечно, совершенно разные. Я имею в виду именно вопрос исследования и решения обратной задачи. Это же действительно обратный синтез?

Гость 1:
На самом деле, если бы они знали точное решение задачи, то и проблем бы не было. Программист делает блок, и он уверен, что он будет работать во всех случаях. На самом деле это не так. Он проверяет на тех случаях, которые он знает, но есть масса случаев, которые он просто не может предусмотреть. Поэтому они начинают прогонять тестовую программу. Где-нибудь вылезает ошибка. Это мы знаем по себе, казалось бы, делаешь программу и в зависимости от того, как ты переставляешь блоки, вдруг она начинает вызывать совершенно другую программу. Логически все было правильно, это так называемая проверка на дурака.

Сергей Викторович Вишняков:
Выходы-то известны, Вы же знаете, что это ошибка.

Гость 1:
Выходы известны только в известных случаях. Вот они как раз ловят ошибку, что вдруг выход не тот, который хотели.

Сергей Викторович Вишняков:
То есть они знают, какую хотелось? Я это и говорю – эталонный выход есть.

Гость 1:
Выход есть только в эталонных задачах. Они на этих эталонных задачах, перебирая все возможные конфигурации, и вдруг совершенно не то, что мы думали. Потому что ошибка в программе.

Алексей Дмитриевич Манеркин:
В данном исследовании выходные какие-то моменты более-менее известны. Но, по сути, на практике это становится известно только постфактум.

Сергей Викторович Вишняков:
Но вы же сравниваете, у вас во всех алгоритмах сравнение выхода одной модели и эталонной модели, то есть эталонная есть.

Алексей Дмитриевич Манеркин:
Эталонная модель – псевдослучайная генерация.

Иван Михайлович Косарев:
Коллеги, извините, можно для саморазвития вопрос? Вот понятие Golden Chip Module, это что такое? Кто-нибудь сталкивался?

Алексей Дмитриевич Манеркин:
Это и есть как раз-таки эталонная модель, которая сравнивается с тестируемым блоком, если у нас возникает несовпадение выхода этого тестируемого блока и Golden модели, то это становится предметом анализа.

Константин Александрович Петров:
Вот здесь на втором слайде как раз Golden Chip Module – это описание микропроцессора на С языке, а RTL-описание – это намного более подробное описание аппаратуры. Именно сравнение RTL-описания и эталонной модели на С – процесс верификации на данном этапе, по сути, тестирование. Мы считаем, что С работает правильно, там ошибок нет. Когда по этому описанию на С пишется углубленная модель, где каждый регистр прописан, и каждая машина стоит подробно, вот тогда-то оно почему-то не работает.

Александр Павлович Еремеев:
Коллеги, развивая мысль Сергея Викторовича, хочу сфокусироваться на вопросе надежности. Есть золотое правило системного анализа: чем сложнее модель, тем она менее надежна. Поэтому я призываю к взвешенному подходу. Если задачу можно решить с помощью предсказуемых формальных моделей вроде вероятностных автоматов — нужно использовать их. Если мы сразу переходим к глубокому обучению, мы жертвуем надежностью ради гибкости, и это нужно осознавать.
То же касается и логического аппарата. Простые исчисления понятны и разрешимы. А логика первого порядка, необходимая для сложных задач, уже несет в себе проблему неразрешимости — система может уйти в бесконечный вывод. Это снова заставляет нас усложнять систему, а значит, снижать ее надежность. Мое предложение — двигаться по принципу «step by step». Начинать с самых простых работающих моделей. Усложнять их следует только тогда, когда это действительно необходимо, и на каждом шаге задавать себе вопрос о надежности. И здесь мы подходим к большим языковым моделям. Их внедрение сопряжено с серьезными рисками:
- Качество данных: они требуют идеальных выборок для обучения и тестов. Как мы решаем эту проблему?
- Галлюцинации: всем известна эта проблема. Сама OpenAI признает, что ошибки в ответах могут составлять 40-60%.
- Неустойчивость: это фундаментальная проблема нейросетей. Система может работать идеально 99% времени, а потом внезапно выдать критическую ошибку.
Поэтому я хочу задать прямой вопрос: прежде чем подключать такой мощный, но нестабильный инструмент, как LLM, мы должны быть уверены, что без него не обойтись. Нужно четкое обоснование, почему нам не подходят более простые и надежные методы. Зачем именно в этой задаче нужна большая языковая модель? Нет ли здесь избыточности, которая только снизит надежность конечного решения?

Алексей Дмитриевич Манеркин:
Хотел бы сказать, что аналитическое решение для постоянно изменемого блока, наверное, все-таки будет проблематично найти. Во-вторых, в данной задаче используется не большая языковая модель, а метод машинного обучения, а именно конкретно градиентный бустинг. То есть это ансамблевая модель, состоящая из деревьев решений и гибридной модели на основе градиентного бустинга

Гревцев Никита Артемович:
Я думаю, вопрос именно про дальнейшее исследование, что ты упоминал про ЛСТМ.

Александр Павлович Еремеев:
На одном из слайдов вы упоминали планы по использованию большой языковой модели. Однако контекст и необходимость ее использования были не до конца понятны. Есть ли у вас ресурсы, которые обеспечат её качественную работу?

Алексей Дмитриевич Манеркин:
В нашем случае задача большой языковой модели сводится к следующему: она должна проанализировать текстовое описание и на его основе сгенерировать формальный код на языке System Verilog.

Александр Павлович Еремеев:
В таком случае, возникает закономерный вопрос: почему не рассматривается классическая продукционная модель? Этот подход также позволяет обрабатывать описания на естественном языке, преобразуя их в структурированную информацию, такую как семантические триплеты. Ключевое преимущество продукционной модели заключается в построении логического дерева вывода. Этот механизм обеспечивает два фундаментальных свойства: полноту и непротиворечивость рассуждений, чего большие языковые модели гарантировать не могут. Более того, древовидная структура вывода обеспечивает полную интерпретируемость решения. Мы можем проследить всю цепочку логических шагов, которые привели к результату. Большие языковые модели, напротив, функционируют по принципу «черного ящика». Они выдают конечный ответ, но не предоставляют объяснения, как именно он был получен, что лишает нас возможности верифицировать его логическую корректность.

Олег Вадимович Глухов:
Да, Александр Павлович, я тут дополню вас. Мы просто на предыдущем семинаре как раз-таки разбирали кейсы с применением экспертных систем для решения каких-то промышленных задач. В экспертных системах используется дерево решений, в которое специалисты закладывают правила для конкретных сценариев. И как раз-таки, наверное, видится, что перспективным это комбинация сложных моделей по типу LLM и экспертных систем, в том числе, потому что там объяснимый компонент присутствует. Рассматривали ли вы возможность применения экспертных систем в этом направлении?

Алексей Дмитриевич Манеркин:
Да, в этом определенно есть потенциал. Идея объединить экспертные знания и LLM — это, пожалуй, хорошее направление. Смысл ведь не в том, чтобы полностью отдать задачу на откуп модели, а в том, чтобы процесс оставался под контролем специалиста. И да, для нашей задачи это выглядит как правильный путь. Ведь кто-то все равно должен проверить, что сгенерированные утверждения полны и корректны. Без эксперта здесь не обойтись.

Олег Вадимович Глухов:
Там еще важно, можно привязать конкретное правило к конкретной фамилии, чтобы знать, с кого спросить.

Гревцев Никита Артемович:

Если можно, я дополню на самом деле. Довольно важный момент нужно прояснить. Во всех процессорах есть ошибки. Во всех выпущенных процессорах. То есть это даже не какая-то наша проблема. В процессорах Intel есть ошибки, в телефонах есть ошибки. На самом деле, процесс верификации заканчивается не по достижении определенного результата, когда мы нашли все ошибки в процессоре или достигли 100% покрытия. Верификация заканчивается, по истечении установленного срока (дедлайна). Поэтому порой нам необходимо получить максимально возможные результаты за минимальное время. Строго говоря, наиболее надежным методом является формальная верификация. В результате у нас получается строгое математическое доказательство, что блок работает полностью корректно, не делает ничего лишнего, то, что в него не внесено, и функционирует в строгом соответствии со своим техническим заданием. Казалось бы, все идеально, но этим методом не пользуется почти никто. Просто потому, что ресурсы, которые необходимо потратить на верификацию небольшого даже блока кода, будут на порядке больше, чем ресурсы на создание этого блока кода. Поэтому порой необходимы модели, которые дадут быстрые ответы, быстро повысят покрытие до какого-то минимального порога, чтобы повысить общее покрытие до отсечки.

Олег Вадимович Глухов:
Следует ли из этого, что на данном этапе верификации приоритетом является время, а не абсолютная надежность?

Гревцев Никита Артемович:
Нет, на самом деле надежность важна. Надежность важна, во-первых, в области применения микропроцессора. То есть типовая задача пользователя – это обязанность работать корректно. То есть в этом плане сильнее уже применяется системный уровень. При этом ошибка может сохраняться в редких гипотетических ситуациях.

Олег Вадимович Глухов:
Тогда такой вопрос. Может быть, как раз опасения коллег избыточны в этом плане? Вопрос такой. Сами по себе программные средства проверки как таковые, они могут проанализировать некоторые галлюцинации, условно, если мы LLM включим? То есть, если откровенный бред LLM будет выдавать у вас, есть какой-то барьер, который не даст этому пройти?

Гревцев Никита Артемович:
Но на самом деле, даже если мы этот бред запустим на симуляцию, то самое страшное, что произойдет, мы потратим машинный ресурс. То есть мы просимулировали некую галлюцинацию, она не дала нам никакого прироста производительности, мы потратили машину и ресурсы, потратили немножко времени до дедлайна. При этом мы не внесли какую-то ошибку в проект и так далее.

Олег Вадимович Глухов:

А кто оценку производит? Человек?

Гревцев Никита Артемович:
Покрытия? А покрытие оценивается системой. То есть это независимо от человека.

Олег Вадимович Глухов:
Ну хорошо, а может быть такое, что вы покрыли, но это ошибка другого рода.

Гревцев Никита Артемович:
Здесь галлюцинации представляют серьезную проблему. Дело в том, что проверочные утверждения (assertions) для системы пишет либо специалист, либо LLM. Если модель при их создании начнет "фантазировать", мы получим некорректные сценарии для проверки. Это значит, что система будет тратить ресурсы на анализ ситуаций, которые либо не имеют отношения к делу, либо просто бессмысленны.

Гость 1:
Позвольте я уточню, правильно ли я уловил мысль. Если наша цель — это конкретный, измеримый результат, например, покрыть 100% тестовых сценариев, то возникает вопрос об эффективности. Мы уже слышали, что самые большие трудности возникают с последними процентами покрытия — до них нужно добраться через очень хитрые комбинации. И вот какое решение мне видится более оптимальным. Вместо того чтобы постоянно гонять тяжелую LLM, можно поступить иначе:
1. Взять нашу историю тестов за прошлые годы — все лучшие результаты, все найденные ошибки — и собрать из этого качественную «экспертную» базу.
2. На этой базе обучить более простые и быстрые алгоритмы, заточенные на поиск аномалий.
3. Именно их и направить на самую сложную работу — выискивать те самые граничные случаи, чтобы добить покрытие до 100%.
4. А все, что они найдут, добавлять обратно в нашу экспертную базу, делая ее еще лучше.
На мой взгляд, это гораздо более экономный по ресурсам и времени подход, чем использовать LLM для всей задачи целиком.

Гревцев Никита Артемович:
На самом деле LLM может в том числе дополнить общие проценты, то есть создать некий новый assertion, то есть утверждение, новый тестовый сценарий, которого еще не было. И, соответственно, после этого процесса все имеющиеся у нас тесты будут открывать уже не 100%, а 99%.

Олег Вадимович Глухов:
Ну понятно, то есть вы изначально посчитали, что там 10 тестов должно быть, в итоге реально для того, чтобы устройство работало, должно быть 1000, допустим.

Гревцев Никита Артемович:
Да, это как раз то, о чем вы говорили в самом начале. Полное покрытие кода может создавать иллюзию безопасности.
Мы можем довольно быстро достичь 100% покрытия, но это не гарантирует, что в программе нет багов. Проблема в том, что тест может просто «пройти» по строке с ошибкой. Счетчик покрытия поставит галочку — «строка задействована». Но если при этом никто не проверил, правильный ли результат выдала эта строка, то ошибка так и останется в коде. Мы формально выполнили требование, но суть проблемы не решили.

Алексей Дмитриевич Манеркин:
Речь может быть о какой-то уникальной последовательности, переключении сигналов, которые эту ошибку как раз таки выводят в свет.

Олег Вадимович Глухов:
Сергей Викторович, ваш вопрос был по поводу использования аналитических методов или все-таки машинного обучения низкоуровневого?

Сергей Викторович Вишняков:
Сейчас как раз все очень логично звучит и про градиентный бустинг, и про разумные модели, которые на своих деревьях решения. Вызывает определенные опасения стремление использовать LLM для решения подобных задач в будущем.

Алексей Дмитриевич Манеркин:
Просто речь идет об обработке естественного языка, поэтому оптимальным решением является использование больших языковых моделей, способных обрабатывать спецификации.

Олег Вадимович Глухов:
Вероятно, ключевым фактором является экономия ресурсов и времени, что и определяет стремление к использованию этих технологий.

Алексей Дмитриевич Манеркин:
Речь идет не о полной автоматизации, а именно об оптимизации решения рутинных задач, более простого решения рутинных задач. И сокращение времени, чтобы меньше терроризировал затрат вычислительных ресурсов, если взять на ту же формальную верификацию, с сильнейшим ростом сложности.

Олег Вадимович Глухов:
Хорошо, Константин, а у вас будет вопрос касательно развития, касательно того, что будет, если, соответственно, мы всех заменим?

Константин Александрович Петров:
Алексей, хочу задать вопрос о будущем наших инженеров, немного конкретизировав то, о чем говорил Геннадий Никитин. Как, по-вашему, ваша разработка повлияет на новичков — выпускников и джуниоров, которые к нам приходят? Давайте посмотрим на две группы: на RTL-дизайнеров, которые пишут код, и на верификаторов, которые его проверяют. Что изменится для них?
Во-первых, как мы будем их учить здесь, у нас? На чем они будут набивать руку, если часть рутины уйдет?
Во-вторых, чему их должны учить в университетах, чтобы они приходили к нам готовыми? На что делать упор в образовании?
И, наконец, самый главный вопрос: что станет с этими профессиями? Они просто изменятся или могут со временем стать менее востребованными?

Алексей Дмитриевич Манеркин:
Верификаторы в обозримой перспективе никуда деться не должны. Наработки, прежде всего, упрощают этап верификации, они не заменяют верификатора.

Константин Александрович Петров:
Почему? Есть набор тестов. Раньше был псевдослучайный выбор, сейчас выбор с использованием некой модели бустинга. То есть мы опять заменили квалифицированного верификатора, который выбирал вместо псевдослучайного, наверное, либо просто псевдослучайный, заменили на машинное обучение и все.

Алексей Дмитриевич Манеркин:
Верификатор, прежде всего, пишет эти тесты. Эти тесты запускают в симуляторе, а в симуляторе уже есть псевдослучайная генерация. Очень многое упирается в то, что большая часть тест-кейсов не очень релевантна. Использование модели снижает затраты времени на отсеивание релевантной транзакции.

Константин Александрович Петров:
То есть ранее отбор тестов был полностью псевдослучайным, без ручной оценки их релевантности верификатором?

Алексей Дмитриевич Манеркин:
Верификатор просто задает какие-то ограничения, область ограничений.

Константин Александрович Петров:
Хочу провести параллель. На системном уровне мы уже проходили этот путь: от «слепого» случайного поиска к осмысленному, который направлял квалифицированный инженер. Теперь мы эту квалификацию пытаемся «зашить» в модели.
Но у вас, на уровне отдельного блока, такой проблемы, кажется, нет. Правильно я понимаю, что вам не нужно выбирать лучшие тесты из тех, что сгенерированы или

написаны вручную? Раньше на этом уровне просто запускали все подряд, и у вас, похоже, та же ситуация — нет сложной задачи выбора. Я прав?

Алексей Дмитриевич Манеркин:
Раньше просто создавались ограничения. То есть диапазон ограничений, в рамках которого у нас генерируется параметр транзакции, так называемая ограниченная случайная генерация.

Константин Александрович Петров:
Раньше ограничения для тестовых сценариев задавал человек, он их продумывал. Сейчас, когда для генерации тестов используются методы машинного обучения, эти ограничения как-то учитываются?

Алексей Дмитриевич Манеркин:
Да, на данный момент эти ограничения существуют и задаются вручную. Однако в теории их также можно было бы настраивать с помощью машинного обучения — по этой теме уже есть научные работы.

Константин Александрович Петров:
Но если и этот процесс будет автоматизирован, какие задачи останутся для начинающего специалиста? Снизится ли потребность в джуниорах-верификаторах? Вероятно, их роль сведется к формализации этих ограничений?

Алексей Дмитриевич Манеркин:
Не совсем. Его задачи сместятся на более высокий уровень: он будет не только формализовать требования, но и продумывать нетривиальные сценарии — так называемые граничные случаи (corner cases) и другие сложные ситуации.

Олег Вадимович Глухов:
Получается аналогия с известной утопией: когда базовые потребности закрыты, человек начинает думать о высоком.

Алексей Дмитриевич Манеркин:
Именно. Освободившись от рутины, специалист сможет сфокусироваться на поиске сложных кейсов, а также на разработке архитектуры тестовых окружений и наборов тестов.

Константин Александрович Петров:
То есть на данном этапе мы не ставим задачу полностью заменить человека искусственным интеллектом в этой творческой части?

Алексей Дмитриевич Манеркин:
На данном этапе — не ставим.

Константин Александрович Петров:
Спасибо, это полностью отвечает на мой вопрос.

Алексей Дмитриевич Манеркин:
Единственная область, где в теории возможна замена человека, — это генерация проверочных утверждений (assertions) с помощью LLM. Но даже в этом случае за специалистом остается ключевая экспертная функция: он обязан проверить и верифицировать то, что сгенерировала модель.

Сергей Иванович Аряшев:
Коллеги, я хочу обратить ваше внимание на серьезную кадровую проблему. Порог входа в RTL-проектирование и верификацию уже сейчас является критически высоким, что приводит к низкому уровню удержания молодых специалистов в профессии. Интеграция ИИ, как мне кажется, еще больше усложняет и расширяет этот порог входа.
Мы рискуем прийти к ситуации, когда на предприятии будет невозможно растить кадры. Если раньше мы могли давать студентам и начинающим сотрудникам простые задачи для постепенного погружения, то с новыми, усложненными инструментами это станет невозможным. В результате мы можем зайти в кадровый тупик, когда набор молодых специалистов станет нерешаемой задачей. Вся ответственность за их базовую подготовку фактически перекладывается на вузы, но смогут ли они справиться с этой задачей — большой вопрос.

Олег Вадимович Глухов:
Сергей Иванович, решением здесь является не отказ от технологий, а смена самой методики подготовки кадров. Текущая парадигма — линейная: мы предполагаем, что специалист должен постоянно накапливать все возрастающий объем знаний для входа в профессию. Этот путь ведет в тупик.
Более перспективным видится переход к специализации. Мы можем прийти к ситуации, когда для одной части инженеров ИИ-ассистенты станут стандартным рабочим инструментом для решения прикладных задач. Одновременно с этим будет формироваться другая группа специалистов, целенаправленно подготовленных для решения фундаментальных, глубоких задач верификации, требующих досконального понимания основ. Путь развития будет расходиться на несколько направлений. Невозможно и не нужно требовать, чтобы каждый специалист знал абсолютно все, так как общий объем знаний в отрасли будет только расти.

Гость 3:
Коллеги можно задать вопрос? Здесь было рассказано что автоматическое проведение доказательств функциональной корректности, что этого сейчас никто не делает. Но

вот я, на самом деле, по смарт контрактам изучал – там есть статьи очень, так сказать, цитируемые, известные статьи, где как раз таки используется автоматизация проведения доказательств функциональной корректности смарт-контрактов. Но я думаю, что и смарт-контракт – это программа, то есть и другой тип программ тоже, наверное. Почему вот вы считаете, что формальная верификация – это уже устаревшее и это не подходит?

Гревцев Никита Артемович:
Дело не в том, что формальная верификация устарела — она просто очень дорогая. У нас зачастую нет ресурсов проверить формально даже крошечную часть микропроцессора. Этот метод подходит для самых важных блоков, но применить его ко всему чипу, состоящему из сотен тысяч строк кода, невозможно.

Гость 2:
А разве ESPRAN не этим занимается?

Гревцев Никита Артемович:
ESPRAN занимается похожими вещами на системном уровне, но это подтверждает мою мысль о дороговизне. Был случай, когда их инженер почти год сидел рядом с нашим разработчиком, чтобы описать внутреннее устройство всего одного блока. А потом устройство блока менялось, и все начиналось заново. Это превращается в бесконечный и очень затратный процесс.

Алексей Дмитриевич Манеркин:
То есть, сложность блока растет линейно, а затраты на его формальную проверку — экспоненциально.

Олег Вадимович Глухов:
Алексей Дмитриевич, я думал, мы все еще про знания говорим. Это был ответ на предыдущий вопрос?

Алексей Дмитриевич Манеркин:
Да, именно так.

Олег Вадимович Глухов:
Отлично. Тогда предлагаю переходить к третьему докладу. Если останется время, сможем продолжить дискуссию позже. Никита Владимирович, вам слово.

Никита Владимирович Желудков:
Добрый вечер, коллеги. Сегодня я хотел бы рассказать о прикладной задаче, которой недавно занимался: об использовании машинного обучения для оценки длины цепей

в чипе еще до этапа их физической трассировки. Ключевая проблема заключается в том, что реальную длину всех соединений мы узнаем только после завершения этапа трассировки — очень длительного процесса, который для больших проектов может занимать несколько дней. Однако эта информация необходима нам гораздо раньше, еще на этапе размещения ячеек, поскольку от длины цепей напрямую зависят скорость работы чипа, его энергопотребление и качество будущей разводки. Соответственно, чем точнее мы сможем оценить длину на этапе размещения, тем меньше итераций проектирования нам потребуется, что позволит сэкономить дни, а порой и недели работы.

Конечно, эта задача не нова, и для ее решения существуют стандартные подходы. Есть классический метод HPWL (Half-Perimeter Wire Length), который очень быстр, но, к сожалению, неточен — он просто вычисляет полупериметр прямоугольника, охватывающего ячейки. Существуют и другие академические методы, например FLUTE, но в промышленных САПР они не прижились из-за ряда проблем с точностью. Последние годы наиболее перспективным направлением стало применение машинного обучения, как в работах Net2 или RouteNet.

Мы также пошли по этому пути, но применили особый подход к представлению данных. Мы представляем схему в виде графа, но делаем это, можно сказать, «наоборот»: узлами в нашем графе выступают сами цепи, а ребрами — стандартные ячейки, которые их соединяют. Такой подход позволяет свести задачу к регрессии на узлах графа, где для каждой цепи-узла мы предсказываем ее будущую длину. Для этого мы извлекаем для каждой цепи девятимерный вектор признаков, включающий логическую информацию, такую как количество входов-выходов и площадь ячеек, а также топологическую, в том числе и значение базовой метрики HPWL.

Для проверки нашего подхода мы собрали датасет из 32 open-source проектов, которые были пропущены через стандартный маршрут проектирования на техпроцессе 28 нанометров с использованием САПР Genus и Innovus. В итоге мы получили данные по более чем 700 тысячам цепей с их реальными, измеренными после трассировки длинами. Далее мы сравнили пять моделей машинного обучения с классическим HPWL: две простые модели, такие как MLP и XGBoost, и три графовые нейронные сети — GCN, GAT и GraphSAGE.

Результаты оказались весьма показательны. Если базовый метод HPWL ошибается в среднем на 72%, а простые ML-модели снижают ошибку до 23-29%, то наша лучшая графовая модель на основе GraphSAGE достигла ошибки всего в 16%. Да, ее работа занимает немного больше времени — около 10 секунд на тестовом блоке против двух у HPWL, — но такой значительный прирост точности, на наш взгляд, является оправданным компромиссом. Интересной особенностью стало то, что на длинных цепях модель работает даже точнее, чем на коротких, что, вероятно, связано с их большей корреляцией с глобальными параметрами, которые мы учитываем.

Таким образом, разработанная модель является не просто теоретическим исследованием, а практическим инструментом. Ее можно интегрировать в САПР для более точной и ранней оценки частоты и энергопотребления чипа, для предсказания проблемных зон при разводке, а также для улучшения самих алгоритмов размещения

стандартных ячеек и макроблоков, предоставив им более точную целевую функцию для оптимизации. На этом мой доклад закончен. Спасибо за внимание.

Олег Вадимович Глухов:
Да, Никита, спасибо. Коллеги, вопросы?

Гревцев Никита Артемович:
Можно простой вопрос? Я немного в замешательстве. Вы сказали, что HPWL используется в Innovus, но при этом у него ошибка 72%. Как такое возможно? Это же значит, что метод почти бесполезен.

Никита Владимирович Желудков:
Отличный вопрос, сейчас поясню. Когда я говорю, что он «используется», я имею в виду его роль в процессе оптимизации. Мы видим, что в логах Innovus на каждом шаге размещения значение wirelength уменьшается. Это говорит о том, что САПР использует эту метрику, чтобы понимать, стало ли размещение лучше или хуже. То есть ее задача — не предсказать точное финальное значение, а показать направление для улучшения.
Если говорить об open-source инструменте OpenROAD, то там HPWL используется именно так, в открытую. Конечно, в Innovus могут быть свои «секретные» доработки, в которые мы заглянуть не можем, но основная логика, скорее всего, та же: HPWL служит компасом для алгоритма, а не точным измерительным прибором.

Константин Александрович Петров:
Тем не менее, несмотря на то что это значение последовательно уменьшается, сама оценка остается крайне неточной. Я правильно понимаю, что средняя погрешность составляет 72%?

Никита Владимирович Желудков:
Да, по результатам моих расчетов, средняя ошибка составляет именно 72%.

Олег Вадимович Глухов:
Здесь важен не только процент ошибки, но и коэффициент детерминации R^2, который для HPWL составляет всего 0.72. Это говорит о том, что модель обладает очень низкой предсказательной силой, и ее результаты ненадежны.

Константин Александрович Петров:
Это вполне ожидаемо для столь простого математического алгоритма, который не способен отразить реальную сложность процесса.

Никита Владимирович Желудков:
Да, он не учитывает множество факторов. Справедливости ради отмечу, что в данной работе не проводилось сравнение с самыми современными state-of-the-art алгоритмами из публикаций 2022 года, так как их реализация требовала значительных временных ресурсов. Безусловно, существуют и более точные подходы.

Константин Александрович Петров:
Возможно, дело в характере ошибки? Является ли она системной — например, HPWL всегда дает оценку сверху, — или же она случайна?
Никита Владимирович Желудков:
К сожалению, я не могу дать однозначный ответ на этот вопрос. Мой анализ ограничивался расчетом метрик ошибки, а не исследованием ее природы.

Олег Вадимович Глухов:
Низкий коэффициент детерминации (0.72) как раз и указывает на высокую долю случайной, непредсказуемой компоненты в ошибке.

Константин Александрович Петров:
Понятно. Хотя, как мы знаем, случайные на первый взгляд ошибки могут иметь системные причины.

Гость 1:
Каков механизм, гарантирующий монотонное улучшение на каждой итерации оптимизации? Существует ли вероятность регрессии к предыдущим или даже худшим состояниям?

Никита Владимирович Желудков:
Этот вопрос относится непосредственно к алгоритмам размещения, которые не являются предметом данного доклада. Однако, по своей сути, это стандартная задача оптимизации. Несмотря на возможные локальные флуктуации, целевая функция алгоритма САПР направлена на поиск глобального минимума.

Константин Александрович Петров:
Именно так. Процессы синтеза и размещения в САПР являются итеративными, где на каждом шаге происходит улучшение схемы на основе предыдущего результата. Таким образом, негативные факторы, такие как длина цепей и энергопотребление, должны последовательно оптимизироваться.

Гревцев Никита Артемович:
Каков порядок числа итераций в данном процессе?

Никита Владимирович Желудков:

Например, Innovus сообщает о 20 крупных итерациях, в то время как open-source инструмент OpenROAD выполняет около 400 более мелких итераций.

Гревцев Никита Артемович:
Следовательно, можно предположить, что неточность базовой метрики (HPWL) нивелируется большим количеством итераций.

Никита Владимирович Желудков:
Да, это логичное предположение. Однако необходимо помнить, что мы делаем выводы на основе лог-файлов. Внутренние механизмы коммерческого САПР Innovus остаются закрытыми, в отличие от OpenROAD, где процесс полностью прозрачен.

Олег Вадимович Глухов:
Возвращаясь к слайду 10, к графику зависимости предсказанной длины от реальной. Вы упомянули, что для больших длин ошибка минимальна. Не связано ли это с малым количеством таких примеров в выборке? На графике их представлено очень мало.

Никита Владимирович Желудков:
На графике представлена лишь репрезентативная выборка из 200 тысяч точек из общего датасета в 700 тысяч. Это сделано для сохранения читаемости визуализации. В действительности, для длинных цепей также имеется достаточное количество примеров.

Олег Вадимович Глухов:
Понятно, спасибо. Коллеги, есть ли еще вопросы?

Иван Михайлович Косарев:
У меня вопрос. В работе использовался стандартный маршрут проектирования без дополнительных инструментов?

Никита Владимирович Желудков:
Да, все операции выполнялись стандартными командами САПР и скриптами на Tcl.

Иван Михайлович Косарев:
В таком случае, позволяет ли ваша модель оценивать такие параметры, как площадь и, как следствие, стоимость кристалла, на более ранних этапах — до разработки RTL-модели, например, на основе TLM-описания или формальных спецификаций?

Никита Владимирович Желудков:

Мне известны работы по оценке параметров на основе готового RTL-кода, но не на более ранних стадиях. Точность прогноза напрямую зависит от уровня детализации входных данных. Можно построить иерархию моделей: наименее точная будет работать с RTL, более точная — с нетлистом после синтеза, и самая точная — с данными после размещения, как в нашем случае.

Иван Михайлович Косарев:
То есть для какой-либо оценки требуется хотя бы формальная спецификация?

Никита Владимирович Желудков:
Теоретически возможно. В качестве примера можно привести САПР Cadence Joules, который оценивает мощность на основе RTL, для чего выполняет быстрый внутренний синтез. Но даже ему требуется на входе как минимум RTL-модель, прошедшая синтаксический анализ.

Иван Михайлович Косарев:
То есть для работы модели необходима структурная, а не поведенческая информация.

Никита Владимирович Желудков:
Да, точность напрямую зависит от глубины прохождения по маршруту проектирования и наличия конкретных структурных данных.

Олег Вадимович Глухов:
Спасибо. Есть ли еще вопросы?

Гость 2:
Хотел бы предложить возможное направление для дальнейшего развития. Проделана отличная работа, и было бы интересно применить вашу модель для оценки congestion при решении задачи партиционирования крупных проектов. Можно было бы анализировать различные варианты разбиения проекта на блоки и с помощью вашей модели предсказывать congestion на межблочных соединениях, выбирая таким образом оптимальную конфигурацию.

Никита Владимирович Желудков:
Да, такой подход возможен. Это отчасти пересекается с работами, которые мы ведем. Однако в качестве следующего шага я планирую использовать данную модель как компонент в более сложной системе для автоматической расстановки макроблоков. Идея в том, чтобы на каждой итерации размещения макроблоков оценивать не только длину соединений, но и комплексный набор метрик, включая временные характеристики, мощность, congestion и доступность выводов (pin accessibility), чтобы выбирать наиболее удачное решение.

Олег Вадимович Глухов:
Уточните, пожалуйста, что представляет собой параметр congestion?

Никита Владимирович Желудков:
Congestion — это метрика, которая позволяет еще до этапа детальной трассировки оценить потенциальные проблемы с разводимостью. Алгоритм разбивает кристалл на условную сетку и для каждой ячейки этой сетки сравнивает два параметра: количество доступных трасс и количество цепей, которые необходимо через них провести. Если спрос на трассы превышает их доступное количество, возникает congestion — «узкое место», где с высокой вероятностью появятся нарушения проектных норм (DRC) или трассировка окажется невозможной. Наличие таких зон напрямую влияет на итоговое время трассировки и качество результата.

Олег Вадимович Глухов:
Понятно. У меня еще пара технических вопросов. Вы упомянули библиотеку PyTorch Geometric. В чем ее специфика?

Никита Владимирович Желудков:
Ее ключевая особенность — наличие готовых, оптимизированных реализаций слоев для графовых нейронных сетей. Три из пяти моих моделей использовали слои GCN, Graph Attention и GraphSAGE, которые доступны в этой библиотеке «из коробки».

Олег Вадимович Глухов:
То есть это принципиально отличается от архитектуры многослойного перцептрона?

Никита Владимирович Желудков:
Да. Графовые слои используются для получения высокоразмерного векторного представления, или эмбеддинга, для каждого узла. Этот эмбеддинг затем подается на вход стандартной полносвязной сети для решения конечной задачи регрессии. Концептуально это схоже с архитектурой Transformer и другими современными моделями, где сначала извлекается богатое признаковое описание объекта, а затем оно используется для классификации или регрессии.

Олег Вадимович Глухов:
И результаты подтверждают, что такой подход эффективнее.

Никита Владимирович Желудков:
Совершенно верно. Результаты показали, что модели, учитывающие только признаки отдельного узла, менее точны. Графовые сети, напротив, выполняют свертку, агрегируя информацию от соседних узлов. Таким образом, эмбеддинг каждого узла начинает содержать в себе информацию о его топологическом контексте в графе.

Наше предположение о том, что эта дополнительная информация о логических связях повысит точность, полностью подтвердилось.

Евгений Александрович Волошин:
Скажите, а планируете ли вы решать «прямую задачу» — то есть не оценивать метрики, а непосредственно выполнять расстановку или трассировку?

Никита Владимирович Желудков:
Что касается расстановки, то, как я уже упоминал, мы ведем работы по автоматизации расстановки макроблоков с использованием обучения с подкреплением. Эта задача сейчас часто решается вручную и сильно зависит от квалификации инженера. Задача же полной детальной трассировки является на порядок более сложной, и на данный момент мы ее не рассматриваем.

Евгений Александрович Волошин:
Понятно, спасибо. Эта задача сейчас является головной болью для многих.

Олег Вадимович Глухов:
Коллеги, если вопросов больше нет, позвольте мне подвести итоги по основной теме нашей сегодняшней повестки. Сегодняшние доклады отчетливо продемонстрировали, что необходимость в дальнейшей автоматизации процессов проектирования микросхем, в частности в области верификации и точного расчета длин цепей, не вызывает сомнений. Однако эта тенденция ставит перед нами два фундаментальных вопроса, на которые пока нет однозначных ответов.
Первый вопрос — кадровый и образовательный. Если мы автоматизируем все больше рутинных задач, то чему мы будем учить студентов и начинающих специалистов, приходящих на предприятие? Станут ли они просто операторами сложных инструментов, или их роль трансформируется?
Второй вопрос касается надежности. Стремясь к максимальной автоматизации, мы вынуждены применять все более сложные инструменты, включая нейронные сети, у которых снижается уровень интерпретируемости и, как следствие, падает надежность получаемых результатов. Это те ключевые вызовы, которые, я полагаю, нам предстоит обсуждать в рамках будущих семинаров.
Теперь несколько организационных моментов. Во-первых, поскольку круг участников наших семинаров расширяется, есть предложение создать общий чат в Телеграме для оперативного обмена информацией и идеями. Если возражений нет, мы это сделаем.
Во-вторых, следующий семинар запланирован на 6 августа. Коллеги представят большой и содержательный доклад о внедрении ИИ-ассистентов для создания гибких образовательных программ на примере пилотного проекта с техникумами Москвы. Этот кейс напрямую связан с нашими интересами в области педагогики ИИ, поскольку он затрагивает вопрос, как создать инструмент, который эффективно решает образовательные задачи. Возможно, это обсуждение поможет нам найти подходы к

решению проблемы, как готовить специалистов к работе в условиях тотальной автоматизации, и в целом поднимет вопросы методики обучения.
Возможно, у вас уже есть идеи, какие темы вы хотели бы обсудить на следующих семинарах или кого из экспертов стоило бы пригласить?

Гость 2:
Позвольте небольшое замечание по поводу сложности систем. Несколько лет назад NVIDIA публиковала статьи об использовании искусственного интеллекта для обучения молодых сотрудников особенностям и тонкостям их сложного маршрута проектирования. ИИ в данном случае выступал в роли интеллектуальной базы знаний, которая помогала находить ответы на специфические вопросы новичков.

Олег Вадимович Глухов:
Это очень точное замечание. Фактически, некоторые организации уже работают в этом направлении, используя тестирование для выявления скрытых предрасположенностей и особенностей мышления студентов. Например, кто-то более усидчив и требует одного подхода к подаче материала, а кто-то, наоборот, страдает дефицитом внимания и нуждается в другом. У каждого есть сильные стороны, которые могут быть применимы в разных задачах. Это особенно актуально для нового поколения, привыкшего к формату коротких видео и быстрой смене информации. Если наше поколение способно к длительной концентрации на кропотливых задачах, но менее гибко в переключении, то у молодежи обратная ситуация: они могут испытывать трудности с долгой, монотонной работой, но зато отлично справляются с многозадачностью. Это важный педагогический и методический вызов. Коллеги, возвращаясь к теме, есть ли у вас пожелания по будущим семинарам?

Сергей Иванович Аряшев:
У меня есть два пожелания.

Константин Александрович Петров:
Позвольте я добавлю. Сегодня мы показали вам те работы в области ИИ, которые ведутся применительно к маршруту проектирования. Но, помимо этого, у нас в НИИСИ есть Центр оптико-нейронных технологий, представителями которого являются Борис Владимирович и Яков Михайлович. Думаю, их участие также было бы интересно. Яков Михайлович мог бы рассказать, для каких прикладных задач в области математики и физического моделирования они применяют ИИ. Этот центр объединяет специалистов, которые непосредственно проектируют нейронные сети, а не только используют их для решения прикладных задач.

Олег Вадимович Глухов:
Уточню, они занимаются аппаратной реализацией нейронных сетей?

Гость 2:

Нет, мы занимаемся математическим моделированием. Однако я хотел бы уточнить: тематика нашего семинара сфокусирована исключительно на педагогике?

Константин Александрович Петров:
Нет, тематика шире. Я предлагаю обсудить это в рабочем порядке, но в качестве официального предложения хотел бы это зафиксировать.

Олег Вадимович Глухов:
Позвольте мне пояснить. Да, основная тема — педагогика. Но чтобы понять, чему и как обучать, мы должны изучать лучшие практики применения этих технологий для решения конкретных отраслевых задач. Готовые решения, такие как ChatGPT, DeepSeek и другие, не работают «из коробки». Их нужно адаптировать под наши нужды: возможно, дообучить, настроить коэффициенты или даже разработать специальные «адаптеры» для преобразования данных. Изучение таких успешных кейсов — это и есть неотъемлемая часть формирования образовательной методики.

Константин Александрович Петров:
Второе предложение касается разработанной в нашем институте образовательной платформы «Мирера». Она используется в нескольких вузах, включая МГУ, и сейчас внедряется в Курчатовском институте. В платформу интегрированы элементы ИИ для автоматической оценки выполнения заданий и выявления отстающих студентов. Этим направлением у нас занимается доктор педагогических наук Александр Георгиевич Леонов. Я бы хотел попросить Олега Вадимовича направить ему материалы следующего семинара и пригласить выступить либо самого Александра Георгиевича, либо его коллег, непосредственно работающих над платформой.
И третье направление. Если мы хотим глубже изучить применение ИИ в проектировании микроэлектроники, стоит обратить внимание на работы, которые ведутся под эгидой Фонда перспективных исследований (ФПИ). ФПИ курирует проекты по внедрению ИИ на всех этапах маршрута проектирования на отечественных фабриках. В России в этой области уже накоплен значительный опыт. Было бы полезно пригласить на один из семинаров соответствующего куратора из ФПИ для обмена информацией и обсуждения возможных точек соприкосновения.

Олег Вадимович Глухов:
Это отличная идея. Коллеги, есть ли еще предложения? Возможно, у наших участников, подключившихся онлайн?

Александр Павлович Еремеев:
Позвольте добавить. Когда мы говорим об автоматизации, речь не идет о полной замене специалиста. Напротив, методы искусственного интеллекта должны стать инструментом, который помогает человеку отсеивать второстепенную информацию и концентрироваться на ключевых аспектах задачи, тем самым повышая его эффективность.

Олег Вадимович Глухов:
Александр Павлович, большое спасибо за это содержательное замечание. Коллеги, у меня последний организационный вопрос: могу ли я опубликовать ваши презентации на портале нашего семинара?

Иван Михайлович Косарев:
Да, конечно. Это публичная работа.

Олег Вадимович Глухов:
Отлично. В таком случае предлагаю на этом завершать. Всем спасибо за участие.

Иван Михайлович Косарев:
Спасибо. До свидания.