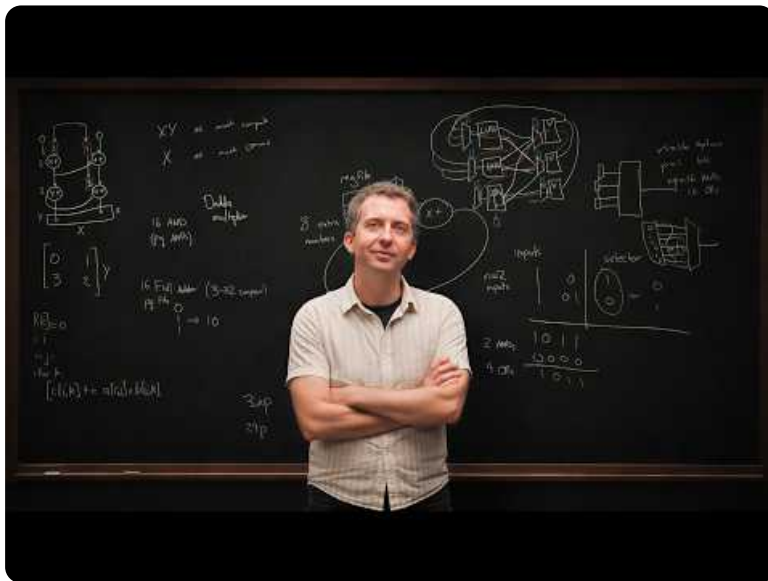


Chip design from the bottom up - Reiner Pope

Dwarkesh Patel · 2026-05-22 · <https://www.youtube.com/watch?v=olk3R-sMX5o>



Выпуск подкаста Dwarkesh Podcast, в котором ведущий Дваркеш Пател беседует с Райнером Поупом, CEO стартапа MatX (разрабатывающего ИИ-чипы). Разговор полностью посвящен низкоуровневой архитектуре процессоров: от базовых логических вентилей до макроархитектуры GPU и TPU.

Главная мысль выпуска: архитектура ИИ-чипов на всех уровнях — от выбора разрядности чисел до компоновки систолических массивов — продиктована необходимостью максимизировать объем полезных вычислений по отношению к колоссальным затратам площади кристалла и энергии на перемещение данных.

Базовые примитивы и математика на уровне вентилей

Умножение с накоплением (MAC) — фундаментальный примитив ИИ-чипов 00:00: Базовая операция нейросетей — перемножение матриц, которое сводится к вложенным циклам операций умножения с накоплением ($\text{output}[i,k] += \text{input}[i,j] \times \text{input}[j,k]$). В ИИ-чипах разрядность аккумулятора всегда выше разрядности множителей (например, умножение двух 4-битных чисел и прибавление результата к 8-битному). Это необходимо, так как при суммировании длинных рядов чисел быстро накапливаются ошибки округления, в то время как в самом умножении участвуют только два числа, и ошибка не масштабируется.

Схемотехника умножения требует квадратичного числа вентилей 03:28: При умножении 4-битного числа на 4-битное формируется 16 частичных произведений. Каждое из них вычисляется с помощью логического вентиля AND

(результат равен 1, только если оба бита равны 1). В общем случае для умножения числа размерностью p бит на число размерностью q бит требуется $p \times q$ вентилях AND.

Сложение реализовано через полные сумматоры (компрессоры 3-в-2)

06:10: Самый крупный базовый вентиль — полный сумматор. Он принимает на вход три 1-битных числа (в одном столбце разрядов) и выдает 2-битный результат (сумму и перенос в следующий разряд). Для сложения всех частичных произведений применяется множитель Дадда (Dadda multiplier): полные сумматоры итеративно применяются к столбцам битов, каждый раз забирая три бита и возвращая два, пока не останется одно итоговое число. Для схемы умножения $p \times q$ требуется ровно $p \times q$ полных сумматоров.

Фактчек: Для умножителя $p \times q$ нужно ровно $p \times q$ полных сумматоров — $\triangle$: Порядок величины верный, слово «ровно» — нет: дерево Дадды для $n \times n$ требует $\sim n^2 - 4n + 3$ полных сумматоров плюс полусумматоры и финальный сумматор с переносом; GPT оценивает строже ($\square$) и добавляет, что полный сумматор — составная схема, а не «самый крупный базовый вентиль».

Площадь кристалла масштабируется квадратично от разрядности чисел

12:31: Схемотехническая площадь умножителя растет пропорционально квадрату длины битового представления. Именно поэтому снижение точности (FP8, FP4) дает колоссальный прирост производительности нейросетей. Исторически при переходе от FP8 к FP4 Nvidia удваивала количество флопсов, просто уравнивая площадь на кристалле, но из-за квадратичного масштабирования реальный прирост должен быть 4х. В архитектуре B300 Nvidia уже заявляет о трехкратном ускорении FP4 по сравнению с FP8.

Фактчек: B300: FP4 в 3х быстрее FP8, а по квадратичному закону «должно быть» 4х — $\triangle$: Расхождение рецензентов по цифре: Gemini считает официальным приростом 2х (но это спецификации B200), Claude сверил именно B300/Blackwell Ultra — ~ 15 PF dense FP4 против ~ 5 PF FP8, т.е. 3х сходится. Все трое согласны, что «идеальные 4х» — ошибка рассуждения: квадратично сжимается только умножитель мантисс (4 бита $\rightarrow$ 2 бита), а аккумулятор FP32/FP16, экспонентная логика, нормализация и шины данных не уменьшаются.

Фактчек: 7/8 площади ядра уходит на перемещение данных; оптимум — 10% на маршрутизацию, 90% на массив — $\square$: Это вентильный подсчёт мультимплексора для игрушечного 8-регистрового файла, а не измерения реальных CPU/GPU, и не универсальный оптимум проектирования. Сам тезис независимо подтверждён (Horowitz, ISSCC 2014): 32-битное сложение $\sim 0,1$ пДж, доступ в регистровый файл ~ 1 пДж, SRAM — единицы пДж, DRAM ~ 640 пДж — движение данных дороже арифметики на 1-3 порядка.

Проблема перемещения данных и систолические массивы

Перемещение данных обходится в разы дороже самих вычислений 16:20:

В классическом ядре CPU или CUDA-ядре данные перемещаются из регистрового файла в АЛУ и обратно. Выбор нужного регистра осуществляется мультиплексором (mux). Для выбора одного из n регистров шириной p бит требуется $n \times p$ вентилях AND и $(n-1) \times p$ вентилях OR. В итоге схема перемещения данных (мультиплексоры на входе и выходе) занимает около $7/8$ площади кристалла, и лишь $1/8$ уходит на само вычислительное устройство. Для 8-строчного регистра перемещение стоит $24 \times p$ вентилях, а само умножение — всего $4 \times p$.

Фактчек: Снижение частоты не экономит энергию на операцию, поэтому «гони на максимум и уходи в idle» — $\triangle$: Расхождение: Gemini подтверждает (при $P \approx CV^2f$ и $t \approx 1/f$ энергия на операцию постоянна), Claude и GPT указывают, что это верно только при фиксированном напряжении. Реальные чипы с DVFS снижают и напряжение, энергия падает как V^2 — в датацентровых режимах это 1,5–3× экономии на операцию. Race-to-idle оправдан утечками и амортизацией фиксированных затрат, но не тем, что эффективность «почти не меняется».

[21:31-22:34] Рекламная вставка Crusoe Cloud.

Систолические массивы (Tensor Cores) решают проблему стоимости маршрутизации 25:37: Чтобы сместить баланс от перемещения данных к вычислениям, ИИ-чипы аппаратно реализуют внешний цикл перемножения матриц. В систолическом массиве матрица весов хранится локально прямо в логических вентилях (регистрах) самого массива и используется повторно для множества входящих векторов. Векторы проходят через массив пространственно, формируя скалярные произведения по столбцам.

Фактчек: Tensor Cores — это систолические массивы — $\triangle$: Weight-stationary систолический массив точно описывает MXU в TPU. Nvidia никогда не называла Tensor Cores систолическими; микробенчмарки указывают на небольшие блоки скалярных произведений (порядка $4 \times 4 \times 4$ / $8 \times 4 \times 16$) с иной организацией потоков. Как учебная модель годится, как факт об архитектуре Nvidia — нет.

Загрузка весов происходит медленно для экономии пропускной способности 32:23: Объем вычислений в систолическом массиве растет как x^2 , а объем ввода-вывода ограничен величиной x . Чтобы не тратить площадь кристалла на широкие шины данных от регистрового файла, матрица весов загружается в массив постепенно (trickle-feed). Данные подаются в верхний ряд и с каждым тактом сдвигаются на строку ниже (daisy chain). Это минимизирует количество проводов, пересекающих границу массива.

Проектирование чипа сводится к балансировке площади 36:12: Главные решения архитектора — это выбор размеров. Размер систолического массива и размер регистрового файла жестко связаны бюджетом площади кристалла.

Увеличение регистрового файла дает больше гибкости для софта, но отнимает транзисторы у вычислительного массива. Оптимальный баланс — выделять около 10% площади на перемещение данных и 90% на систолический массив.

Фактчек: Scratchpad в TPU гарантирует 100% детерминированность времени выполнения — $\triangle$: Расхождение: Gemini подтверждает (компилятор XLA явно планирует пересылки HBM $\leftrightarrow$ SRAM, промахов кэша нет), GPT возражает ($\square$): устраняется непредсказуемость кэша, но остаются задержки HBM, конкуренция за каналы и межчиповая связь. Итог: сильно предсказуемее CPU — да; абсолютная гарантия для всей программы — преувеличение.

[37:59-39:00] Рекламная вставка Cursor.

Тактовая частота, синхронизация и конвейеры

Тактовый сигнал синхронизирует миллиарды транзисторов 39:00: В отличие от софта, где потоки синхронизируются мьютексами, в чипах используется глобальный тактовый сигнал. Каждую наносекунду все схемы синхронизируются: значения на проводах фиксируются в регистрах. Это необходимо из-за производственных погрешностей (manufacturing variance) — разные логические пути выполняются с разной скоростью. Без синхронизации быстрый сигнал придет на следующий узел слишком рано и смешается с данными из предыдущего такта.

Вставка конвейерных регистров разменивает площадь на частоту 43:33: Задержка прохождения сигнала через облако логики должна укладываться в один такт. Если разбить длинную логическую цепочку пополам и вставить посередине регистр (pipeline register insertion), можно удвоить тактовую частоту чипа. Однако это требует дополнительных транзисторов для хранения промежуточных состояний.

Циклы обратной связи ограничивают максимальную тактовую частоту 45:53: Если вычисление замыкается само на себя (например, аккумулятор, прибавляющий новое число каждый такт), вставить конвейерный регистр внутрь цикла нельзя — это изменит саму математику операции (вместо общей суммы получится две независимые суммы четных и нечетных тактов). Именно такие циклы задают предел тактовой частоты чипа.

Слишком высокая частота снижает общую пропускную способность 48:10: Базовые примитивы от TSMC (вентили AND, сумматоры) срабатывают за ~ 10 пикосекунд. В один такт можно уместить 10-30 таких операций. Теоретически можно поднять частоту до 5-6 ГГц, ставя регистр после каждого вентиля. Но базовый вентиль занимает 1 единицу площади, а регистр — 8 единиц. При экстремальном разгоне почти вся площадь кристалла уйдет на регистры синхронизации, что катастрофически снизит количество полезной работы за такт (throughput).

[50:52-51:25] Рекламная вставка Jane Street.

FPGA против ASIC: архитектура и экономика

ASIC в 10 раз дешевле и энергоэффективнее, но FPGA не требует

тейпаута 51:25: Производство первого экземпляра ASIC стоит около 30 млн из-за необходимости создания фотомасок (tape-out). FPGA (ПЛИС) стоит ~10 000 за штуку, но позволяет менять аппаратную логику прямо в дата-центре. FPGA используется там, где нужна детерминированная задержка и высокая параллельность, но алгоритмы меняются слишком часто (например, в высокочастотном трейдинге), чтобы окупать выпуск ASIC.

Фактчек: Тейпаут ASIC ~30 млн, FPGA ~10 000, ASIC в 10х эффективнее — □:

Верно по порядку величины: набор масок на 5/3 нм — 15–30 млн (28 нм — 2–5 млн, 3 нм может превышать 100 млн), полный NRE с дизайном и верификацией — 100 млн+. По Kuon & Rose (2007) FPGA проигрывает ASIC ~35х по площади, ~14х по динамической мощности, 3–4х по частоте — 10х даже консервативно. GPT добавляет: разрыв зависит от задачи и использования встроенных DSP/SRAM в FPGA, а FPGA бывают и сильно дешевле \$10k.

Логика FPGA построена на таблицах истинности (LUT) и мультиплексорах

53:42: FPGA состоит из регистров, LUT (Lookup Tables) и огромного количества мультиплексоров. «Программирование» FPGA заключается в настройке битов управления мультиплексорами, которые определяют, как физические провода соединяют элементы друг с другом.

LUT реализует любой логический вентиль через память 56:43: Стандартный LUT имеет 4 входа и 1 выход. Существует 16 возможных комбинаций 4 битов, поэтому LUT содержит 16 бит конфигурационной памяти (таблицу истинности). По сути, LUT — это 16-канальный мультиплексор, который выбирает один из 16 сохраненных битов на основе 4 входных сигналов.

Фактчек: Стандартный LUT — 4 входа, 16 бит — △: Устарело с середины 2000-х: современные Xilinx/AMD используют 6-входовые LUT (64 бита конфигурации), Intel — 8-входовые ALM. Для объяснения принципа не критично.

Универсальность FPGA дается ценой 10-кратного перерасхода

транзисторов 59:04: В ASIC логическое И для 4 входов (4-way AND) реализуется всего 3 вентилями AND. В FPGA для этой же операции используется один LUT. Схемотехника самого LUT (16-канального мультиплексора) требует 16 вентиляй AND и 16 вентиляй OR — в сумме 32 вентиль. Это объясняет, почему FPGA на порядок дороже и прожорливее ASIC при той же вычислительной мощности.

Детерминированность, CPU, GPU и TPU

Кэш-память лишает CPU детерминированной задержки 1:03:28: В CPU аппаратное обеспечение само решает, брать данные из быстрого кэша или из

медленной DDR-памяти. Попадание в кэш зависит от фоновых процессов и истории операций, что делает время выполнения инструкций непредсказуемым. В TPU используется архитектура «scratchpad» (SRAM-память): софт явно вызывает разные инструкции для обращения к быстрой SRAM и к медленной HBM. Это гарантирует 100% детерминированность времени выполнения.

CPU тратят огромную площадь на предсказатели ветвлений 1:07:28:

Оценка условия ветвления (if/else), обновление счетчика команд и чтение новой инструкции занимает около 5 наносекунд (эквивалент частоты 200 МГц). Чтобы CPU мог работать на частоте 1-2 ГГц, предсказатель ветвлений угадывает путь за 5 тактов до самого ветвления и начинает выполнять инструкции авансом. GPU избавлены от этой сложной логики, что позволяет разместить на кристалле гораздо больше АЛУ и регистров.

Фактчек: Оценка ветвления в CPU занимает ~5 нс — □: Ошибка предсказания сбрасывает конвейер на 10–20 тактов, на 3–4 ГГц это 3–5 нс — согласуется.

Снижение тактовой частоты не дает кратной экономии энергии 1:11:34:

Основное энергопотребление чипа (динамическая мощность) уходит на зарядку и разрядку микроконденсаторов при переключении битов с 0 на 1 и обратно. Если снизить частоту чипа в 1000 раз, он совершит в 1000 раз меньше переключений и потратит в 1000 раз меньше энергии в секунду, но энергоэффективность на одну операцию почти не изменится. Поэтому чипы выгоднее запускать на максимальной скорости для высокой пропускной способности, а затем переводить в режим простоя.

GPU и TPU имеют принципиально разную макроархитектуру 1:15:41:

- **GPU** состоит из множества мелких одинаковых блоков (Streaming Multiprocessors, SM), в каждом из которых есть свои векторные и матричные (Tensor Cores) устройства. Это обеспечивает отличную локальную маршрутизацию данных внутри SM (через 16+ шин), но усложняет глобальную синхронизацию.

- **TPU** состоит из нескольких гигантских матричных блоков (MXU) и одного центрального векторного устройства. Это лучше амортизирует затраты на регистровые файлы, но создает узкое горлышко: данные между векторным и матричным блоками перемещаются всего через две линии периметра.

- Стартап MatX разрабатывает «расщепляемые систолические массивы» (splittable systolic arrays), чтобы объединить масштабы TPU с гибкостью локальной маршрутизации GPU.

ЛИЧНО ДЛЯ ТЕБЯ

Цену токена определяет движение байтов, а не флопсы: регистры → SRAM → HBM → сеть, каждый уровень дороже на порядок. Когда провайдер обещает «дешёвый инференс» или «чип в 10× быстрее», спрашивай про разрядность, batch и utilization, а не про пиковые TFLOPS; при малом batch узкое место — чтение весов и KV-кэша, при длинном промпте — вычисления. Сравнивай отдельно время до первого токена, скорость генерации и цену на нужном контексте.

FP4/FP8-квантизация — не маркетинг, а единственный физический способ снизить цену в разы, но качество проверяй сам: устойчивость модели к квантованию разная. Для своего AI-кодинга прогони одинаковые задачи на квантованной и полной версии: корректность патча, прохождение тестов, число повторных попыток важнее паспортного ускорения.

Дисциплина «фиксируй бюджет, потом торгуйся за каждый компонент»: логика 10%/90% и «пайплайн-регистр стоит 8 вентилях против 1 у операции» — антидот к «добавим гибкость на всякий случай». Гибкость (FPGA vs ASIC) стоит ~10× ресурсов и оправдана, только пока логика реально меняется. Переносится на архитектуру кода, смету стройки и личный бюджет.

Фильтр к выпуску: это питч CEO MatX: физика вентилях изложена честно, но линия «GPU плохо масштабируется, TPU негибок» подводит к продукту («расщепляемые систолические массивы»). Тезис «гони частоту и уходи в idle» не переноси на бытовые решения вроде undervolting ноутбука — там DVFS реально экономит энергию и нагрев.

КОНСЕНСУС РЕЦЕНЗЕНТОВ

Все трое сошлись: базовая физика (квадратичный рост умножителя, дороговизна перемещения данных, экономика FPGA vs ASIC, перерасход LUT) изложена корректно по порядку величины, а «7/8» и «10/90» — игрушечные оценки, не измерения. Разошлись по трём пунктам: Gemini подтверждает race-to-idle и 100% детерминированность TPU, Claude и GPT считают оба тезиса неполными (DVFS, задержки HBM и межчиповой связи); по B300 Gemini даёт 2× (цифры B200), Claude — 3× для Blackwell Ultra.