Конспект к экзамену по билетам (архаичные ЭВМ) 1-й семестр

Латыпов Владимир (конспектор)
t.me/donRumata03, github.com/donRumata03, donrumata03@gmail.com

Скаков Павел Сергеевич (лектор)
t.me/pavelxs

24 января 2022 г.

Содержание

1. Введение

Максимально сжатый материал: если читатель не знаком с курсом, возможно, стоит сначала изучить конспект Тимофея на Overleaf.

2. Названия билетов (ровно как в оригинале)

1. Устройство памяти
    1.1. Элементная база вычислительной системы: логические элементы, триггеры.
    1.2. Оперативная память: статическая/динамическая, организация.
    1.3. Оперативная память: характеристики, типы динамической памяти. NUMA.
    1.4. Кэш-память.
    1.5. Протоколы когерентности кэш-памяти.
    1.6. Носители информации: магнитные, оптические и на основе флеш-памяти. RAID.
2. Архитектура процессорных систем
    2.1. Архитектура фон Неймана и её альтернативы.
    2.2. Архитектура набора команд (ISA) и микроархитектура.
    2.3. Конвейерная архитектура. Конвейер MIPS.
    2.4. Проблемы конвейера (hazards) и пути их решения.
    2.5. Суперскалярная и VLIW архитектуры. Спекулятивное исполнение. Уязвимости классов Spectre и Meltdown.
    2.6. Многоядерные/многопроцессорные системы, одновременная многопоточность (SMT/HT).

Нужно ответить на два вопроса: по одному из каждой части. Пользоваться ничем нельзя, отвечать сразу.

3. О чём говорить при каждом из билетов?

3.1. Элементная база вычислительной системы: логические элементы, триггеры.

Европейские и американские обозначения логических элементов

Полусумматоры и сумматоры (медиана + XOR или два полусумматора + OR)

RS-триггер, каноническая и енаиболее эффективная версия — через два ИЛИ-НЕ.

Проблемы высоких частот и малого размера, «В Ethernet соотношение сигнал/шум — как разговаривать рядом с турбиной самолёта»

$\Longrightarrow$ Синхронная версия RS-триггера, D-триггер через «не» на входе.

3.1.1. MEM

Декодер 3to8 (для каждого из 8 выходов AND от трёх, возможно, инвертированных входов) $\Longrightarrow$ мультиплексор ($3 + 8 \to 1$) и демультиплексор ($3 + 1 \to 8$).

Из этого можно сделать модуль памяти «Mem» на 8 бит:

Входы: 3 адресных бита, R/W, C (clock), D (запись, если выбран режим W).

Выход: «Q» — если выбран режим R (не обязательно только в этому случае) — на нём значение, соответствующее биту по адресу $\overline{A_0 A_1 A_2}$.

3.1.2. JK-триггер

JK-триггер (Входы J, K, синхронный, $\Leftarrow$ умеет инвертировать состояние при двух единицах). Крафтится из двух RS-триггеров с тактированием в противофазе, на вход первого кроме оригинальных входов J, K через «AND» даётся то, что запомнил второй (то есть что было на первом до начала такта). В нормальной версии результат (Q и $\tilde{Q}$) берётся из выходов первого триггера.

3.1.3. Физические основы работы логических элементов

Нас интересуют в основном транзисторы. По многим причинам победила $CMOS$-логика, конструирующая только транзисторы (причём

полевые) для конструкции элементов. Это позволяет минимизировать потребляемую энергию, так как ток течёт только при переключении. В отличие от полевых транзисторов (где он нужен через базу для поддержания открытого состояния), а также $NMOS$ логики, где он течёт по резистору в некоторых случаях (когда транзистор открыт).

В случае $CMOS$ логики расход энергии растёт при увеличении частоты почти линейно по понятным причинам.

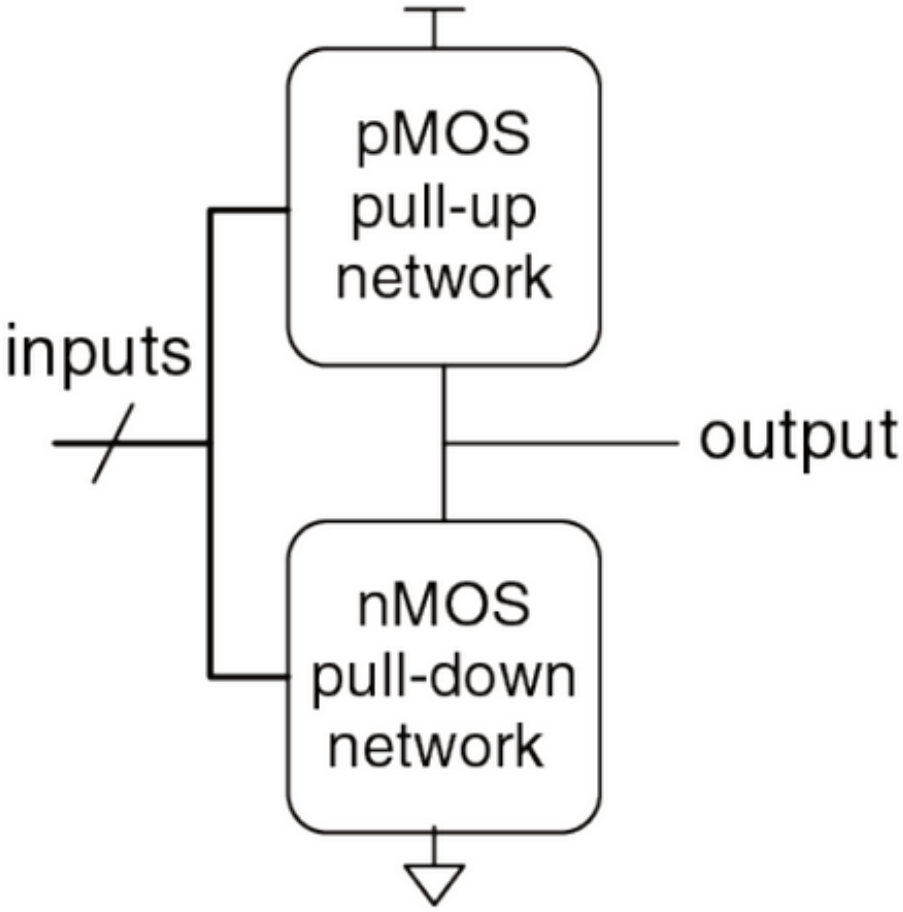

Рис. 1: Общий принцип построения схем в CMOS логике

В качестве элементарных частиц используем $NOT, NAND, NOR$ (нельзя сделать AND и OR эффективнее, чем соответствующий $NAND/NOR \circ NOT$)

Fun Fact: можно сделать NOR и $NAND$ на много входов эффективнее,

чем просто внешне композировать (например, для трёх — 6 вместо 8-и).

3.1.4. Дребезг контактов

Кнопки из реальной жизни при нажатии не сразу устанавливается в новой состояние, а сначала колеблется. Это назвается Contact bounce (дребезг контактов). Причём чем старше и некачественнее контакты, тем дольше будет происходить bouncing.

Есть несколько подходов к борьбе с ним. Бывает, программно, бывает аппаратно. Причём криме вопросов реализации возникают ещё и концептуальные. Проще всего реагировать на нажатие с запозданием: по истечении времени с начала или с последнего изменения. Другой вариант — реагировать как только произошло изменение после затишья, но после этого игнорировать дребезг, пока не установится. Однако, если у нас не просто бинарная кнопка с одним контактом «нажат/не нажат», а имеющая положения «не нажата, положение 1 и положение 2» и мы хотим на выход подавать бинарный сигнал в виде последнего «кас´анного» контакта, то всё гораздо очевиднее и у нас есть ультимативное решение через триггер и, возможно, транзистор. Не забыть про подтягивающие резисторы.

3.2. Оперативная память: статическая/динамическая, организация.

Папмять располагают в 2D решётчатой структуре.

Количество проводов $\propto cols + rows \Rightarrow O(\sqrt{memory\_size})$

Row отвечает за выбор ряда. Если он ноль, ячейка вообще не работает.

Если подаётся $true$, ячейка открывается. Можно считывать информацию с соответствующих Col и $\overline{Col}$. Также можно записывать на эти входы.

Статическая память строится как

	Статическая память	**Динамическая память**
Скорость	быстрая	медленная
Количество транзисторов	Много (=6)	Мало (обычно один + кондерсатор)
Надёжность	Наличествует	Постоянно дегенерирует $\Rightarrow$ надо регенерировать
Примеры	Регистры, кэш	Оперативная память, видеопамять

Параметры, по которым оценивается память:

Объём,

скорость доступа («latency»),

скорость передачи («throughput», пропускная способность)

Уязвимость Raw-Hammer: если очень долго обращаться к ячейкам, соседним с данной, можно установить её в ноль.

3.3. Оперативная память: характеристики, типы динамической памяти. NUMA.

Порядок работы с памятью:

1. Процессор сообщает о таймингах
2. Всё время идет синхронизация
3. Выбор строки $\Rightarrow$ модуль динамической памяти заносит строку в статическую
4.