СЕТЬ 2030

Синхронная
Символьная
Иерархия

Балыбердин А.Л
Rutel.@mail.ru

НСКФ
НАЦИОНАЛЬНЫЙ СУПЕРКОМПЬЮТЕРНЫЙ ФОРУМ

Авторское право

Все идеи и алгоритмы, описываемые в данной презентации, являются результатом моей независимой и полностью самостоятельной интеллектуальной деятельности.

Как автор, разрешаю свободно использовать, изменять, дополнять все идеи и алгоритмы любому человеку или организации в любых типах проектов при обязательном указании моего авторства.

rutel@mail.ru

Решаемые проблемы

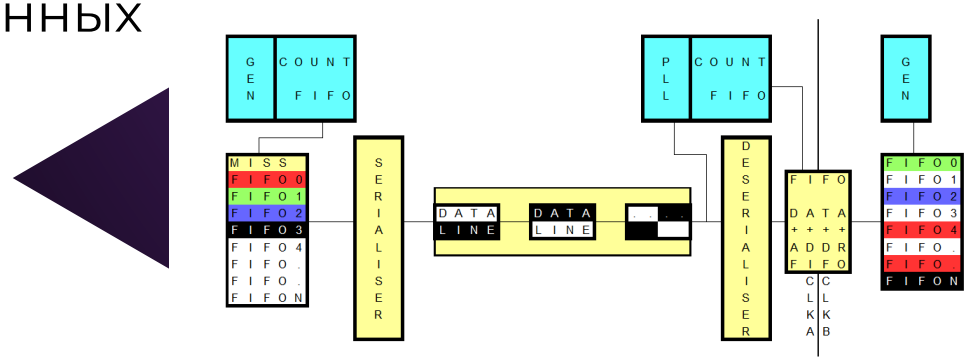
01 В 2018 году 13-й Исследовательской комиссией (ИК-13) МСЭ-Т была создана фокус-группа технологий Network 2030 (FG NET-2030), группа изучила возможности фиксированных сетей связи на период до 2030 года. Предполагалось, что "Сеть-2030" может быть построена на новой или усовершенствованной сетевой архитектуре, которая может развиваться из современных сетей или сильно отличаться от них.

02 Прошу рассмотреть и принять за основу перспективных сетей результаты моих работ. Предлагается сеть с коммутацией каналов, с рабочим названием «Синхронная Символьная Иерархия» (ССИ). Сеть ССИ полностью соответствует всем базовым требованиям, сформулированным к перспективным сетям «Сети 2030».

03 Новая сетевая архитектура позволяет прозрачно для пользователя заместить все существующие на данные момент сети, а также различные локальные интерфейсы (в том числе низкоуровневые). Данный документ содержит только телекоммуникационную составляющую проекта, но есть еще и вычислительная составляющая (DataFlow вычислительные системы).

Устройство типичного линка

Практически все высокоскоростные соединения построены по единому принципу. Блок синхронизации, блок выбора FIFO буфера с данными для передачи, Блок линейного кодирования и серилизации, Кабель связи, Блок восстановления тактовой частоты, Блок десерилизации, Блок вычисления адреса записи принятых данных, Блок FIFO для перехода через границу с различными тактовыми частотами, Блок FIFO для хранения принятых данных



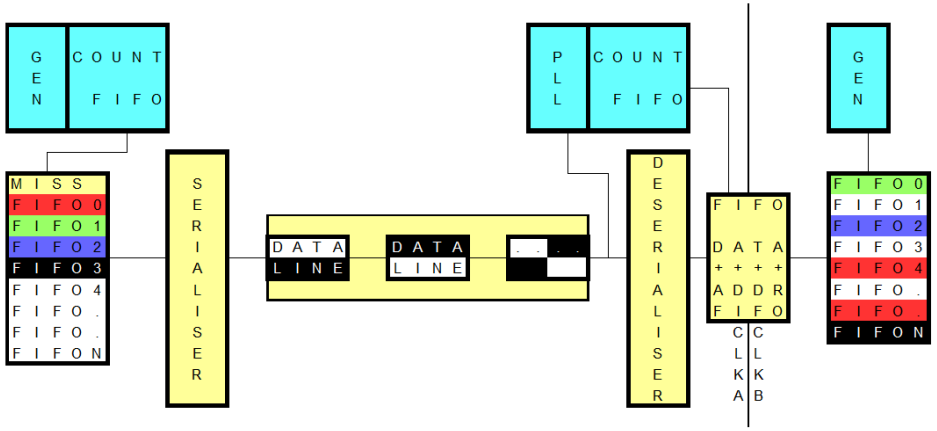
Данные передаются синхронно, что делает такой линк пригодным для использования в перспективной Сети 2030.

Данные передаются порциями, называемыми символами. Размер символа в битах определяется соотношением скорости передачи данных в канале и тактовой частоты коммутатора. Кроме того символ включает различные данные линейного кодирования, корректирующие и служебные коды.

Проблема «проскальзывания»

Плюсами синхронных систем передачи и коммутации являются большие скорости передачи данных и малые размеры буфера для промежуточного хранения передаваемых данных.

Но есть и минусы, один из них это эффект проскальзывания. Данный эффект проявляется на границах зон с различной тактовой частотой. Сделать полностью идентичные генераторы тактовой частоты невозможно, рано или поздно они начнут разбегаться. Для передаваемых данных это выразится в переполнении или опустошении промежуточных буферов



Для решения этой проблемы, предлагаю использовать следующий алгоритм: Тактовая частота приемника и передатчика примерно равны (плезиохронны). Из передаваемых символов выделяем одну комбинацию бит и закрепляем за ней свойство «нет данных». Передатчик при генерации потока символов автоматически и равномерно вставляет такой символ на каждые две тысячи обычных, что компенсирует неточность генераторов до +-250ppm. На приемной стороне данный символ просто отбрасывается.

Можно ввести символ «ошибка передачи» и вставлять его при ошибках декодирования символа на приемной стороне,такой прием позволит детектировать конкретный символ содержащий ошибку.

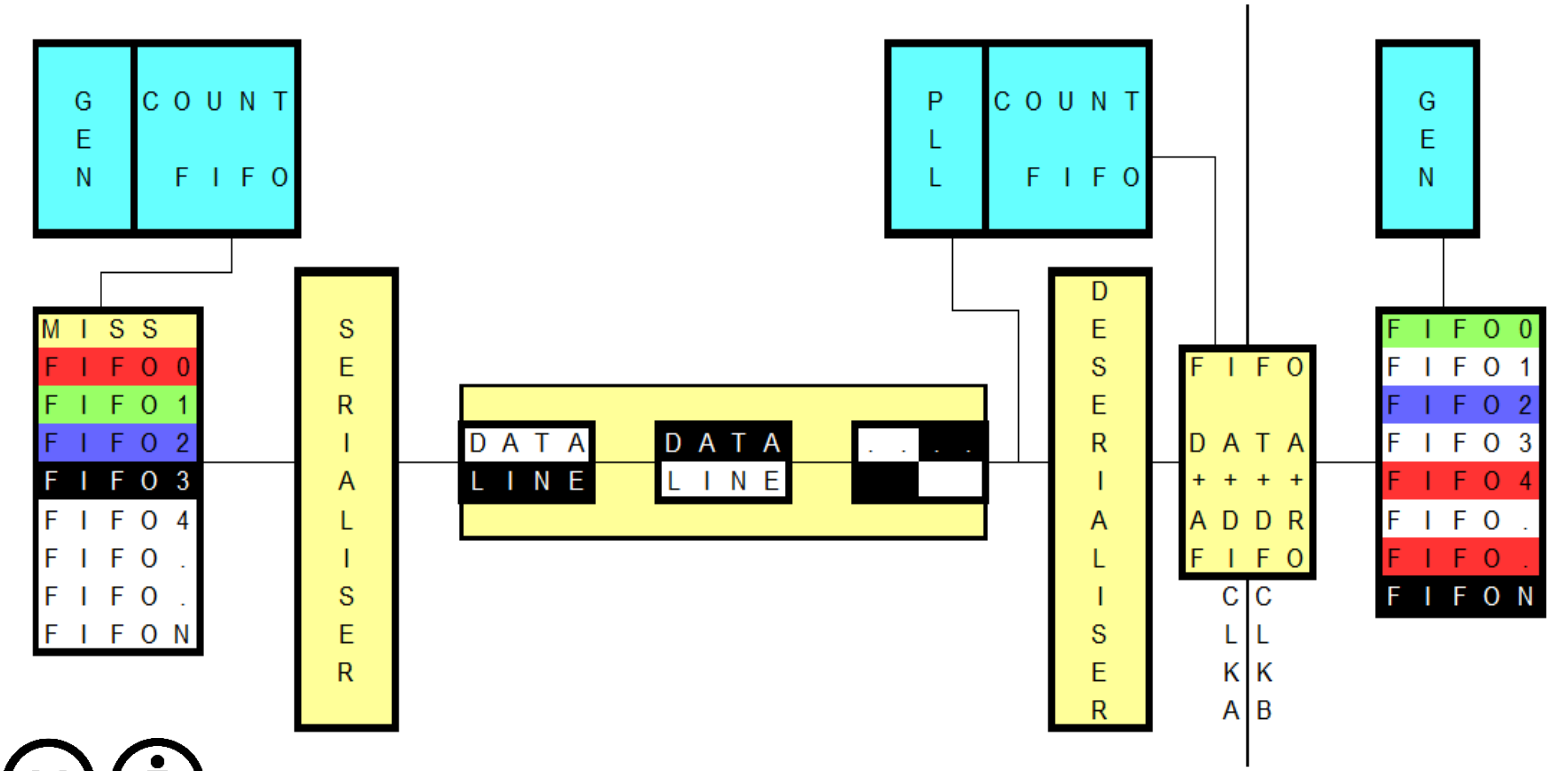
Плюсы компенсации «проскальзывания»

Плюсы получаемые при использовании символа «нет данных». Добавление данного символа дает гарантию того, что на приемной стороне (для промежуточных коммутаторов тоже) не возникнет ситуация при которой произойдет переполнение приемного буфера

Да, есть уменьшение КПД использования пропускной способности канала, но 0.05% это не та величина из-за которой стоит «переживать»

Кроме того в промежуточных буферах никогда не будет больше чем 1-2 символа, соответственно гарантированно достаточный размер буфера будет те же 1-2 символа (для любого числа промежуточных коммутаторов)

Среднее число данных в промежуточном буфере будет 0.5 символа, что гарантирует среднее время на коммутацию на уровне 50% от периода передачи (но не более 2х периодов)



Создание виртуальных каналов

Требуемое число одновременных сеансов связи в процессе работы оборудования может превышать триллионы, что на много порядков превышает число физических линий связи. Необходим механизм разделения одного физического канала между большим числом процессов.

Сформулируем требования к такой системе:

- Сохранение свойств достигнутых на этапе решения проблемы проскальзывания
- Мгновенное создание канала без предварительного уведомления приемника
- Отсутствие влияния ошибок создания виртуального канала как на уже созданные, так и на последующие создаваемые каналы
- Создание виртуального канала с произвольной и точно заданной скоростью, как с постоянной так и с асинхронной ее частью
- Отсутствие взаимного влияния уже созданных каналов

Алгоритм разделения физического канала

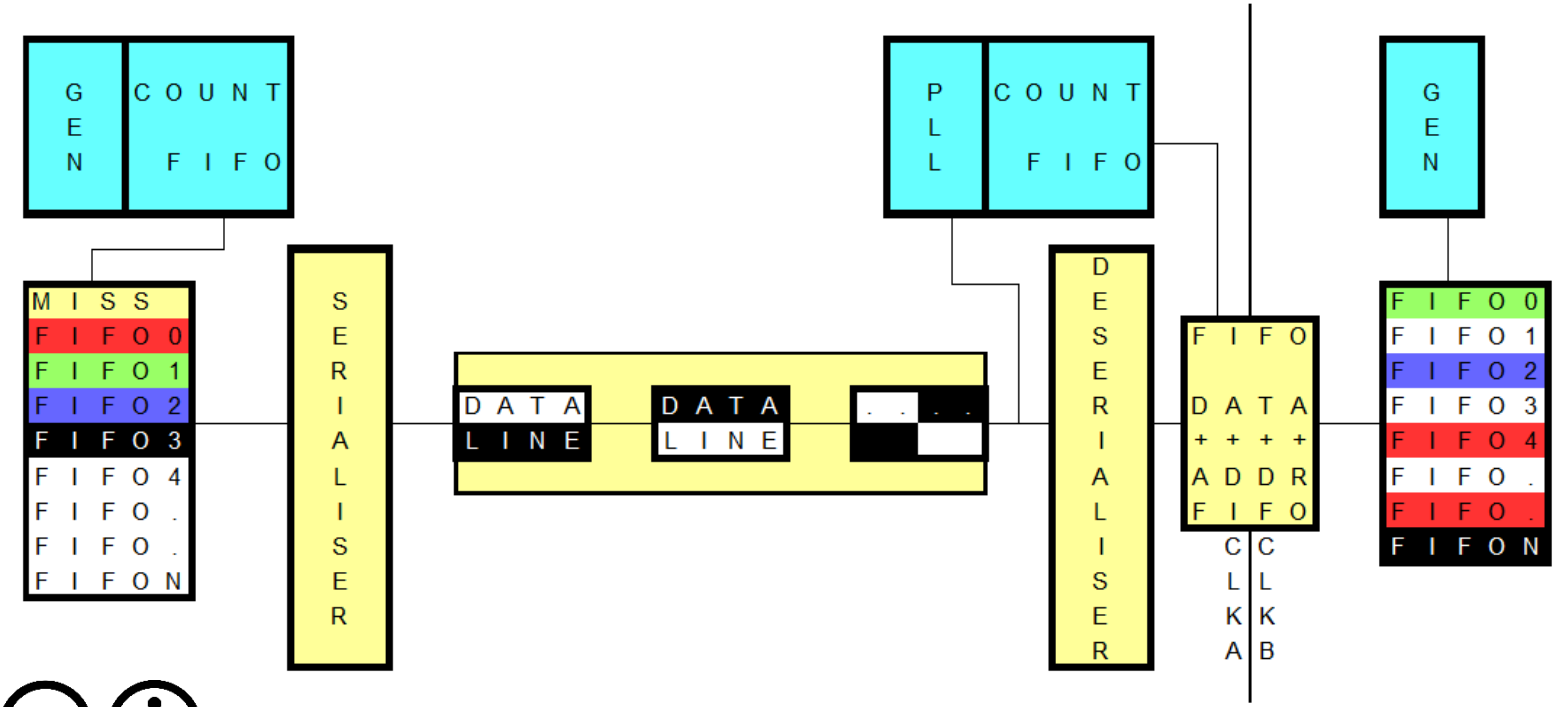
Очевидно, что разделяемый физический канал должен иметь некоторую структуру и необходим синхронизированный для приемника и передатчика указатель на текущее положение в пределах этой структуры

В качестве указателя можно использовать циклический счетчик передаваемых символов

Для синхронизации можно использовать символы «нет данных», размер символа достаточно большой и часть битов можно использовать для передачи значения счетчика.

В момент передачи в эти битовые позиции копируется значение счетчика на передающей стороне

При приеме происходит сравнение принятого значения с локальным счетчиком и если они различаются, то производить коррекцию



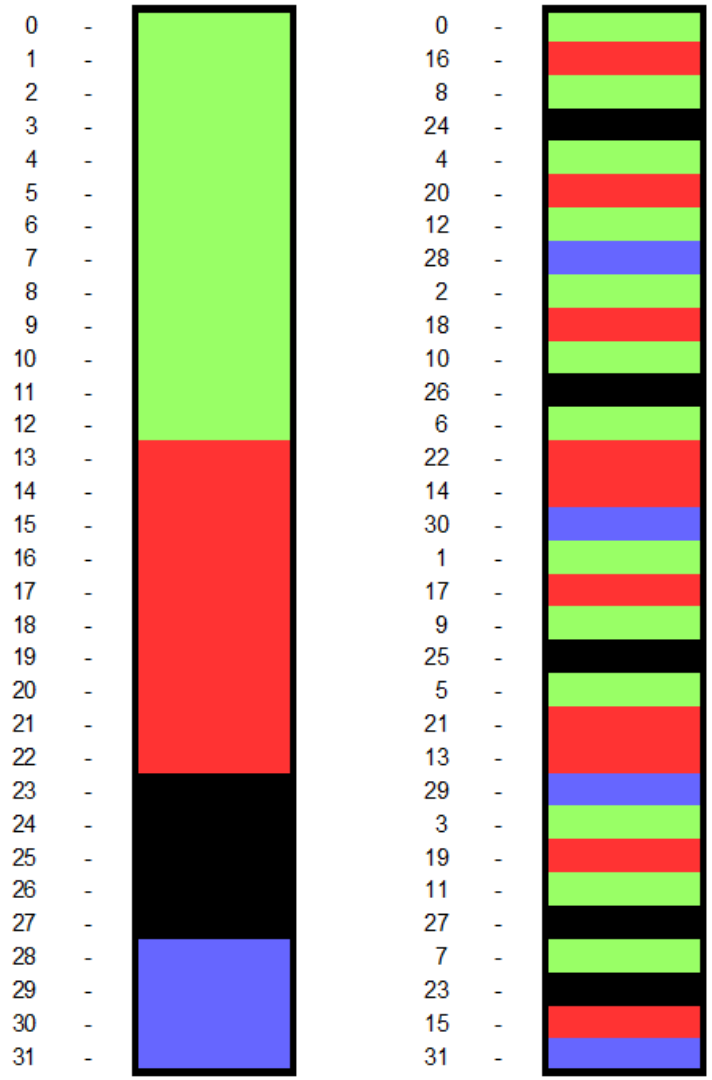
Алгоритм равномерного опроса виртуальных каналов

Виртуальные каналы в своей основе должны быть синхронными. Для получения таких каналов используется механизм TDM (временное разделение), но у этого принципа есть проблема: виртуальные каналы получаются или с одинаковой скоростью (E1-E3) или в виде пачек с некоторым достаточно большим периодом передачи (SDH)

Для соблюдения принципа равномерности передачи в каждом виртуальном канале нужно добиться отклонения не более двух периодов передачи, причем с гарантированным распределением во времени позволяющим использовать FIFO буфера с размером менее двух символов.

В предыдущем пункте были созданы счетчики и механизм поддержания их согласованности с передаваемой последовательностью символов

Счетчики имеют некоторый период (для конкретного случая будем считать что это число символов в секунду). Счетчик принимает значение от нуля до число символов в секунду минус один, представляет собой эквивалент TDM с тайм-слотом в один символ в секунду



Алгоритм равномерного опроса виртуальных каналов

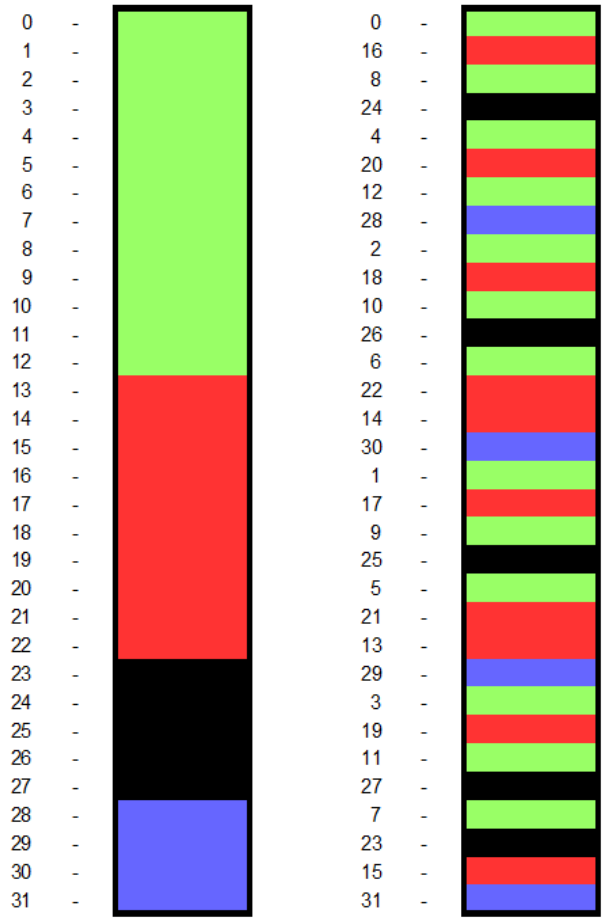
Каждый виртуальный канал занимает в этом пространстве некоторое число смежных комбинаций (ячеек)

Число комбинаций пропорционально требуемой синхронной составляющей скорости виртуального канала (похоже на выделение оперативной памяти в обычном компьютере)

Значение счетчика на момент приема символа можно использовать для адресации конкретного FIFO. Каждый буфер откликается на некоторый, заданный при создании канала, диапазон значений

У такого способа есть проблема: Данные будут передаваться пачками с периодом в секунду, что недопустимо. Необходим алгоритм преобразования значения в счетчике в номер FIFO с гарантированным равномерным распределением во времени при любом варианте разбиения исходного физического канала на составляющие.

Был найден простой алгоритм: достаточно поменять старшие биты счетчика с младшими (зеркальное отображение) и будет происходить равномерное обращение в любому заданному непрерывному участку значений исходного счетчика с периодом равным отношению символьной скорости физического канала к ширине данного участка (требуется доказать математически во всем диапазоне скоростей и вариантов разбиения).



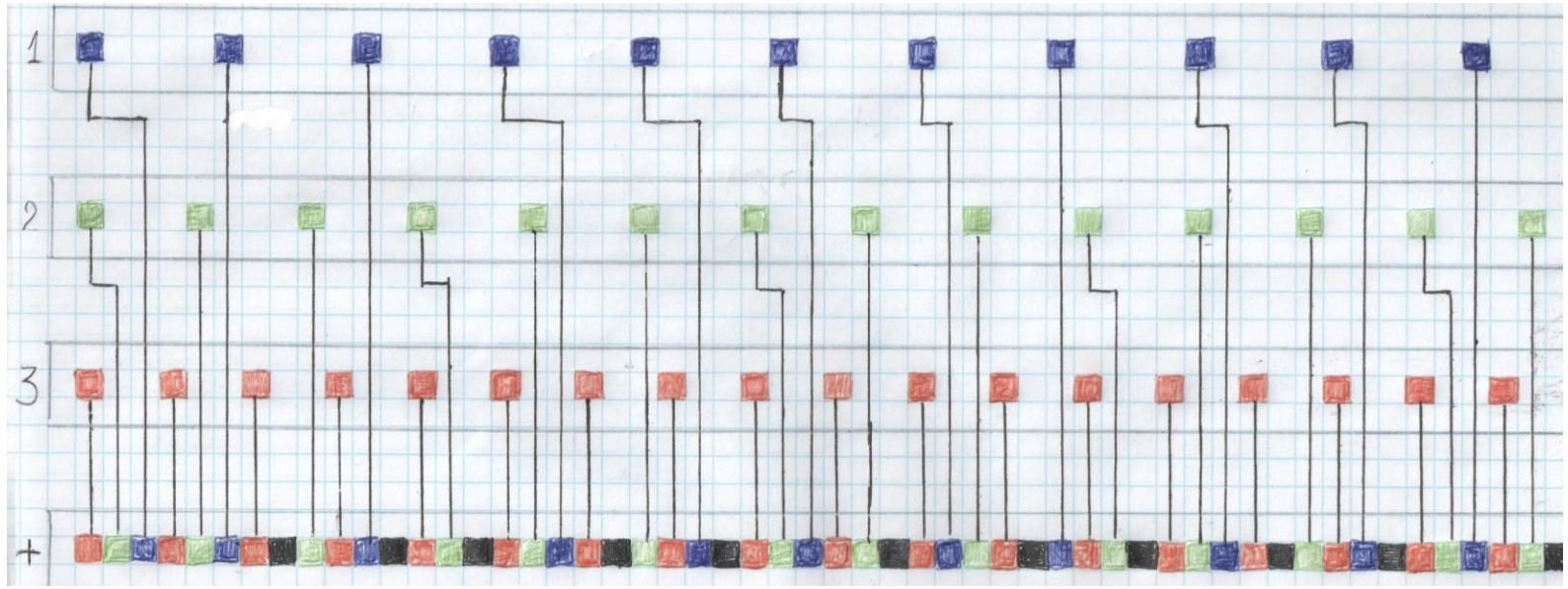
Данный способ совершенно не чувствителен ошибкам создания канала, когда запрос на создание канала не дошел до приемника (ошибка передачи, декодирования). В таком случае данные будут занимать тайм-слоты не относящиеся к какому либо FIFO и контроллер должен их просто игнорировать.

Алгоритм частотного метода виртуальных каналов

Можно выделять пропускную способность используя другой способ. Для этого необходимо отсортировать список запросов на создание виртуальных каналов по возрастанию требуемой скорости

Добавлять в канал данные по истечению периода передачи данных каждого виртуального канала (принцип иллюстрирован на картинке)

Данный алгоритм менее эффективен, чем первый и требует специальных мер защиты от потерь запросов на создание канала.

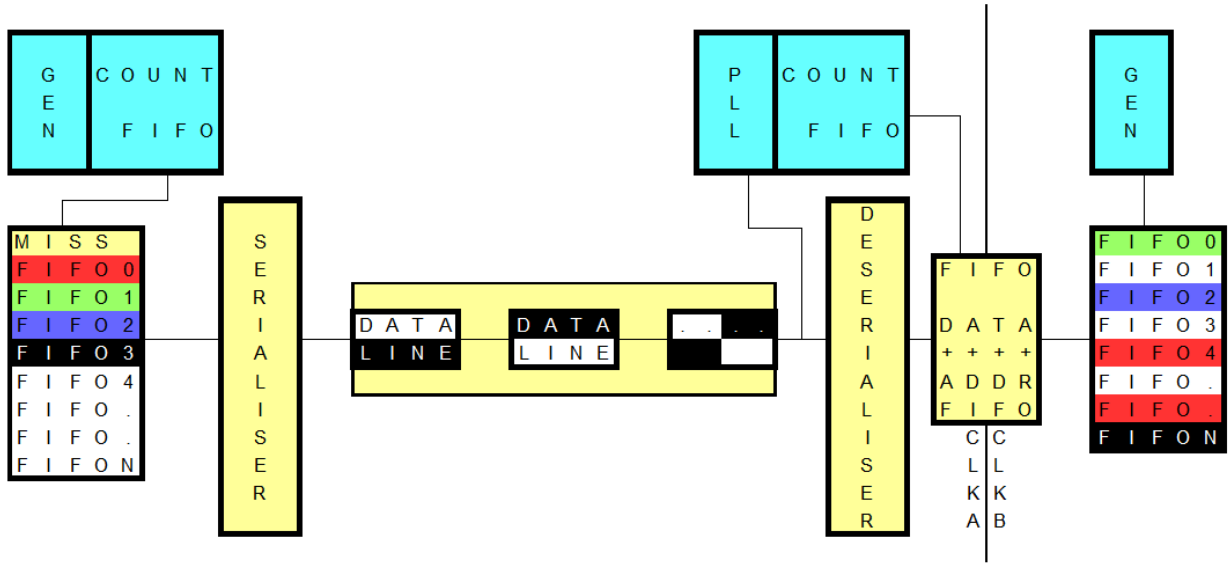


Создание, удаление канала

Передаваемые символы разделены на большие группы, среди которых есть: Пользовательские данные (синхронный трафик) и Данные управления коммутатором (асинхронный трафик)

Данные управления коммутатора передаются с использованием не задействованной пользователем полосы пропускания физического канала, причем можно использовать не только постоянно незадействованную часть физического канала, но и локальную (в данный конкретный момент)

Канал данных управления коммутатором является асинхронным или «источником асинхронной составляющей» Если в момент чтения пользовательских данных из FIFO выясняется что данных нет, то производится вставка символа «нет данных». Этот символ необязательно передавать в канал, можно заменить на служебный трафик управления коммутатором.



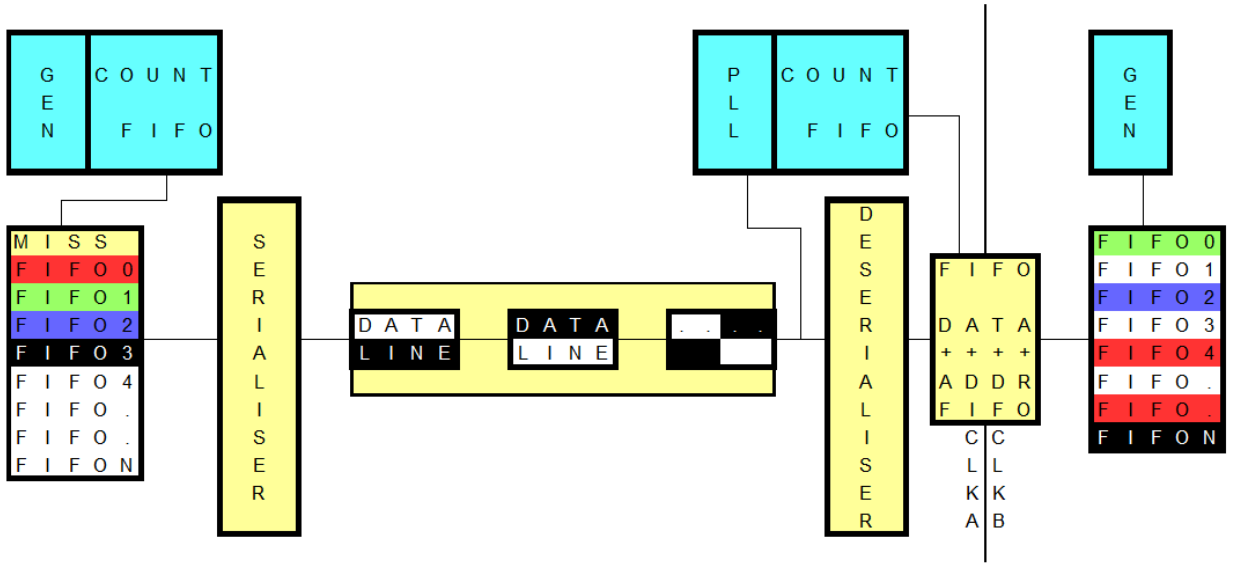
Создание, удаление канала

На приемной стороне произвести обратную замену, все служебные символы заменяются на «нет данных» перед записью в FIFO приемника. Такой подход позволяет поднять КПД использования физического канала до 98% (оставшиеся 2% используются для указания типа данных)

Получаем один быстрый канал с данными управления коммутатором, в нем можно передавать запросы на создание виртуальных каналов и асинхронную составляющую пользовательского трафика

После передачи запроса отправляем служебный символ «смена конфигурации» и дальше идут данные в которых уже присутствует новый канал

Удалять канал лучше добавлением пользовательского служебного символа «удалить канал», после передачи его в физический канал должно произойти освобождение FIFO, буфер перестанет реагировать на свой диапазон адресов (будет передаваться символ “нет данных”)



Маршрутизация, разветвление, резервирование каналов

Маршрутизация может быть любой (абсолютной или относительной), в качестве базовой выбрана относительная потому, что проста в реализации.

Адрес приемника представляет собой последовательность выходных портов коммутаторов, через которые должен быть проложен создаваемый виртуальный поток что бы достигнуть приемника

Если в каждом коммутаторе будет назначаться ровно один порт, то получится канал связи точка-точка

Если есть возможность перенаправить поток одновременно в два выходных физических канала, то получаем структуру типа дерева (один передатчик — много приемников). На предыдущем слайде видно один красный виртуальный канал (передающая сторона) превратился в два канала на приемной стороне. Такой эффект достигается достаточно простым методом: Больше чем один буфер FIFO откликается на один диапазон адресов (на запись)

Происходит запись в несколько буферов, которые в последующем считываются для передачи в различные виртуальные каналы (в том числе и в различные физические каналы)

Возможность управлять конкретным маршрутом (относительная адресация) позволяет гарантировать, что несколько каналов с передаваемыми данными никогда не будут переданы в одном физическом канале. В совокупности с возможностью разветвления виртуального канала, получаем технологию многократного дублирования канала связи с пропорциональным увеличением надежности работы виртуального канала

Создание иерархии виртуальных каналов

Если в момент распределения тайм-слотов выделить для нескольких виртуальных каналов смежную область для записи, а для чтения ту же область читать как единый канал, то получим виртуальный канал содержащий несколько исходных возможно одиночных каналов.

Далее полученный суммарный канал можно транспортировать используя всего один канал FIFO, а не несколько (по числу исходных каналов).

После передачи производим обратную процедуру и снова получаем много исходных каналов. Принципы работы с суммарным каналом не сильно отличаются от основного (физического канала).

Асинхронная передача данных

При создании виртуального канала передачи данных могут быть востребована и асинхронная составляющая скорости. Например, есть процесс отправляющий данные в заранее неизвестные моменты времени, частично известна только средняя скорость передаваемых данных за достаточно большой промежуток времени

Для максимальной эффективности требуется максимально быстро (с заданной латентностью) доставлять эти данные. Если есть только синхронные (с постоянной скоростью) каналы передачи данных, придется выделить каждому такому процессу пропускную способность пропорциональную затребованной латентности, при этом выделенный канал не будет полностью загружен

Для понимания от каких факторов зависит асинхронная пропускная способность:

- Исходный канал занят синхронными каналами, остается не занятая область и пропуски в предоставлении синхронных данных. Наличие этих ресурсов практически невозможно прогнозировать заранее и это слабая сторона таких асинхронных сетей как пакетная коммутация
- С другой стороны, для использования мгновенных пропусков (пауз в передачи данных) необходим буфер в котором будут находиться данные, в количестве сверх того что требуется для синхронной передачи
- Для создания виртуального канала с синхронной и асинхронной составляющей необходимо занять ровно столько синхронной пропускной способности, сколько в среднем потребуется на длительном интервале времени

Асинхронная передача данных

- Для обеспечения асинхронной составляющей, увеличить размер буфера FIFO пропорционально отношению синхронной и асинхронной составляющей скоростей.
- В момент, когда нет данных для передачи (буфер пуст или не занятые тайм-слоты), можно используя служебную часть набора символов передавать данные из буферов каналов имеющих не нулевое значение асинхронной скорости.
- Для передающей стороны можно выделить два варианта поведения. Первый это передать за один раз число символов равное размеру выделенного FIFO и ждать уведомления о приеме, полностью асинхронный режим хорошо работающий на малых расстояниях (дата-центр, суперкомпьютер).
- Для больших расстояний (более 100 км) выгоднее пытаться передавать данные без ограничений скорости, но это будет вызывать переполнение FIFO буферов промежуточных коммутаторов. При таком способе передачи, FIFO, где происходит переполнение, должно заменять последний символ в буфере и символ который пытаются записать на служебный символ «Переполнение данных» с счетчиком числа таких данных
- На приемной стороне происходит анализ принятой последовательности и запрос повторной передачи потерянных данных.
- Передатчик, получив такой запрос, уменьшит скорость передачи и повторно передаст затребованные символы (похожий механизм используется в TCP IP).

Интерфейс с вычислительной системой

Как может выглядеть интерфейс сети с различными вариантами вычислительных устройств?

Где должен находиться интерфейс сеть-вычислитель: Тут все просто, самая удобная часть это коммутатор. Коммутатор представляет собой большое число одинаковых FIFO, доступ к которым осуществляется через модули физических каналов

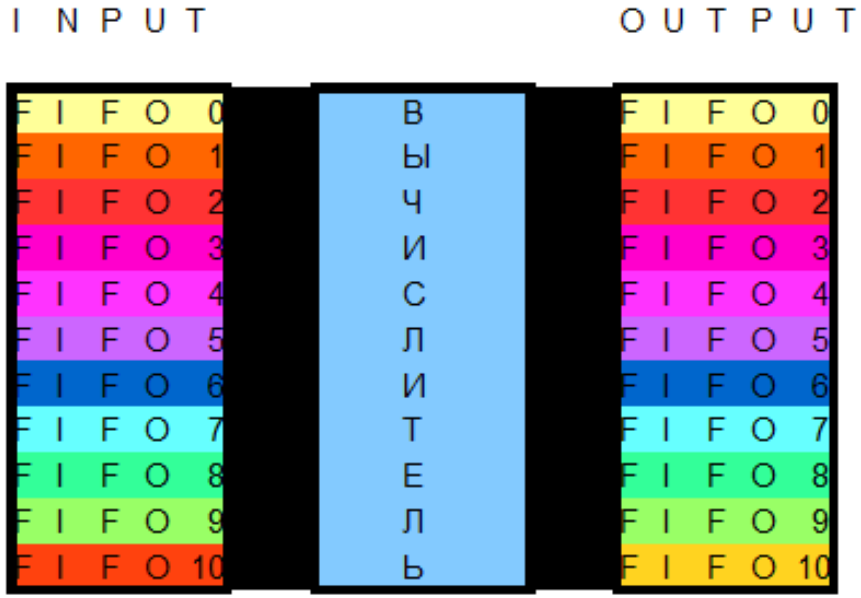
Наиболее эффективный способ это «мимикрировать» под еще один физический канал

Самый простой вариант, это отключить входы или выходы отдельных FIFO от интерфейса к физическим каналам и осуществлять запись (чтение) данных аппаратными средствами конкретного вычислителя

Для ASIC, FPGA и чего то подобного все очень просто, FIFO это по факту регистры с синхронным управлением которые очень просто стыкуются с подобными типами устройств

Для вычислителей имеющих интерфейс по типу памяти (адрес, данные), выгоднее «мимикрировать», выделить в адресном пространстве некоторую область, назначить каждому FIFO свой адрес и если есть возможность, то реализовать специальные команды управления доступом

Кроме того можно добавить сервис прерываний, вызывающий обработчик (программа) при появлении данных в нужных FIFO или при угрозе переполнения и тд



Дезагрегация оперативной памяти (DSM)

Как примерно может выглядеть технология дезагрегации оперативной памяти (на материнской плате останется только процессор с КЭШ памятью и чиплетами сети передачи данных)

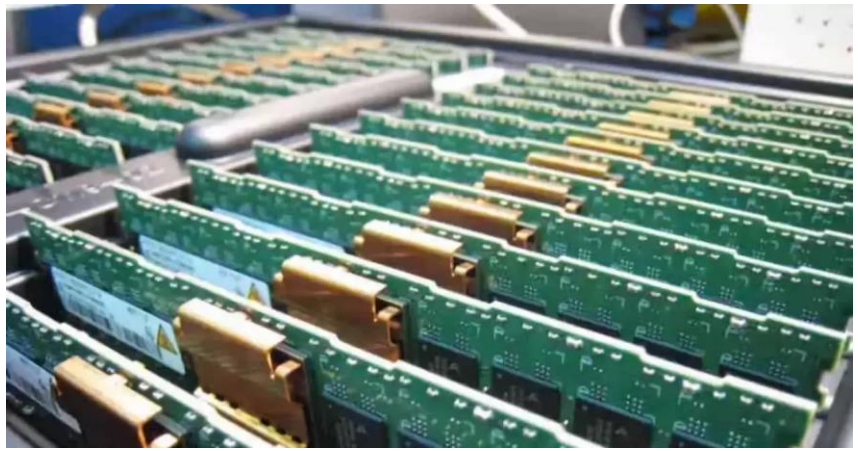
Оперативная память имеет время доступа 50-70 нс, для того что бы «не чувствовать» разницы между локальной и удаленной памятью, необходимо обеспечить похожие задержки

Считаем, что сервер оперативной памяти выполняет кэширование (примерно как кэш-память четвертого уровня) и в 99% случаев обращение идет именно к кэш памяти

Для удобства, считаем что процессор состоит из отдельных модулей расположенных в трехмерном кубе с соединениями между ближайшими соседями (100 мм + размер кристалла 60мм)

В центре этого куба находится сервер оперативной памяти. Время коммутации 1 нс (думаю реально достижимое при тактовой частоте коммутатора в 5 ГГц). Получаем примерное время передачи сообщения в соседний модуль на уровне 2 нс (1 нс коммутация 1 нс кабель связи). Чтение данных из кэш памяти — 10нс

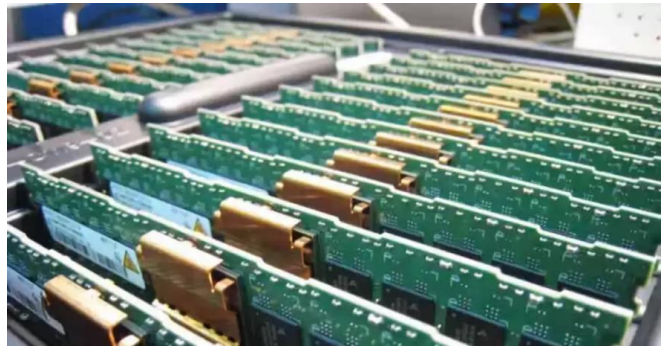
Несложные расчеты показывают расстояние, при котором нет разницы между локальной памятью и внешней, равно примерно 5 метров. Сторона куба при этом равна 1.6 метра, что по объему сопоставимо со стандартной стойкой для телекоммуникационного или вычислительного оборудования



Учитывая максимальную связность с соседними элементами и чтение большими страницами, накладные расходы на доступ к памяти будут меньше чем задержки доступа к локальной памяти

Дезагрегация оперативной памяти (DSM)

Как может выглядеть примерный сценарий доступа к памяти:



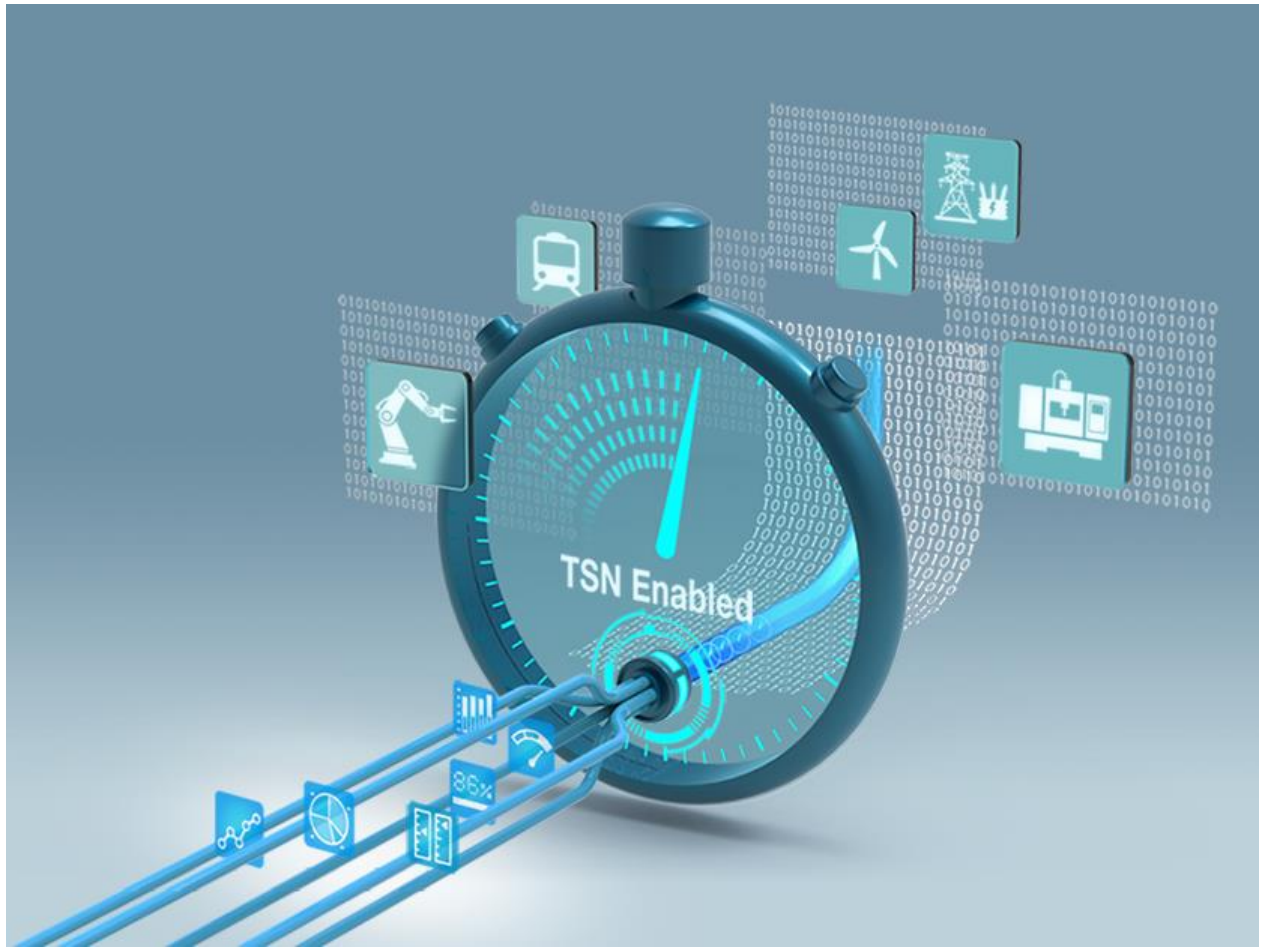
- Есть удаленный сервер оперативной памяти, процессор создает виртуальный канал с большой асинхронной составляющей по скорости
- Сервер в ответ подключает процессорный элемент к единому синхронному каналу (с большой синхронной скоростью)
- В момент начала работы высылается запрос на заполнение локального КЭШ. Происходит передача страницы (например 4К), ее одновременно видят и все остальные «подписавшиеся» на этот участок памяти
- При необходимости чтения, читаем из локальной памяти. При необходимости записи, отправляется запрос на модификацию ячейки памяти серверу DSM. Если нет необходимости последующего чтения модифицированной ячейки, то вычислительный процесс продолжается. В противном случае, процессор останавливается и ожидает уведомления об обновлении содержимого ячейки по высокоскоростному синхронному каналу.
- Данный вариант распределенной памяти DSM гарантирует, что каждый элемент системы будет видеть одинаковую последовательность изменений содержимого разделяемой памяти.

Телекоммуникационная сеть для систем реального времени

За счет гарантированной скорости, заранее вычисляемой и неизменной максимальной латентности, возможности копирования данных, управления маршрутом исполняются все требования систем реального времени

Можно сказать, что предлагаемая архитектура сети ССИ является вообще первой и пока единственной реализацией всех требований сети реального времени

Все остальные или являются ограниченно функциональными (TDM) или вообще не являются сетями реального времени (любые пакетные сети)



БЛАГОДАРЮ ЗА ВНИМАНИЕ, ВОПРОСЫ ?

Для получения дополнительной информации можно связаться:
Балыбердин Андрей Леонидович **rutel@mail.ru**