Вопросы

1. Без сильного увеличения аппаратных затрат — что и почему?

В данном случае сравнение происходило между двумя видами параллелизма. В пространственном параллелизме используется несколько копий аппаратных блоков, так что в одно и то же время можно выполнять несколько задач. Временной параллелизм предполагает разделение задачи на несколько стадий (или ступеней), как это происходит на сборочном конвейере.

В основе пространственного параллелизма лежит идея разделения вычислительной нагрузки на несколько частей, которые выполняются одновременно на различных вычислительных ресурсах, таких как ядра процессора или вычислительные блоки графических процессоров (GPU).

Пространственный параллелизм часто используется в задачах, которые могут быть разбиты на независимые подзадачи или операции, которые могут быть выполнены параллельно.

Для его реализации естественно необходимо использовать кратно больше ресурсов, как например добавление в процессор более чем одного ядра, что влечет за собой много дополнительных доработок и изменений.

2. Сохранить прежнюю латентность — что это значит? Пояснить

Токен - группа входов, которая обрабатывается для того, чтобы получить группу выходов. Задержка, или латентность (latency), системы – время, которое необходимо для прохождения одного токена через всю систему с ее входа на выход. Пропускная способность (throughput) – количество токенов, которое может быть обработано системой в единицу времени.

Пропускная способность может быть увеличена путем обработки нескольких токенов в одно и то же время. Это называется параллелизмом и используется в двух формах: пространственной и временной. В данной задаче речь идет именно о временном параллелизме. Временной параллелизм предполагает разделение задачи на несколько стадий (или ступеней), как это происходит на сборочном конвейере. Несколько задач могут быть распределены по стадиям. Хотя все задачи должны пройти по всем стадиям, разные задачи в любой заданный момент времени будут находиться на своей стадии, так что несколько задач могут одновременно обрабатываться на разных стадиях. Данный вид параллелизма называется конвейеризацией.

Разбив нашу схему на стадии, мы столкнемся с одной проблемой: для того, чтобы наша схема корректно работала мы должны выбрать правильную тактовую частоту. Данная частота очевидно будет равна самому долгому промежутку новой схемы. Например, если начальная задержка была 10нс, и мы разбили ее на две стадии 4нс и 6нс соответственно, то мы должны будем использовать тактовую частоту не менее 6нс. Тогда итоговая задержка новой схемы будет равна 6 + 6 = 12нс. Однако если сможем разделить нашу схему так, чтобы задержки на стадиях была равны 10 / 2 = 5, то используя частоту в 5нс мы получим начальную задержку в 10нс.

3. Анализ критических путей — а без репорта тула есть возможность оценить, какие пути какую имеют критическую длину (в штуках, без конкретных пикосекунд)?

Существует три вила путей, которые анализируется

- Clock path - данные пути идут от тактового генератора до регистров
- Data path - путь от одного регистра до другого
- Асинхронные пути - данные пути могут менять свое содержимое не зависимо от тактового сигнала

Из этого мы получаем два типа анализа

- Синхронный - анализ clock, data path - тактовая частота сравнивается с каналом передачи данных это гарантирует что два сигнала не поступят на регистр одновременно, что может привести к его метастабильности.
- Асинхронный сигнал - сравнивается тактовая частота с асинхронной для того, чтобы не возникло метастабильности.

Синхронный анализ

Проверяет для каждого регистра и интерфейса hold time, setup time.

Для того чтобы схема корректно среагировала на сигнал, информационный вход (или входы) схемы должен быть стабильным в течение некоторого времени предустановки (setup time) до прихода переднего фронта тактового сигнала и не должен изменяться в течение времени удержания (hold time) после прихода переднего фронта тактового сигнала. Сумма времен предустановки и удержания называется апертурным временем схемы. Это общее время, в течение которого информационный входной сигнал должен быть стабилен для его фиксации на выходе.

Данный анализ гарантирует

- Тактовый и синхронный сигналы (например, данные, включение тактовой частоты) не поступают на вход регистра в одно и то же время или примерно в одно и то же время
- Выходные данные регистра стабильны
- Передача данных проходит в стабильном режиме

Момент прибытия данных (Data Arrival Time) определяется как время фактического прибытия данных на регистр-получатель. Data Arrival Time = Launch Edge + max(t CLK) + tCO + max(t DT) здесь

- Фронт захвата (Latch Edge) — это фронт клока, который приходит на регистр-получатель и заставляет его захватить данные на входе, несет смысл точки отсчета, относительно которой развиваются события, а не какой-то величины, измеряемой в наносекундах.
- max(t CLK) — это максимальное время, за которое фронт запуска дойдет от тактового входа всей схемы до тактового входа источника. Анализатор как правило просто берет диапазон времени от «точно не менее чем» до «точно не более чем» и в данную формулу подставляет верхнюю границу «точно не более чем».
- tCO — это время срабатывания регистра (clock-to-output time), которое регистр тратит на то, чтобы, увидев фронт на тактовом входе поменять данные на своем выходе.
- T D - максимальное время прохождения события (данных) через комбинационную логику между регистрами, которая определяется пользователем. Вот эта величина как раз сильно зависит от пользователя.

Момент прибытия клока (Clock Arrival Time) определяется как время прохождения фронта захвата от тактового входа всей схемы к тактовому входу получателя. Причем под фронтом захвата понимается следующий фронт после фронта запуска. Фронт запуска пускает данные от источника к получателю, а через один период клока фронт захвата эти данные ловит на стороне получателя.

Момент ожидания данных (Data Required Time) определяется как время, за которое данные должны дойти до получателя до наступления времени предустановки на регистре-получателе.

Алгоритм синхронного анализа схемы

1. Setup time - мы можем определить слэк предустановки как наименьшую разницу между временем, которое разрешено потратить на путь до получателя, и временем, которое этот путь фактически займет. Положительный запас — хорошо, отрицательный — плохо. Слэк буквально переводится как провисание. Значит если слэк есть — значит межрегистровая передача настроена не «внатяг», условная «ниточка» свободно провисает.
2. Hold time слэк удержания - рассчитывается как min (Data Arrival Time) - max(Data Required Time)
   Важно отметить, что в случае рассмотрения Hold Slack фронты Launch Edge и Latch Edge — это теперь уже один и тот же фронт, а не два разных фронта, разделенных периодом клока. Регистру-получателю в данной ситуации нужно успеть удержать данные на входе в течение времени удержания от прихода фронта клока. Но данные меняет на его входе этот же фронт, пришедший где-то в другом месте на регистр-источник.

Асинхронный анализ

Данный анализ фоккусируется на параметрах восстановления и удаления

Recovery time (время восстановления) - Проверка времени восстановления гарантирует, что существует минимальный промежуток времени между тем, как асинхронный сигнал становится неактивным, и следующим активным тактового фронта. Другими словами, эта проверка гарантирует, что после того, как асинхронный сигнал становится неактивным сигнал становится неактивным, есть достаточно времени для восстановления, чтобы следующий активного фронта тактового импульса.

Removal time - Проверка времени удаления обеспечивает достаточное время между активным фронтом тактового сигнала и освобождением асинхронного управляющего сигнала. На сайте проверка гарантирует, что активный фронт тактового импульса не имеет никакого эффекта, поскольку асинхронный управляющий сигнал остается активным до момента удаления после активного фронта тактового импульса. фронта. Другими словами, сигнал асинхронного управления освобождается (становится неактивным) намного позже активного фронта тактового импульса, так что фронт тактового импульса не может иметь никакого эффекта влияние

Соответственно если у нас есть схема и мы знаем временных характеристики регистров и комбинационных элементов, то мы сможем вычислить критические пути.

4. В чем разница между блокирующим (=) и не блокирующим(<=) присваиваниями?

Blocking assigment

Оператором блокирующего присваивания является знак равенства ("="). Блокирующее присваивание получило свое название потому, что блокирующее присваивание должно оценивать аргументы RHS и завершать присваивание без прерывания любым другим оператором Verilog. Считается, что присваивание "блокирует" другие присваивания до тех пор, пока не завершится текущее присваивание.

Выполнение блокирующих заданий можно рассматривать как одноэтапный процесс:

1. Оценить RHS (уравнение правой части) и обновить LHS (выражение левой части) блокирующего присваивания без прерывания любым другим оператором Verilog.

Блокирующее задание "блокирует" выполнение последующих заданий в том же всегдашнем блоке до тех пор, пока не будет выполнено текущее задание

Non-blocking assigment

Оператор неблокирующего присваивания - это то же самое, что и оператор less-than orequal-to ("<="). Неблокирующее присваивание получило свое название потому, что присваивание оценивает выражение RHS неблокирующего оператора в начале временного шага и планирует обновление LHS в конце временного шага. Между оценкой выражения RHS и обновлением выражения LHS могут быть оценены и обновлены другие утверждения Verilog, а также могут быть оценены выражения RHS других не блокирующих назначений Verilog и запланировано обновление LHS. Неблокирующее присваивание не блокирует выполнение других операторов Verilog.

Выполнение неблокирующих заданий можно рассматривать как двухэтапный процесс:

1. Оценивать RHS неблокирующих операторов в начале временного шага.
2. Обновление LHS неблокирующих операторов в конце временного шага.

Неблокирующие присваивания выполняются только для регистровых типов данных и поэтому допустимы только внутри процедурных блоков, таких как начальные блоки и блоки always. Неблокирующие присваивания не допускаются в непрерывных присваиваниях.