Потоковый арбитр с политикой Quolity-of-Service (QoS)

1. Проектирование

1.1 Описание устройства (из условия тестового задания)

На вход модуля по STREAM_COUNT параллельным независимым потокам поступают транзакции. Из всех входных транзакций выбирается наиболее приоритетная, которая принимается и транслируется на выход модуля. Сигнал m_id_o принимает значение номера входного потока, с которого получена выходная транзакция. Приоритет входящий транзакции определяется сигналом s_qos_i. Приоритет транзакции является общим для всех пакетов транзакции (не меняется от пакета к пакету). Чем значение выше, тем более приоритетной является транзакция. Если на вход модуля подано несколько транзакций с совпадающим приоритетом, то выбор из них осуществляется по принципу Round-Robin. Нулевое значение сигнала s_qos_i не участвует в схеме сравнения приоритетов: транзакции с s_qos_i равным нулю рассматриваются по схеме Round-Robin наравне с наиболее приоритетными. Арбитраж производится потранзакционно: переключение арбитра на следующий входной порт производится только после завершения передачи транзакции с текущего выбранного арбитром порта.

1.2 Проектирование

Полученная модель должна включать в себя две основных части:

1. Выбор активного потока (выдача гранта)
2. Арбитраж входных потоков на основе вычисленного гранта.

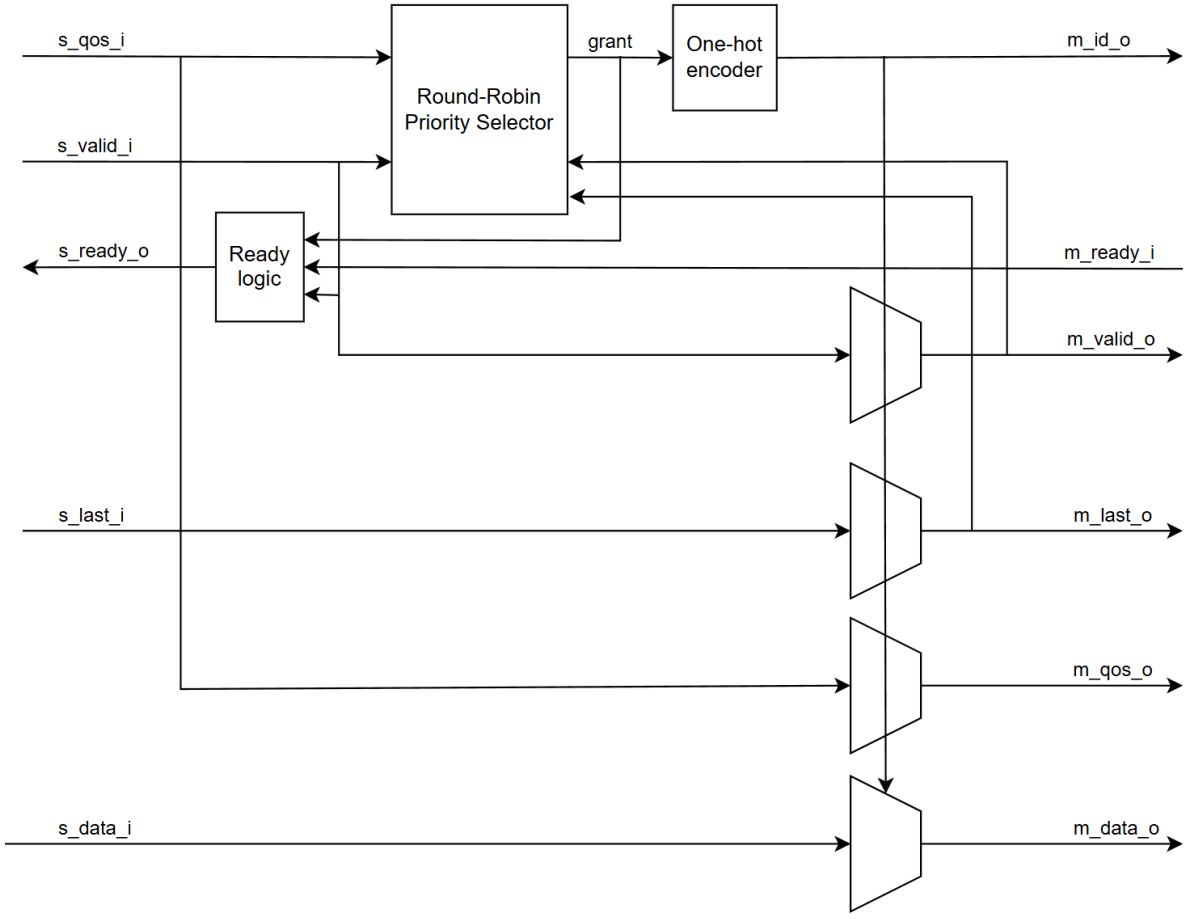


Рисунок 1. Модель потокового арбитра

Приоритет в проектировании был отдан минимизации площади и латентности при передаче данных. Также было принято ввиду, что длительность передачи пакета один такт и тактовые домены передающих мастеров и арбитра синхронны.

Выбор активного потока происходит на основе максимального значения приоритета на входах модуля s_qos_i. Все валидные транзакции в потоках, имеющие в качестве приоритета максимальное значение из ныне присутствующих или имеющие приоритет 0 далее арбитруются по принципу Round-Robin. Таким образом, в блок выбора можно функционально выделить два этапа:

- Этап составления вектора req, который имеет 1 на битах, соответствующих потокам с валидными данными и максимальными либо нулевыми s_qos_i
- Round-Robin арбитр

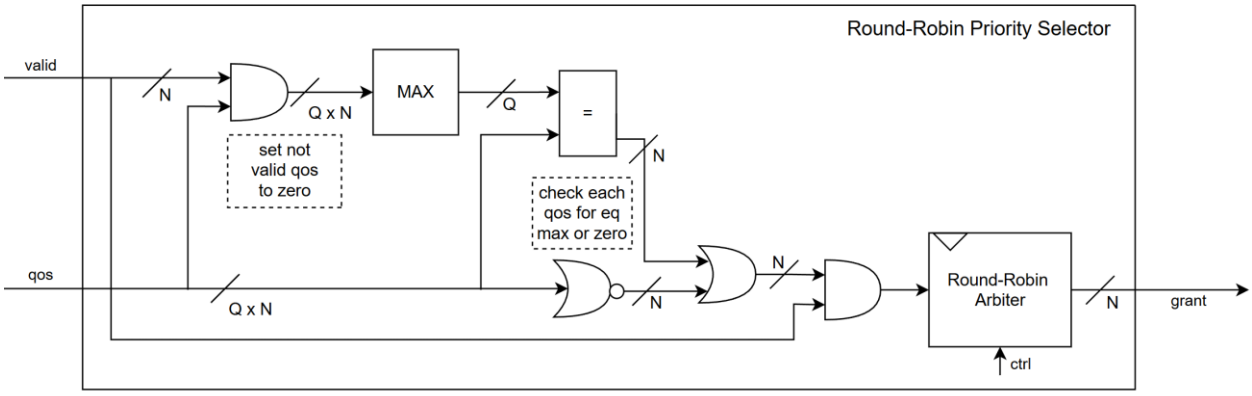


Рисунок 2. Блок выбора активного потока

Для реализации Round-Robin арбитра была выбрана архитектура с маской, использующая два simple-priority арбитра, основная задача которых передавать на выход позицию первой встреченной единицы в виде one-hot:

$$grant_i = \overline{req_0}\,\overline{req_1}\,\overline{req_2} \ldots \overline{req_{i-1}}\,req_i$$

Также был рассмотрен вариант rotate + simple-priority + rotate, однако предпочтение было другому варианту, так как модуль по условию задания параметризуемый, а архитектура с маской хорошо показывает себя при масштабировании как по временным характеристикам, так и по занимаемой площади.

На рисунке 3 можно заметить, что в данном блоке присутствуют 2 регистра, используемые для хранения прошлого значения указателя Round-Robin и нынешнего значения гранта. Выбор активного потока должен оставаться неизменным всю транзакцию, однако входы потоков могут изменяться таким образом, что комбинационная логика выбора может сработать и поменять значение гранта до конца текущей транзакции. Таким образом хранение посчитанного в начале транзакции значения гранта помогает сохранить неизменным выбор потока в течение всей транзакции. Указатель хранится в one-hot кодировке для удобства получения маски.

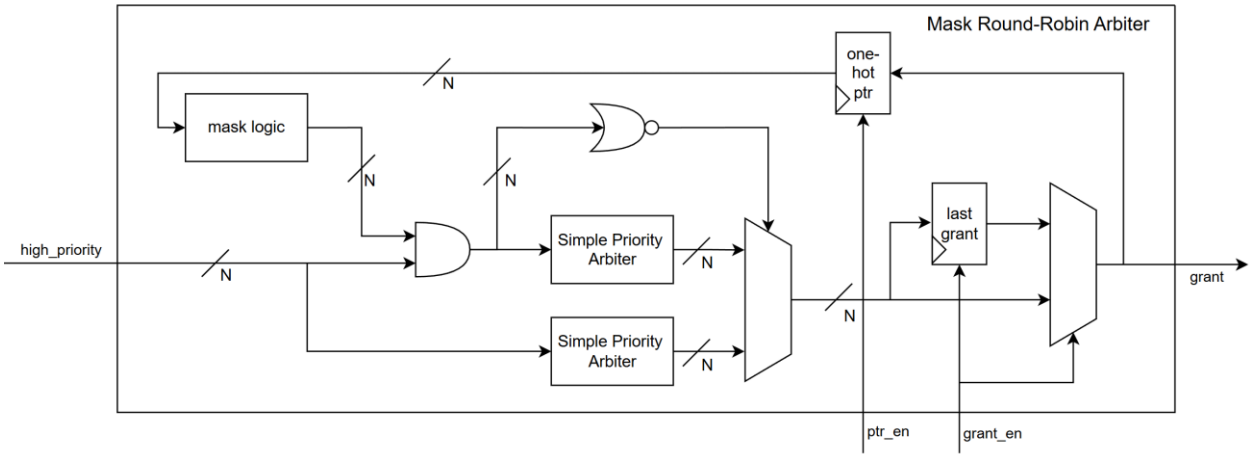


Рисунок 3. Round-Robin арбитр

Появляется необходимость управлять обновлением регистров. Логика обновления следующая:

Регистр указателя обновляется:

- Если транзакция завершилась сигналом last
- Если транзакция не завершилась (не было сигнала last), но потока valid по какой-либо причине оказался в 0

Регистр гранта обновляется:

- Если обновился указатель
- Если пришла валидная транзакция

Таким образом отметим, что обрабатывается ситуация отказа какого-либо из master потоков, сохраняя честность (пойнтер обновляется).

Выходы s_ready_o должны соответствовать гранту с поправкой на валидность потоков и входной сигнал m_ready_i следующим образом. Если m_ready_i = 0 готовности приема нет, следовательно, все s_ready_o = 0. Иначе, если нет валидных потоков, выставляется готовность принять на все потоки, иначе грант. Можем представить это в виде логических элементов следующим образом:

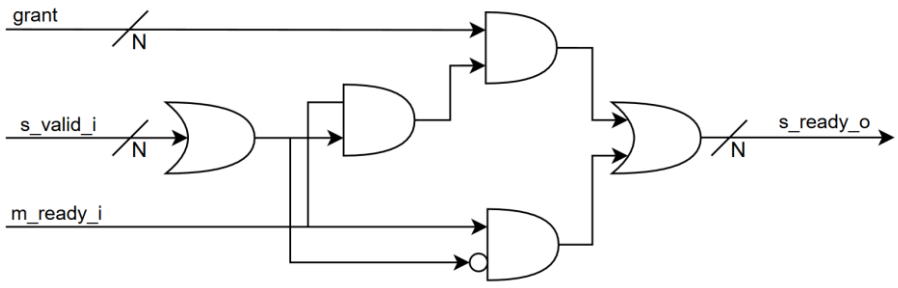


Рисунок 4. Логика s_ready_o

2. Тестирование

Выделим паттерны входов при которых дизайн должен работать соответствующим образом:

Паттерн	Поведение
2 и более валидных транзакции с одинаковыми приоритетами	Цикличный выбор следующего входа с максимальным приоритетом после завершения транзакции (либо ее сбоя)
2 и более валидных транзакции с одинаковыми приоритетами либо 0-м приоритетом	Цикличный выбор следующего входа с максимальным приоритетом либо нулевым приоритетом после завершения транзакции (либо ее сбоя)
Различные ненулевые приоритеты валидных транзакций	Упорядоченный выбор в порядке убывания приоритета после завершения транзакции (либо ее сбоя)
В течение передачи текущей транзакции сигнал last был выставлен в 1	Сигнал конца транзакции, на следующем такте происходит выбор следующей транзакции.
Отсутствие валидных входов	Все выходные ready устанавливаются в 1, выходной valid = 0.
Присутствует невалидный вход высшего приоритета	Данный вход не участвует в выборе
Валидная транзакция с высшим приоритетом появляется до завершения передачи текущей транзакции	Игнорируется до завершения передачи текущей транзакции. Учитывается при следующем выборе.
В течение передачи текущей транзакции сигнал valid был выставлен в 0 до поступления сигнала last	Сбой в потоке. Выходной valid устанавливается в 0, на следующем такте происходит выбор следующей транзакции.
Входной сигнал ready установлен в 0	Все выходные сигналы ready установлены в 0

В текстовых файлах в папке sim как раз лежат вектора для проверки всех вышеперечисленных случаев.

3. Синтез и рекомендации по модификации дизайна

3.1. Запуск синтеза

1. Создаем RTL-проект в Vivado Xilinx на базе Virtex-7 VC707
2. В Add design sources добавляем файлы из папочки src
3. В Add constraint sources добавляем XDC файл из папочки synth
4. Выбираем, если автоматически не выбран wrapper как top модуль
5. Нажимаем кнопочку Run Synthesis

3.2. Отчеты о синтезе

Синтез производился для 2-х, 4-х и 8-ми потоков с параметрами T_DATA_WIDTH = 8, T_QOS__WIDTH = 4.

STREAM_COUNT	WNS, ns	LUT	FF
2	8.062	29	8
4	6.621	75	12
8	5.120	188	20

3.3. Рекомендации по модификации дизайна

В данном устройстве нулевая латентность упирается в необходимость вычислять грант в такт поступления новой транзакции. Однако для этого в первый такт передачи транзакции используется комбинационно вычисленный id потока, что обуславливает довольно длинную цепочку логики от входа сигнала до выхода. Более того, это приводит к стремительному ухудшению временных характеристик схемы при увеличении числа потоков.

T_DATA_WIDTH	T_QOS__WIDTH	STREAM_COUNT	WNS 100МГц	Fmax, МГц
8	4	2	8.062	516
8	4	4	6.621	296
8	4	8	5.120	205

Чтобы увеличить максимальную рабочую частоту схемы, а тем увеличив пропускную способность можно прибегнуть к простой модификации, как показано на рисунке 5. Таким образом теперь для выбора активного потока используется только насчитанный в начале транзакции грант. Однако ценой за это является +1 такт задержки в начале каждой новой транзакции, поэтому такая модификация может дать прирост к пропускной способности только если среднее количество пакетов в транзакции соизмеримо с приростом к максимальной рабочей частоте. Отметим также, что в такт выбора потока необходимо держать выходной сигнал m_valid_o в 0 для избегания передачи мусорных данных.

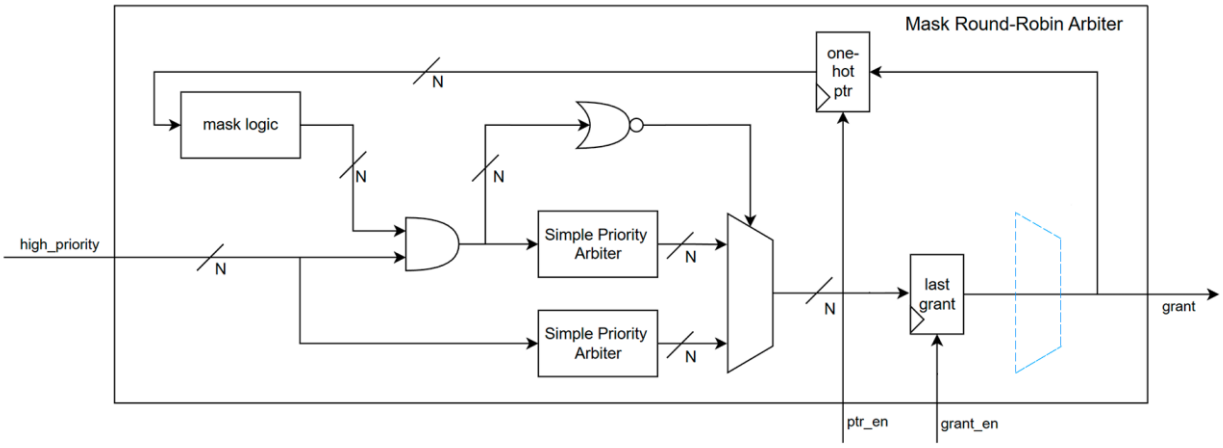
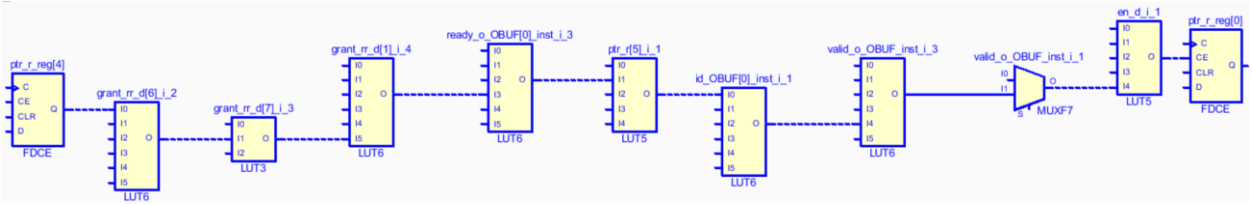


Рисунок 5. Предложения по модификации модели

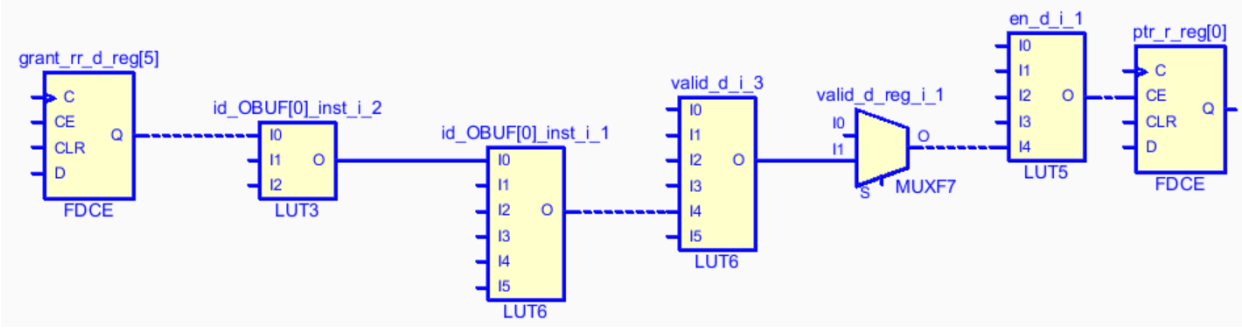
Как оказалось, такая модификация позволяет увеличить устойчивость модели к масштабированию числа потоков:

T_DATA_WIDTH	T_QOS__WIDTH	STREAM_COUNT	WNS 100МГц	Fmax, МГц
8	4	2	8.294	586
8	4	4	7.841	463
8	4	8	7.304	371

Сравним также критические пути в полученных синтезированных дизайнах для двух вариантов. В первом дизайне критический путь лежит от клокового входа регистра пойнтера до его входа enable, и проходит он через логику выбора grant, вычисление id, мультиплексирование valid и вычисление самого enable. Во втором случае цепочка сокращается за счет того, что grant уже вычислен и зарегистрован, поэтому критический путь лежит от клокового входа регистра гранта до входа CE регистра пойнтера.



(а)



(б)

Рисунок 6. Критический путь в: а) оригинальной модели; б) модели с модификацией

Также стоит учитывать input и output задержки модуля, вероятно может понадобиться буферизовать данные на входе и/или выходе модуля.