УДК 004.021 DOI: 10.31114/2078-7707-2021-4-99-105

Анализ пропускной способности многобанковой памяти в системе на кристалле

А. В. Воронов, Р. В. Воронов, Р. Ф. Ильясов

АНО ВО «Университет Иннополис», г. Иннополис,

a.voronov@innopolis.university, r.voronov@innopolis.university, r.iliasov@innopolis.ru

***Аннотация* — Современные системы на кристалле используют многопортовую память. Одной из ее реализаций является многобанковая память, которая увеличивает пропускную способности при малых затратах логических элементов по сравнению с другими вариантами. Целью исследования является анализ задержки транзакций, при различных количествах портов и банков. В ходе работы были рассмотрены основные элементы, входящие в состав многобанковой памяти, варианты ее ускорения, а также другие способы реализации многопортовой памяти. После этого было проведено измерение задержек по предложенным тестам. В заключении было выявлено рекомендуемое количество банков по отношению к количеству портов.**

***Ключевые слова* — многопортовая память, многобанковая память, арбитраж памяти, ready/valid интерфейс.**

I. Введение

В наши дни память на кристалле является одним из главных компонентов дизайна микроэлектронных плат. Как правило программируемые логические интегральные схемы (ПЛИС) комплектуются тремя блочными конфигурациями оперативной памяти (**BRAM**) [1]:

1) **Single Port**. Один порт на чтение/запись. Каждый такт можно запускать либо запрос на чтение, либо запрос на запись.

2) **Pseudo Dual Port**. Один порт на чтение и один порт на запись. Запрос на чтение и запрос на запись в каждом такте можно делать независимо.

3) **Dual Port**. Два порта на чтение/запись. Позволяет в каждом такте запускать два независимых запроса разных типов на разные адреса - два чтения, или две записи, или одно чтение и одну запись.

При проектировании сложных систем, таких как сетевые микросхемы или графические процессоры, часто возникает потребность в многопортовой памяти, позволяющей делать несколько операций чтения и записи одновременно [2].

В данной статье рассматривается разработка дизайна многопортовой памяти по принципу арбитража запросов, которая развивается в многобанковую память для устранения ряда недостатков. В завершение проведена оценка пропускной способности в зависимости от количества портов и количества банков.

Основные свойства реализованной памяти:

- Взаимодействие модуля сделано на основе ready/valid интерфейса;
- Память масштабируема относительно портов чтения, портов записи, и банков памяти.
- При одновременном обращении по нескольким портам порядок обработки запросов произвольный;
- После подтверждения транзакции чтения, ответ приходит через несколько тактов. Количество тактов фиксировано и равно задержке чтения блока памяти. Данные сопровождаются дополнительным сигналом готовности;
- При операции записи данные будут записаны в память в течении нескольких тактов. Количество тактов фиксировано и равно задержке записи блока памяти.

II. Pseudo dual port

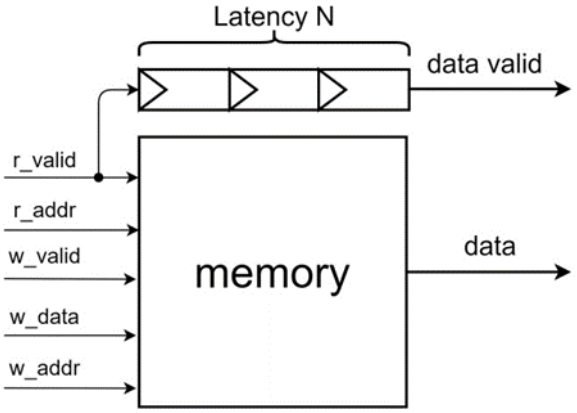

Рис. 1. Устройство Pseudo dual port

В качестве основы дизайна была выбрана память **Pseudo Dual Port** BRAM рис. 1. Использование блока памяти такого типа позволяет разнести отдельно порты чтения и записи из-за их независимой работы [3]. Следует учесть, что при единовременном чтении и записи по одному адресу, данные вернутся не обновленными, то есть чтение происходит раньше записи.

III. Многопортовая память

В процессе создания многопортовой памяти необходимо решить проблему конфликтов, возникающую при запросе на чтение или на запись по двум портам одновременно. Это проблема порождается ограничением стандартного блока памяти.

Существует два общих принципа решения данной проблемы, которые были рассмотрены в статьях [4], [5], [6]:

A. Дублирование памяти

Один из вариантов, это дублирование блоков памяти для одновременного предоставления доступа многим портам. Существует несколько реализаций данного способа, например, таблица текущих значений (**Live Value Table**) рис. 2 или ОЗУ на основе **XOR**, исследованная в статье [2]. Для примера рассмотрим таблицу текущих значений.

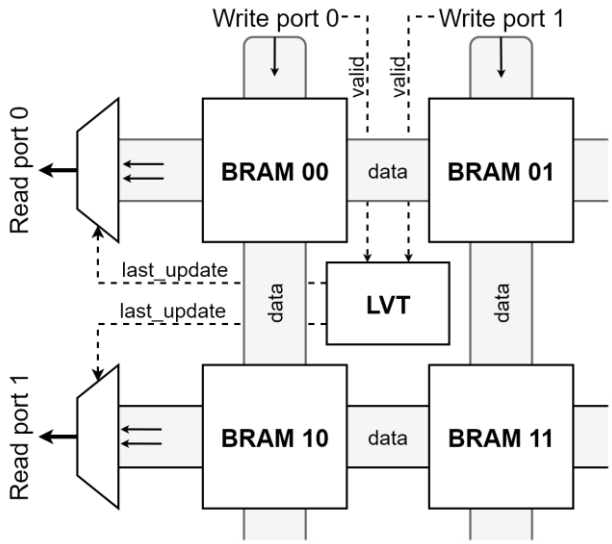

Рис. 2. LVT с 2 портами чтения и 2 портами записи

В данном случае используются блоки памяти в количестве (порты чтения * порты записи) штук. Память по вертикали дублирует друг друга, а каждая комбинация блоков по горизонтали составляет единую память, где при чтении выбирается нужная секция с помощью таблицы значений. Так как каждому блоку соответствует только один порт чтения и один порт записи, то все операции могут происходить в один такт времени.

Плюсом данного метода является параллельная обработка портов, то есть время одного запроса не зависит от количества других. Минус данного подхода состоит в лишнем использование памяти, суммарный объем памяти при этом равен объему одного блока. Другим минусом согласно [4] является существенное увеличение размера чипа.

B. Задержка запроса

Другой из вариантов, это обработка одного запроса, и приостановка других. Для данного способа также существует несколько реализаций, одной из которых является установка **FIFO** перед блоком памяти. В этом случае запросы складываются в очередь, и выталкиваются, когда память свободна. Если очередь полная, то новые транзакции не инициируются. Другой вариант реализации рассмотрен в статьях [4], [7] – арбитраж запросов рис. 3, где основным принципом является предоставление доступа только одному порту единовременно, в то время, как остальные ожидают своей очереди

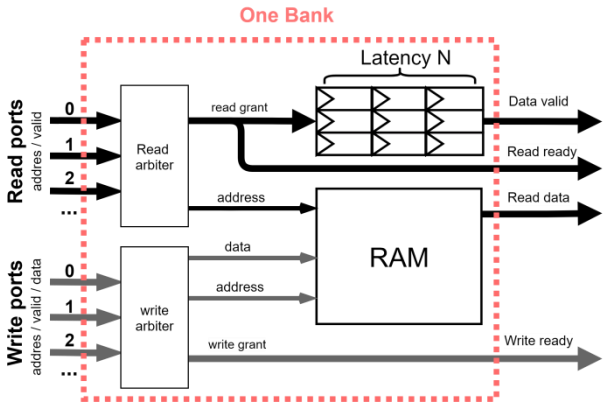

Рис. 3. Устройство многопортовой памяти по принципу арбитража запросов.

В устройстве памяти добавляется новый элемент - арбитр, который занимается распределением доступа к памяти, при этом обрабатывая ранние запросы в первую очередь. Таким образом максимальное время ожидания порта ограничено количеством запросчиков.

```
parameter N = 4; // Number of requesters

 input  wire [N-1:0] req;
 output wire [N-1:0] grant;

 reg  [N-1:0] pointer_req;
 wire [2*N-1:0] double_req   = {req,req};
 wire [2*N-1:0] double_grant =
double_req & ~(double_req - pointer_req);
 assign grant [N-1:0] =
double_grant[N-1:0] | double_grant[2*N-1:N];
```

Рис. 4. Вариант реализации кругового асинхронного арбитра [8], [9]

В устройстве арбитра на рис. 4 основную роль выполняет ***указатель на запрос*** (***pointer request***), так как он определяет очередность выдачи доступа. Логика в арбитре устроена так, что выбирается запрос, на который ссылается указатель, или, если такого запроса нет, выбирается ближайший следующий к нему.

К примеру, при активном 1 и 3 порту из пяти, картина запросов будет следующая ***req*** = 01010_2:

если ***pointer_req*** = 01000_2, то ***grant*** = 01000_2
если ***pointer_req*** = 10000_2, то ***grant*** = 00010_2

Значение указателя по фронту клока смещается на позицию после выданного доступа для сохранения кругового (**round-robin**) эффекта.

Кроме этого, нужно чтобы порты работали по valid/ready протоколу, для обеспечения возможности приостанавливать их.

Плюсом метода арбитража запросов в отличие от таблицы текущих значений является использование только одного блока памяти, но из-за этого время осуществления транзакций зависит от количества запросчиков.

IV. Порты чтения и записи

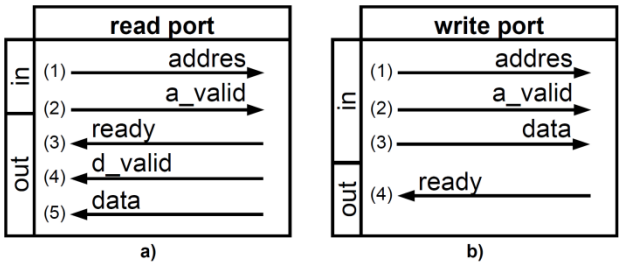

Рис. 5. Устройство порта чтения и записи

Разберём устройство портов с ready/valid интерфейсом, согласно [10], на примере порта чтения рис. 5a.:

1) **Address** – шина, по которой передается адрес ячейки памяти, к которой производится доступ.
2) **Address valid** – провод, по которому выставляется логическая единица, когда производится запрос и не снимается до тех пор, пока не было получено подтверждение по порту ready.
3) **Ready** – провод, который сигнализирует о принятии запроса и начале транзакции.
4) **Data valid** – провод, где выставляется логическая единица, когда проходят актуальные данные на шину данных. В целях упрощения, предполагается, что запрашивающий модуль всегда готов принять данные. В другом случае, сигнал true должен удерживаться до получения подтверждении о принятие данных от порта. Однако для этого на выходе необходимо поставить блок fifo, а всю конструкцию окружить кредитным счётчиком. Впрочем, это выходит за рамки статьи, и здесь рассматриваться не будет.
5) **Data** – шина, по которой передаются данные, полученные в ходе транзакции.

Порт записи (рис. 5b.) устроен похожим образом, но немного проще, т.к. отсутствует необходимость в сигнале valid на выходе.

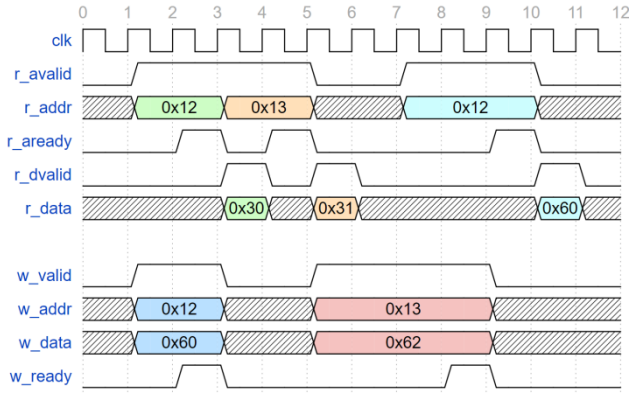

Рис. 6. Пример временной диаграммы чтения и записи

V. Многобанковая память

Одним из минусов многопортовой памяти с арбитражем портов является задержка запросов. Однако использование множества банков такой памяти позволит увеличить пропускную способность без увеличения количества ячеек памяти.

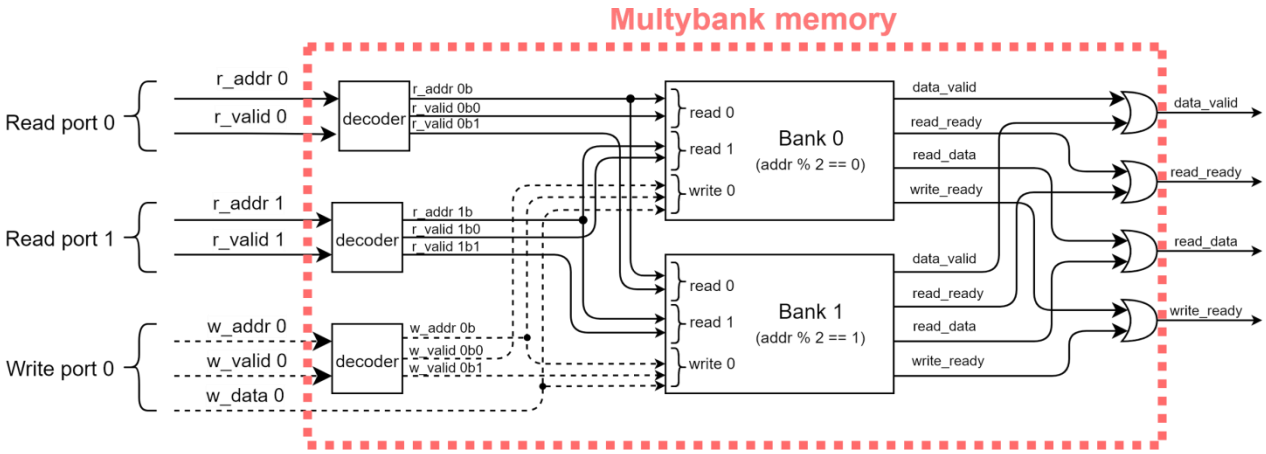

Рис. 7. Устройство многобанковой памяти

Многобанковая память создается на основе многопортовой памяти и является ее усложнением. Идея заключается в разделении памяти на N разделов (банков), где каждый банк является частью общей памяти. Причём блоки памяти могут быть произвольного размера, и организованы так, чтобы устройства разного типа, у которых общие участки памяти малы, обращались преимущественно в

собственные банки. [11] Тем самым можно добиться лучшей пропускной способности.

Случаи обращения нескольких портов к одному банку приводят к проблеме конфликтов, которая решается задержкой запроса.

Вероятность конфликтов можно уменьшить путем увеличения количества банков или грамотным разбиением памяти на области. Ещё одним из способов оптимизации конфликтов является планирование выполнения инструкций на программном уровне или на уровне компилятора таким образом, чтобы максимально снизить количество одновременных записей в один и тот же банк.

Память на рисунке 7 имеет 2 порта чтения, 1 порт записи и 2 банка. Для каждого порта создается модуль декодера, который на основе адреса перенаправляет сигнал valid нужному банку. Помимо этого, результаты банков складываются. Таким образом, интерфейс многобанковой памяти внешне не отличим от многопортовой.

Плюсами использования многобанковой памяти является увеличение пропускного потока, при полном использовании памяти.

VI. Оценка пропускной способности

Оценка пропускной способности была осуществлена путем измерения среднего времени обработки транзакций. Так как порты чтения и записи работают независимо, рассматривались случаи равного количества этих портов. Помимо этого, общая память разделена на банки одинакового объёма, и задержка чтения из памяти составляет 1 такт.

Измерения проходили путем трех тестов:

- Тест 1 (разделенный), при котором каждый порт обращается только к одному банку. Однако каждому банку может соответствовать несколько портов. Порты по банкам распределяются равномерно.
- Тест 2 (частично разделенный), при котором каждый порт обращается с вероятностью 80% к своему банку и 20% к другим. Порты по банкам распределяются равномерно. Этот случай моделирует условия наиболее приближенное к реальному распределение запросов.
- Тест 3 (случайный), при котором каждый порт обращается по случайному адресу.

Средняя задержка (average delay) измеряется в тактовых сигналах.

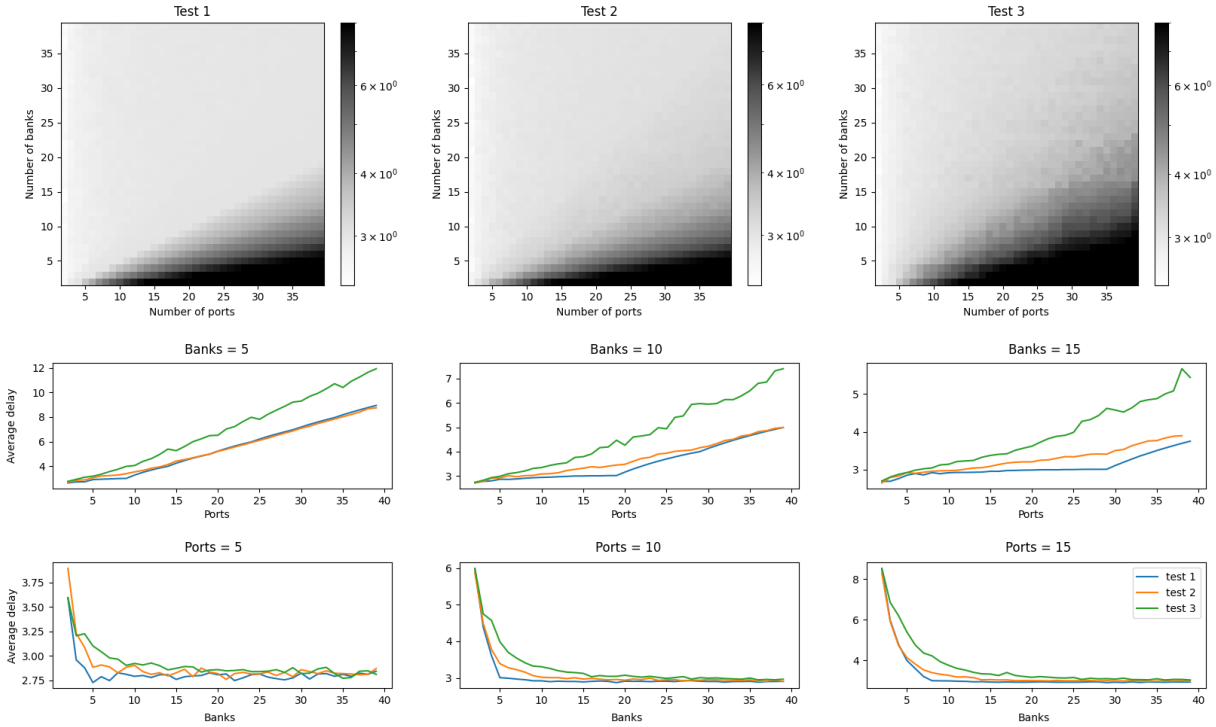

Рис. 8. Измерение средней задержки относительно количества портов и банков

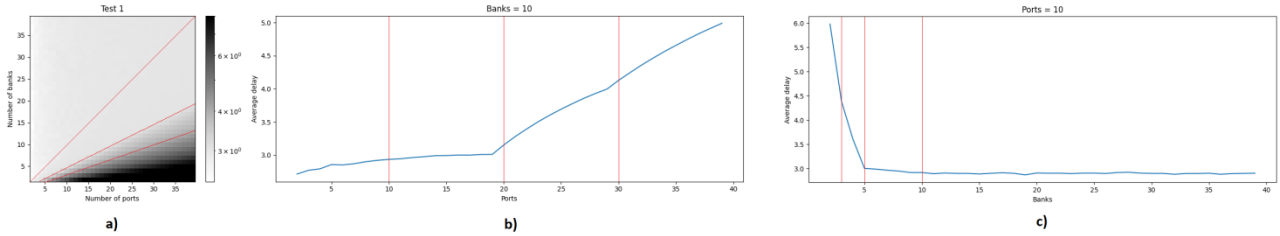

Рис. 9. Зависимость задержки 1 теста: a) от портов и банков, b) при фиксированном количестве банков, c) при фиксированном количестве портов

По данным графиков рис. 8 видно, что чем случайнее распределение запросов по банкам, тем больше средняя задержка транзакции, что объясняется увеличением количества конфликтов. Помимо этого, согласно графикам, зависимость задержки прямо пропорциональна количеству портов и обратно пропорциональна количеству банков.

На рисунке 9 представлена ситуация, когда порты обращаются только к своему банку. Красными линиями отмечены отношения количество портов к количеству банков кратные 1, 2, 3, что соответствуют характерным изменениям на графиках. Прослеживается следующая зависимость: пока количество портов меньше чем удвоенное количество банков, то увеличение задержки мало, при 2x кратных и 3x кратных превышениях происходит резкое увеличение задержки. Исходя из этого, можно сделать вывод, что оптимальное количество банков не должно быть меньше половины количества портов.

VII. Дальнейшее улучшение памяти

Дополнительно были рассмотрены варианты улучшения многобанковой памяти, одним из которых является предоставление банкам возможности угадывать следующий запрос при их отсутствии, что позволяет сократить время транзакции на 1 такт. Это обеспечивается путем выставления **Ready** сигнала равным одному, для предполагаемого порта.

К примеру, изначально нет никаких запросов по портам, тогда банк выставляет **Ready** для предполагаемого порта, например, для порта под номером 2. Если порт был угадан, то в тот момент, когда по нему выставляется запрос, сразу начинается транзакция, рис. 11. В противном случае, если запрос приходит по другим портам, то **Ready** выставляется с задержкой в один такт, из-за чего время транзакции остается неизменным, рис. 10.

Для выбора предполагаемого порта одной из лучших стратегий является распределение вероятности на основе частоты запросов. В простом случае бывает достаточным выбирать случайный порт.

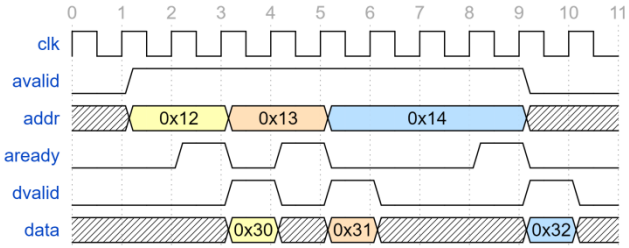

Рис. 10. Пример неэффективного ready/valid интерфейса (по умолчанию Ready = 0)

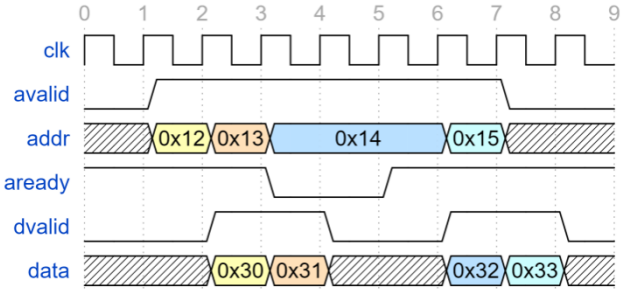

Рис. 11. Пример эффективного ready/valid интерфейса (по умолчанию Ready = 1)

VIII. Заключение

Таким образом, многобанковая организация памяти позволяет повысить пропускную способность системы за счет снижения количества конфликтов, во время которых к одному блоку памяти обращается несколько портов. Этот приём используется в графических процессорах, роутерах и других микросхемах. Альтернативой многобанковой памяти является дублирование памяти, но этот прием используется редко, из-за существенного увеличения размера чипа. В ходе исследования были установлены зависимости задержки транзакций от количества портов и количества банков памяти, а также проанализированы результаты и предложены улучшения.

Благодарности

Выражаем благодарность Юрию Панчулу за помощь в постановке изучения микроархитектуры цифровых схем в Иннополисе с 2018 год.

Литература

[1] Application specific multi-port memory customization in FPGAs / A. M. Gorker, E. Y. Hasan, Y. Arda, N. Smail// 24th International Conference on Field Programmable Logic and Applications (FPL). — 2014. — С. 1-4.

[2] Composing Multi-Ported Memories on FPGAs / C. E. Laforest, Z. Li, T. O'rourke [и др.]. // ACM Transactions on Reconfigurable Technology and Systems. — 2014. — № 7. — С. 1-23.
[3] Мамаева Т. IDT: многопортовая память: как она работает? // Компоненты и Технологии. 2001. №14.
[4] Verbeure, T. Building Multiport Memories with Block RAMs / T. Verbeure. // URL: https://tomverbeure.github.io/2019/08/03/Multiport-Memories.html (дата обращения: 31.08.2021).
[5] Wang, Z. An Intelligent Multi-Port Memory / Z. Wang, Q. Zuo, J. Li. // International Symposium on Intelligent Information Technology Application Workshops. — 2008. — С. 251-254.
[6] A multiported register file with register renaming for configurable softcore VLIW processors / Anjam F., Wong S., Nadeem F. // International Conference on Field-Programmable Technology. — 2010. — С. 403-408.
[7] Орлов С. П., Биктимиркин Е. Ю., Тютнев А. А. Имитационная модель многопортовой памяти микропроцессоров. // Перспективные информационные технологии (ПИТ 2013): Тр. Междунар. науч.-техн. конф.–Самара: Изд. Самарск. науч. центра РАН 2013 (pp. 357-360).
[9] Weber, M. Arbiters: Design Ideas and Coding Styles / M. Weber: Silicon Logic Engineering, Inc.
[8] Advanced Synthesis Cookbook / Altera Corporation — 2011
[10] Charles, E. L. Rules for Ready/Valid Handshakes / E. L. Charles.// FPGA Design Elements — URL: http://fpgacpu.ca/fpga/handshake.html (дата обращения: 05.09.2021).
[11] Multibank Memory Optimization for Parallel Data Access in Multiple Data Arrays / Yin, a. X. Shouyi, a. M. Zhicong [и др.]. // Association for Computing Machinery. — 2016. — С. 1–8.
[12] Салищев С. И., Шеин Р. Е. Новые алгоритмы для конвейерного вычисления БПФ по смешанному основанию без копирования на многобанковой памяти с произвольным доступом // КИО. 2013. №2.
[13] Фролов М. С. Исследование временных параметров многопортовой памяти, реализованной на ПЛИС / М. С. Фролов; науч. рук. А. И. Солдатов // Неразрушающий контроль: сборник трудов VI Всероссийской научно-практической конференции "Неразрушающий контроль: электронное приборостроение, технологии, безопасность", Томск, 23-27 мая 2016 г.: Изд-во ТПУ, 2016. — [4 с.].
[14] Dean, R. A. Multi-Port Memories / R. A. Dean. // High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test. — 2003. — № 22A. — С. 47-56.
[15] Лементуев В. А. Многопортовая память микропроцессорных систем / В. А. Лементуев // Информационные технологии и вычислительные системы, № 2, 2009. – С. 96 – 101.

Multibank Memory Bandwidth Analysis in on-chip System

A. V. Voronov, R. V. Voronov, R. F. Iliasov

"University Innopolis", Innopolis, a.voronov@innopolis.university, r.voronov@innopolis.university, r.iliasov@innopolis.ru

***Abstract* — nowadays, memory on a chip is one of the main components of microelectronic design. Usually, field-programmable gate arrays (FPGAs) are equipped with three block configurations of RAM: Single Port, Pseudo Dual Port, and Dual Port. This article discusses the development of a multiport memory design based on the principle of request arbitration, which is then redesigned into multibank memory. The purpose of the study is to analyze the transaction delay, with different numbers of ports and banks. The Pseudo Dual Port BRAM was chosen as the basis of the design. Using this typeof memory block allows separating the read and write ports because of their independent operation. In the process of creating multiport memory, it is necessary to solve the problem of conflicts that occur when a read or write requests are made on two ports simultaneously. This problem is caused by the limitation of the standard memory block. There are two general principles for solving this problem: memory duplication and request delay. The advantages and disadvantages of each option were considered. Finally, the bandwidth of the multibank memory wasevaluated as a function of the number of ports and the number of banks. The bandwidth was estimated by measuring the average transaction processing time on 3 different tests. It was concluded that the optimal number of banks should not be less than half the number of ports.**

***Keywords* — multiport memory, multibank memory, memory arbitration, ready/valid interface.**

REFERENCES

[1] Application specific multi-port memory customization in FPGAs / A. M. Gorker, E. Y. Hasan, Y. Arda, N. Smail// 24th International Conference on Field Programmable Logic and Applications (FPL). — 2014. — pp. 1-4.
[2] Composing Multi-Ported Memories on FPGAs / Charles, E. L., Z. Li, T. O'rourke [et al.]. // ACM Transactions on Reconfigurable Technology and Systems. — 2014. — № 7. — pp. 1-23.
[3] Mamaeva T. IDT: multiport memory: how does it work? // Components and Technology. 2001. № 14. (In Russian)
[4] Verbeure, T. Building Multiport Memories with Block RAMs / T. Verbeure. // URL: https://tomverbeure.github.io/2019/08/03/Multiport-Memories.html (date of access: 31.08.2021).
[5] Wang, Z. An Intelligent Multi-Port Memory / Z. Wang, Q. Zuo, J. Li. // International Symposium on Intelligent Information Technology Application Workshops. — 2008. — pp. 251-254.
[6] A multiported register file with register renaming for configurable softcore VLIW processors / Anjam F., Wong S., Nadeem F. // International Conference on Field-Programmable Technology. — 2010. — pp. 403-408.
[7] Orlov S. P., Biktimirkin E. Yu., Tyutnev A. A. A Simulation model of multiport microprocessor memory. // Advanced Information Technology (PIT 2013): International Scientific and Technical Conf.: Samarsk. Scientific Center of the RAS 2013 (pp. 357-360). (In Russian)

[8] Advanced Synthesis Cookbook / Altera Corporation — 2011
[9] Weber, M. Arbiters: Design Ideas and Coding Styles / M. Weber: Silicon Logic Engineering, Inc.
[10] Charles, E. L. Rules for Ready/Valid Handshakes / E. L. Charles.// FPGA Design Elements — URL: http://fpgacpu.ca/fpga/handshake.html (date of access: 05.09.2021).
[11] Multibank Memory Optimization for Parallel Data Access in Multiple Data Arrays / Yin, a. X. Shouyi, a. M. Zhicong [et al.]. // Association for Computing Machinery. — 2016. — pp. 1–8.
[12] Salishhev S. I., Shein R. E. New algorithms for pipeline computation of mixed basis FFT without copying on multibank random access memory // KIO. 2013. №2. (In Russian)
[13] Frolov M. S. Study of time parameters of multiport memory implemented on FPGA / Frolov M. S.; supervisor. A. I. Soldatov // Nondestructive testing: VI All-Russian Scientific and Practical Conference "Nondestructive testing: electronic instrumentation, technology, safety", Tomsk, 23-27 may 2016 y.: TPU Press, 2016. — [4 p.]. (In Russian)
[14] Dean, R. A. Multi-Port Memories / R. A. Dean. // High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test. — 2003. — № 22A. — pp. 47-56.
[15] Lementuev V. A. Multiport memory of microprocessor systems / Lementuev V. A. // Information Technology and Computer Systems, № 2, 2009. – (pp. 96-101). (in Russian)