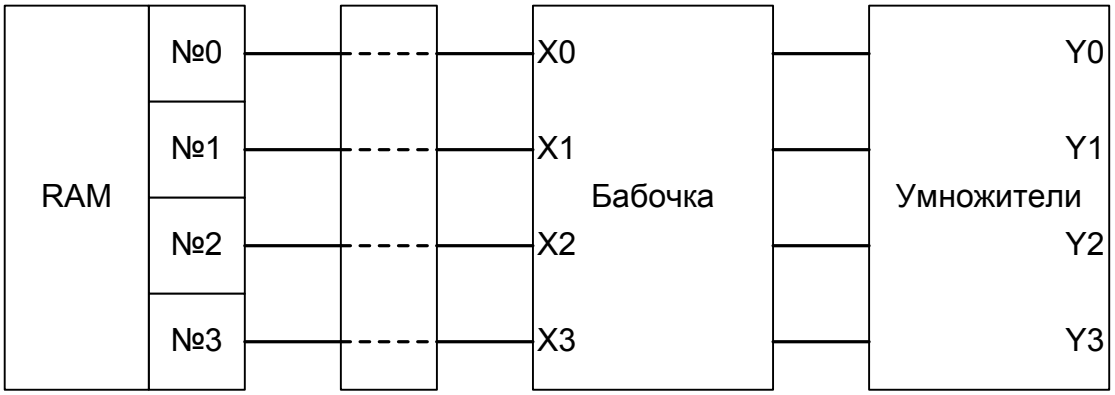
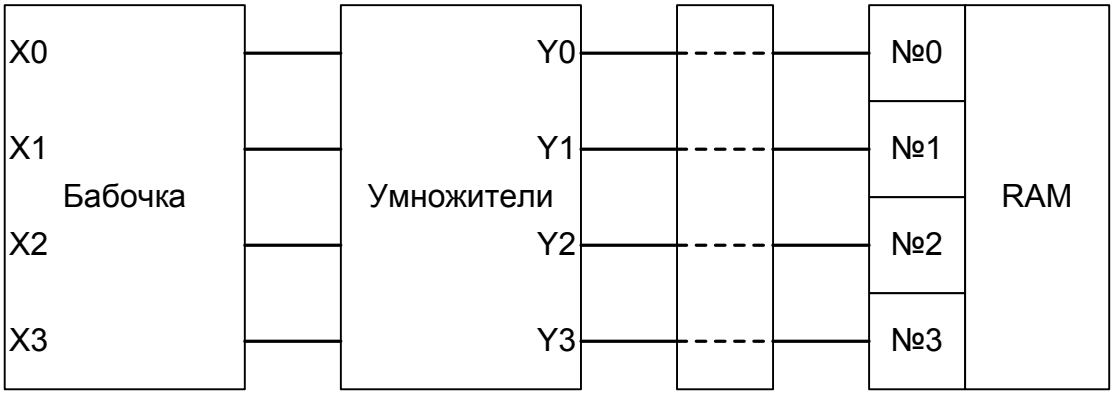
То устройство, которое нам нужно поставить с обеих сторон памяти строго говоря не называется мультиплексором. Назовем его пока миксером – это ближе по сути и ничего лучшего сходу придумать не удалось. Нам нужно сделать так, чтобы X0 подавался на вход бабочки не всегда из банка 0, а из любого банка, а после всех вычислений результат Y0 попадал не всегда в банк 0, а в любой банк памяти. Причем особенностью в работе данного устройства является то, что 2 разных точки не могут претендовать на то, чтобы быть записанными/прочитанными в/из одного и того же банка одновременно. Т.е. если мы, к примеру, решили записать Y0 в банк №1, то Y1 мы должны записать в какой-нибудь другой банк – например в банк №3, тогда Y2 и Y3должны поделить между собой оставшиеся банки №0 и №2. Аналогично для чтения. Если точка из банка №0 подается на вход бабочки X3, то точка из банка №3 уже не может быть подана на X3 – ее надо подать, к примеру на X2, тогда точки из банков №1 и №2 поделять между собой X0 и X1.

Теперь давайте посмотрим по какому алгоритму выбирать из какого банка на какой вход бабочки подавать данные и с какого умножителя в какой банк записывать и, что тоже не маловажно какие адреса подавать на банки памяти.

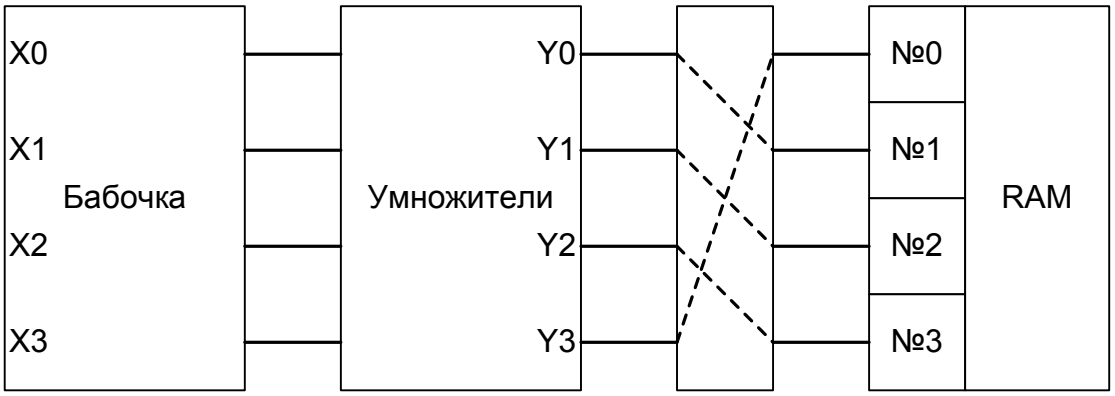
На первом этапе, очевидно, что при чтении на все банки памяти нужно подавать один и тот же адрес и банки подсоеденены ко входам бабочки "напрямую"



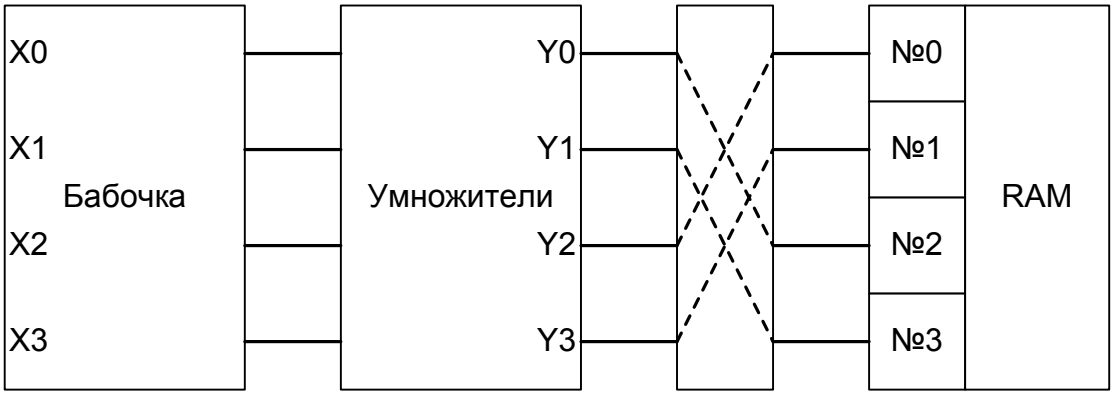
Однако, при записи результатов так сделать нельзя – все 4 точки одной и той же бабочки окажутся в одном и том же банке памяти и прочитать их одновременно не получится. Но т.к. на втором этапе X0 "отстоит" от X1 на 128 адресов (равно как и X1 от X2 и X2 от X3) мы можем через каждые 128 бабочек циклически "вращать" банки памяти относительно входов бабочки с помощью миксера. Т.е. первые 128 результатов (под результатом имеется ввиду набор Y0 – Y3) записывается так



Следующие 128, повернув циклически банки памяти на 1 "шаг"



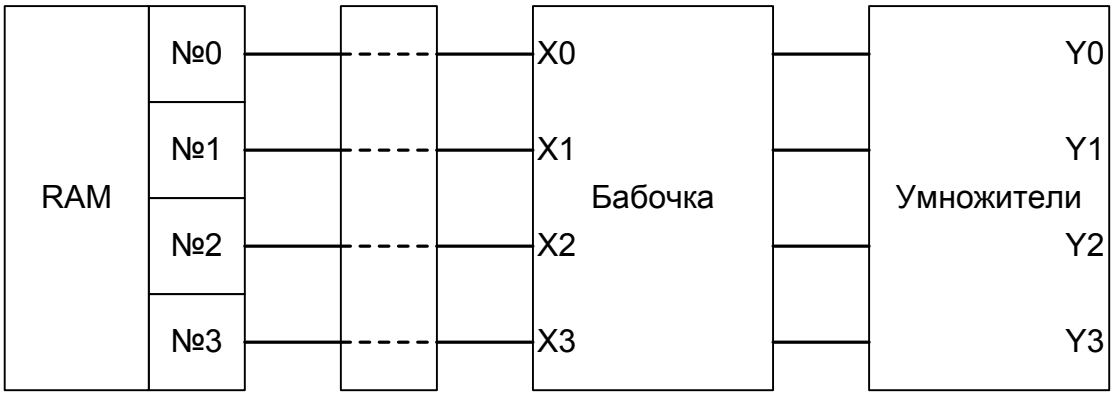
Следующие 128 повернув циклически банки памяти еще на 1 "шаг"



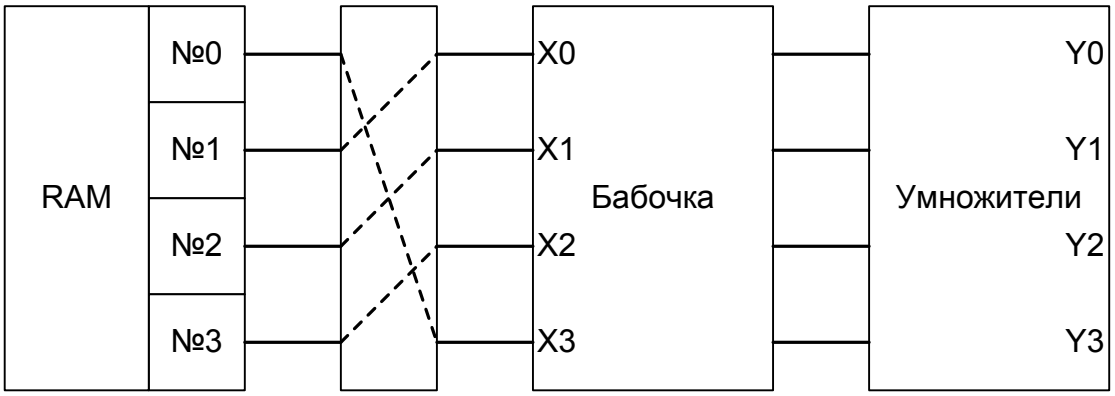
И так далее. При этом на все банки подается один и тотже адрес.

Теперь посмотрим как нам вычитывать данные при выполнении второго этапа. Для нулевой бабочки нам потребуются точки 0, 128, 256 и 384. Они находятся в банках №0, №1, №2 и №3 соответственно. Но вот адреса у них отличаются. 0-я точка лежит в банке №0 по адресу 0, 128-я в банке №1 по адресу 128, 256-я в банке №2 по адресу 256, 384-я в банке №3 по адресу 384. Когда мы доберемся до вычисления 128-й бабочки, то потребуются точки 512, 640, 768 и 896. Они лежат в банках №1, №2, №3, №0 соответственно. Адреса, как ни удивительно, 0, 128, 256 и 384.

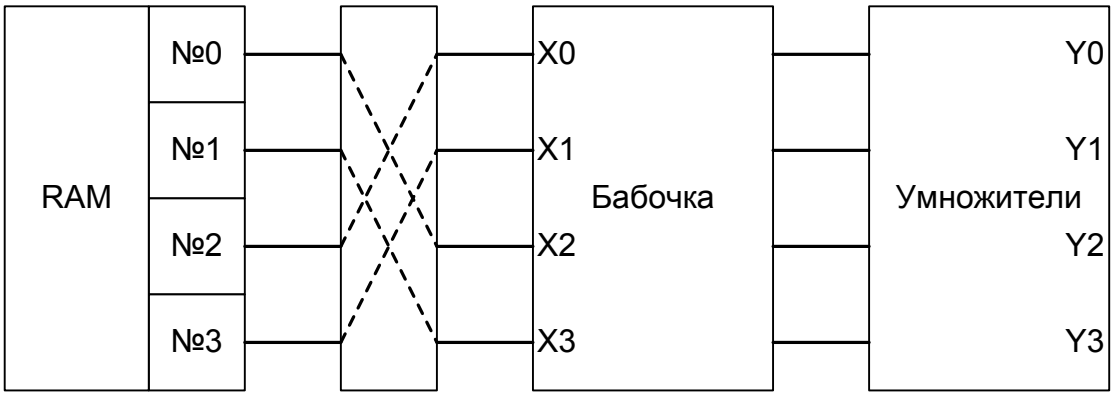
Для первых 128-ми бабочек



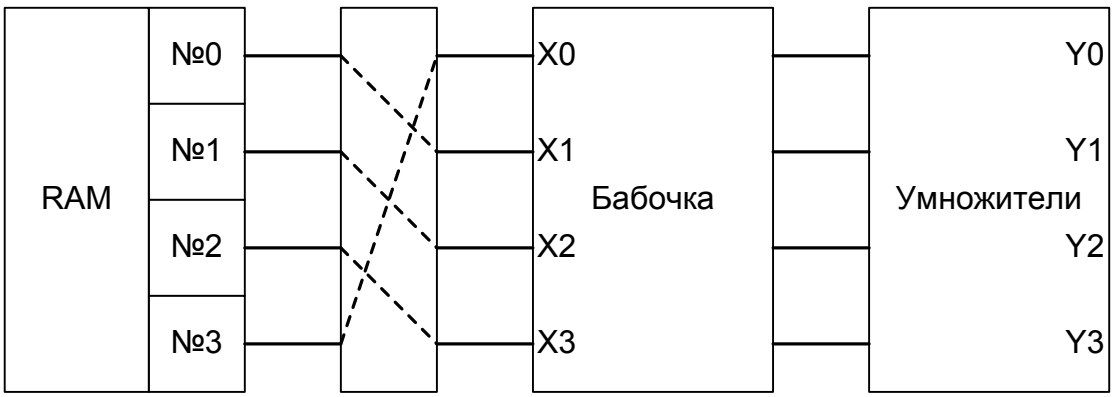
Для вторых 128-ми бабочек



Для третьих 128-ми бабочек



Для четвертых 128-ми бабочек



Таким образом, можно сделать следующее обобщение для 1-5 этапа (6-ой, к сожалению, выбивается из общего ряда). Можно условно разделить бабочки на блоки. На первом этапе у нас один блок из 512 бабочек. На втором этапе 4 блока по 128 бабочек, на третьем 16 блоков по 32 бабочки и т.д. На каждом этапе при чтении при переходе к следующему блоку мы циклически "вращаем" банки памяти относительно входов бабочки на один шаг. При записи циклическое "вращение" банков памяти производится в 4 раза чаще (т.е. при записи мы как будто бы уже на след. этапе).

На каждом этапе адреса для банков памяти при чтении сдвинуты друг относительно друга на 512 адресов для 1-го этапа (см. замечание ниже), на 128 для второго, на 32 для третьего и т.д. Начальные значения разные для разных блоков бабочек и определяются порядком записи точек на предыдущем этапе. При записи же, на все банки памяти подается один и тот же адрес.

Утверждение, что на 1-ом этапе при чтении адреса сдвинуты друг относительно друга на 512 просто удобно. Удобство в том, что при таком предположении 1-ый этап ничем не отличается от остальных 4-ех (6-ой, пока не трогаем) и при реализации схемы генерации адресов мы задаем начальные смещения для адресов 0, 512, 1024 и 1536 и затем при переходе от этапа к этапу делим их на 4 – никакой лишней логики. Учитывая, что одресная шина банков памяти всего 7 разрядов, младшие 7 раз рядов от 0, 512, 1024 и 1536 дадут нам необходимы 0,0,0,0 на адресных шинах для 1-го этапа.

Еще одно важное замечание. Как видно из рисунков миксер между памятью и входами бабочки отличается от миксера между выходами умножителей и памятью. Первый "вращает" входные данные относительно выходных, а второй наоборот – выходные данные относительно входных.