ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2021 Управление, вычислительная техника и информатика № 55

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 621.396.62
DOI: 10.17223/19988605/55/12

Д.В. Шехалев

СТРУКТУРА МАТРИЧНОГО УМНОЖИТЕЛЯ ДЛЯ ДЕКОДЕРОВ QC-LDPC КОДА С МИНИМАЛЬНЫМ РЕСУРСОМ ПАМЯТИ ПЛИС

Работа выполнена в рамках программы повышения конкурентоспособности ТГУ, НИР 8.2.37.2020.

Рассмотрена архитектура построения декодера QC-LDPC кодов на основе алгоритма распространения доверия для кода с изменениями скорости кодирования и размера блока. Показаны архитектуры матричного умножителя для высокопроизводительного декодирования с переменным размером блока. Предложено решение задачи оптимизации архитектуры матричного умножителя по критерию занимаемого ресурса ПЛИС.
Ключевые слова: помехоустойчивое кодирование; ПЛИС; коды с минимальной плотностью проверок на четность; матричное умножение.

Развитие современных систем связи идет по пути роста скорости передачи данных и увеличения помехоустойчивости канала данных. Это приводит к тому, что все более широко внедряются современные методы помехоустойчивого кодирования, основанные на итерационном декодировании. Одними из таких кодов являются коды с минимальной проверкой на четность (LDPC). Коды задаются проверочной матрицей размера $m \times n$, где m – количество бит в блоке кодирования, n – количество проверочных бит [1. С. 851]. В качестве примера можно привести проверочную матрицу LDPC кода (6, 12) вида:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \quad (1)$$

Наличие единицы в строке проверочной матрицы определяет участие бита данных в проверочном уравнении четности. Таким образом, проверочная матрица описывает связь символов блока кодирования друг с другом в n уравнениях четности. Скорость кодирования определяется по формуле

$$coderate = \frac{m-n}{m}.$$

Изменение размера блока или скорости кодирования кода осуществляется через изменение размеров и значений ячеек проверочной матрицы. Одним из основных алгоритмов декодирования таких кодов является алгоритм распространения доверия [1. С. 875]. Данный алгоритм обладает повышенными требованиями к вычислительной мощности модемного оборудования, так как ему необходимо большое количество итераций декодирования для получения хорошего энергетического выигрыша от кодирования.

В общем случае проверочная матрица может быть произвольной и не обладать структурой, что усложняет распараллеливание вычислений при декодировании. Но существует несколько семейств LDPC кодов, в которых эта проблема решена. Например, квазициклические коды с минимальной проверкой на четность (QC-LDPC) [2, 3].

Проверочная матрица таких кодов

$$H = \begin{bmatrix} H(1,1) & \dots & H(1,M) \\ \dots & \dots & .. \\ H(N,1) & \dots & H(N,M) \end{bmatrix}$$

состоит из $M \times N$ базовых матриц $H(i, j)$ ($i = \overline{1,\ N}$, $j = \overline{1,\ M}$) размером $z \times z$ элементов. Базовые матрицы могут быть двух видов: единичная матрица с циклическим сдвигом или нулевая матрица.

Таким образом, проверочная матрица задается порождающей матрицей сдвигов, отрицательное значение элемента которой соответствует нулевой базовой матрице и означает, что биты данного блока не участвуют в вычислениях. Остальные значения задают циклический сдвиг блока бит соответствующего координатам элемента матрицы сдвигов. Скорость кодирования определяется аналогично LDPC кодам:

$$coderate = \frac{M - N}{M}, \tag{2}$$

а размер базовой матрицы определяет размер блока декодирования:

$$m = M \times z. \tag{3}$$

Единичная матрица – это квадратная матрица, элементы главной диагонали которой равны единице, а остальные равны нулю. Следовательно, при любом циклическом сдвиге базовой матрицы количество единиц в ряде и столбце матрицы будет равно единице. Это приведет к тому, что после подстановки базовых матриц со сдвигами, определенными порождающей матрицей, ряды и столбцы проверочной матрицы будут независимы друг от друга в пределах размера z базовых матриц; таким образом, эти элементы могут быть вычислены одновременно. А значит, такой код обладает хорошими возможностями по распараллеливанию вычислений.

При этом сохраняется строгая структура кода, не зависящая от размеров базовой матрицы z, определяющей размер блока кодирования по формуле (3), и от количества рядов порождающей матрицы сдвигов, определяющей скорость кодирования по формуле (2).

Эти особенности делают QC-LDPC коды удобными для высокоскоростных систем передачи данных и для систем с адаптивной системой кодирования и модуляции (АСМ). Поэтому коды именно этого семейства приняты в современных стандартах связи в качестве помехоустойчивых кодов для каналов данных [2, 3].

Реализация высокопроизводительных QC-LDPC кодов на ПЛИС требует использования большого количества вычислительных ресурсов ПЛИС, особенно одного из критических ресурсов – встроенной блочной памяти, используемой для хранения промежуточных результатов вычисления.

Количество промежуточных результатов зависит от используемого алгоритма декодирования, а производительность декодера – от скорости реализации матричного перемножения и степени распараллеливания алгоритма. Одним из популярных способов реализации высокопроизводительного декодирования является последовательная реализация алгоритма распространения доверия с минимизацией количества хранимых промежуточных результатов [4]. Несмотря на последовательные вычисления, высокая производительность такого декодера обеспечивается матричным умножителем, основанным на мультиплексорах и регистрах ПЛИС при работе с широким словом данных, представляющим собой все элементы, соответствующие базовой матрице. При использовании программного слова меньшего размера производительность такого декодера резко снижается.

В статье [4] рассмотрен стандарт 802.11n [5], в котором определены четыре порождающих матрицы и три размера базовых матриц 27/54/81 бит. Тогда как, например, в стандарте 5G NR [2] количество возможных размеров базовых матриц 51, от 2 до 384 бит, количество порождающих матриц 16.

При реализации декодера с возможностью изменения размера блока подобный матричный умножитель будет занимать большое количество логических ресурсов ПЛИС, так как должен быть рассчитан на использование базовой матрицы наибольшего размера. Таким образом, подобный декодер можно использовать только в режиме высокой производительности, при значительном ресурсе логики ПЛИС и малом ресурсе блочной памяти.

В статье рассмотрена альтернативная архитектура ядра декодера, основанная на параллельной реализации алгоритма распространения доверия, для реализации QC-LDPC декодеров с возможностью адаптации производительности с учетом отводимого под декодер ресурса ПЛИС. Рассмотрена версия подобного ядра для применения в системах с высокой пропускной способностью. Исследованы пути оптимизации использования ресурса ПЛИС для декодеров подобного вида. Предложено решение для экономии ресурсов ПЛИС для декодера QC-LDPC кодов с перестраиваемыми размерами блока и скорости кодирования.

1. Декодирование LDPC кодов по алгоритму распространения доверия

Алгоритм распространения доверия [1. С. 875; 6] основан на использовании представления проверочной матрицы в виде графа Таннера (рис. 1).

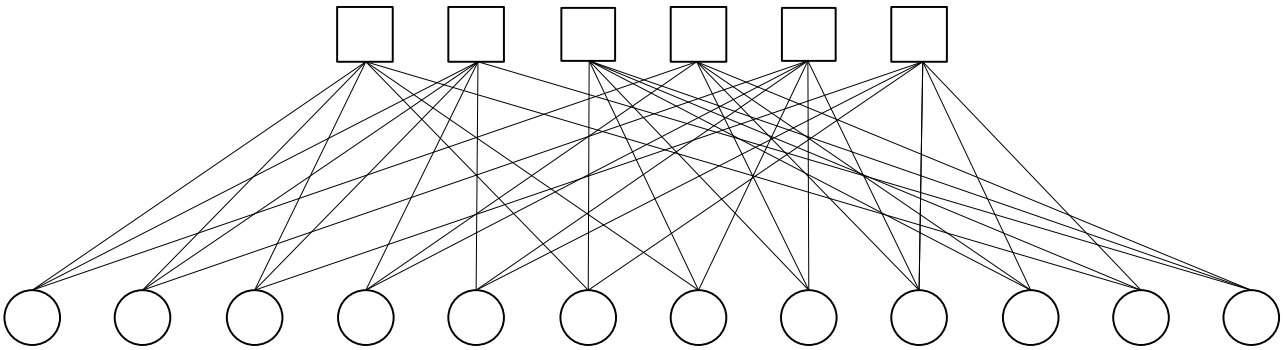

Рис. 1. Граф Таннера LDPC кода (6, 12) с проверочной матрицей по формуле (1)
Fig. 1. Tanner graph of LDPC code (6, 12) based upon check matrix equation (1)

В алгоритме вводится понятие узлов (node), кодовых узлов (cnode) и вертикальных узлов (vnode). Узел (node) представляет собой элемент памяти, в котором хранятся промежуточные результаты вычисления надежностей символов. Размер поля узлов в общем случае равен $m \times n$ и соответствует размеру проверочной матрицы. В операциях участвуют только те узлы, координатам которых в проверочной матрице соответствуют единицы.

Горизонтальные уравнения четности определяют собой кодовый узел (cnode) – результат вычисления совместной вероятности символов, входящих в уравнение четности. На основании совместной вероятности производится вычисление внешней вероятности каждого символа (extrincis).

Вертикальное сложение узлов определяет вертикальный узел (vnode) – сумма исходной вероятности символа со всеми внешними вероятностями этого же символа, полученными на предыдущем шаге. На основании суммы вероятностей одного символа производится вычисление внешней вероятности для следующего шага.

Таким образом, каждая итерация алгоритма декодирования состоит из двух шагов:

1. Горизонтальный шаг: вычисление кодового узла и внешних вероятностей символа горизонтальных связей, обновление поля узлов.

2. Вертикальный шаг: вычисление вертикального узла, определение внешней вероятности вертикальных связей, обновление поля узлов.

Спустя заданное количество итераций декодирования работа алгоритма заканчивается на вертикальном шаге, на котором, после вычисления вертикального узла, принимается жесткое решение по символу.

В общем случае LDPC кодов, при произвольной проверочной матрице, вычисления производятся последовательно символ за символом. Это требует наличия быстрого, произвольного и атомарного доступа к памяти узлов. Поэтому при разработке такого декодера нельзя использовать какие-либо упорядоченные структуры при реализации генераторов адресов памяти узлов. Это увеличивает сложность реализации декодера на ПЛИС.

В отличие от LDPC кодов, QC-LDPC коды обладают четкой структурой, позволяющей значительно увеличить скорость декодирования, так как в этом случае узлы, соответствующие разным базовым матрицам, могут обрабатываться одновременно.

2. Типовая архитектура вычислительного ядра декодера QC-LDPC кодов на ПЛИС

В QC-LDPC кодах обработка строк и столбцов может выполняться одновременно, это позволяет значительно увеличить скорость декодирования, но требует организации памяти узлов с возможностью одновременного доступа к данным. При этом нужно помнить, что в памяти узлов размером $M \times N \times z$ данные должны храниться постоянно и передаваться между итерациями. При этом узлы, относящиеся к одной базовой матрице, при обработке на горизонтальном шаге необходимо умножить на матрицу сдвига при чтении и обратной записи.

Типовая архитектура вычислительного ядра QC-LDPC декодера с использованием естественного параллелизма для реализации на ПЛИС приведена на рис 2.

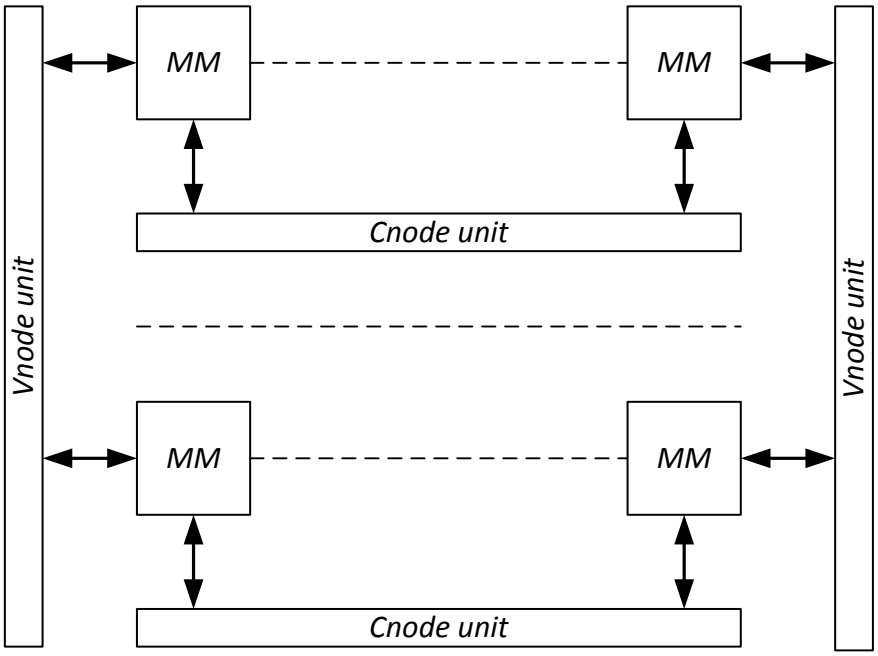

Рис. 2. Архитектора вычислительного ядра QC-LDPC декодера
Fig. 2. QC-LDPC decoder core architecture

На рис. 2 обозначены: *MM* – матричный умножитель, *Vnode unit* – модуль операций вертикального шага, *Cnode unit* – модуль операций горизонтального шага. Из рисунка видно, что общее ускорение операций декодирования будет определяться количеством узлов, одновременно обрабатываемых на горизонтальном и вертикальном шагах. Итоговый результат ускорения зависит от количества доступных ядер обработки данных и может варьировать в широких пределах, исходя из доступного вычислительного ресурса ПЛИС.

Но даже такой организации вычислений может быть недостаточно для реализации высоких скоростей декодирования. А в случае изменения скорости кодирования оно может быть неэффективным, так как зарезервированные ресурсы ПЛИС под быстрое декодирование кодов с низкой скоростью кодирования могут не использоваться большую часть времени.

Подобная ситуация возникает при высоком отношении сигнал–шум в радиоканале. В таких условиях используют модуляции с высоким индексом и большой скоростью кодирования ($\geq 0{,}75$), так как это позволяет максимизировать пропускную способность системы связи для текущего состояния радиоканала, и в этом случае требуется высокая производительность декодера. При малом отношении сигнал–шум используются модуляции с низким индексом и коды с низкой скоростью кодирования, в таких условиях требования к пропускной способности декодера снижаются.

Таким образом, разумно введение еще одной степени параллелизма: на уровне узлов базовых матриц, т.е. обрабатывать больше одного узла, относящегося к одной базовой матрице, за один такт системной частоты. В этом случае избыточный ресурс, зарезервированный для декодирования кодов с низкой скоростью кодирования, будет задействован для вычислений при декодировании кодов с любой скоростью.

При обработке больше одного узла базовой матрицы за один такт возникает сложность организации умножения базовых матриц на матрицу сдвигов, так как элементы матрицы сдвигов задаются с кратностью один бит, а гранулярность слова данных становится больше. Типовое решение данной проблемы заключается в разделении памяти узлов, относящихся к базовой матрице, на несколько блоков одинакового размера, использовании индивидуального генератора адреса узла и бареловского сдвигателя. Традиционная архитектура матричного умножения представлена на рис. 3.

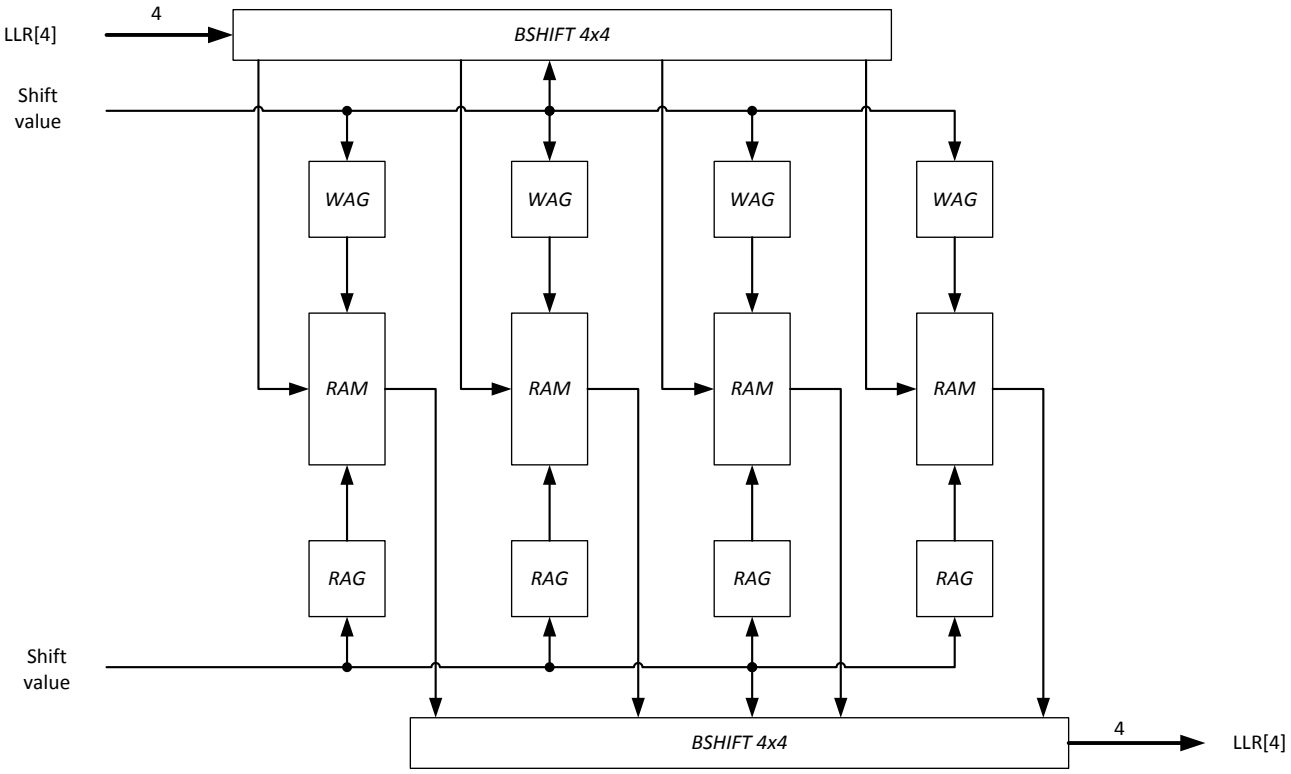

Рис. 3. Традиционный умножитель на матрицу сдвига для обработки 4 узлов за 1 такт частоты
Fig. 3. Classic shift matrix multiplier for 4 nodes per cycle processing

На рис. 3 обозначены: *BSHIFT* – бареловский сдвигатель, *RAG* / *WAG* – генератор адреса чтения / записи, *RAM* – массив памяти узлов. Данный умножитель позволяет выполнить умножение блока данных на матрицу сдвига с размером, кратным количеству данных. При этом умножение на матрицу сдвига происходит на лету, при чтении / записи памяти. Недостатками подобной схемы являются использование большого количества блоков памяти ПЛИС и неоптимальный режим их использования, так как типовой максимальный размер базовых блоков кодов составляет до 512 бит, а разрядность метрики узла 4–8 бит. Таким образом, требуемая емкость блока памяти составляет 2–4 килобита, а емкость блока памяти современных ПЛИС составляет 16 килобит. В результате большая часть па-

мяти не используется. Помимо этого, каждый блок памяти должен обладать собственным генератором адреса, что увеличивает сложность формирования адресов при изменении матрицы сдвига. В совокупности все это увеличивает требуемый ресурс ПЛИС для реализации умножителя. Достоинством схемы является минимальное количество тактов чтения / записи, необходимых для выполнения операции. Без учета латентности это время составит

$$Twrite/Tread = \frac{z}{node\_num},$$

где z – размер базовой матрицы, *node_num* – количество узлов, обрабатываемых за 1 такт частоты работы ядра декодера

3. Предлагаемое решение

Для оптимизации ресурсов памяти ПЛИС предлагается архитектура матричного умножителя, представленная на рис. 4.

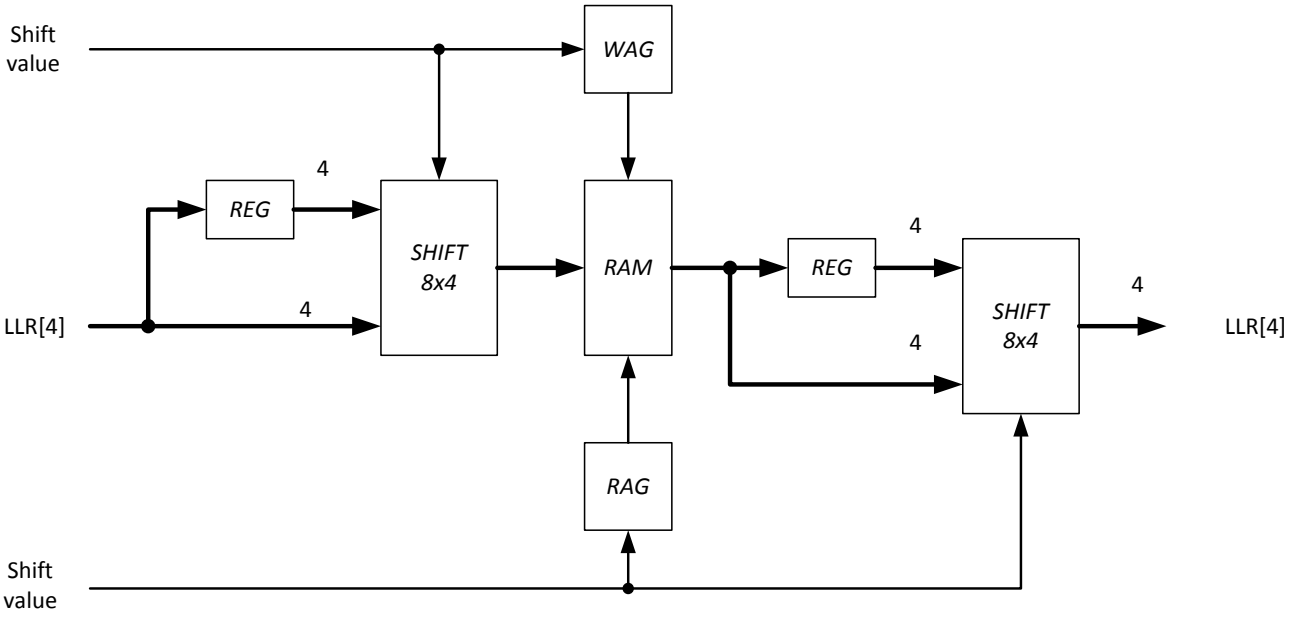

Рис. 4. Оптимизированный умножитель на матрицу сдвига
Fig. 4. Optimized shift matrix multiplier for 4 nodes per cycle processing

На рис. 4 обозначены: *REG* – регистр, *SHIFT* – регистр сдвига, *RAG* / *WAG* – генератор адреса чтения / записи, *RAM* – массив памяти узлов. Используется один блок памяти, хранящий несколько узлов в одной ячейке, по одному адресу. Для адресации используются один генератор адреса, дополнительный регистр и регистр сдвига. Предлагаемое решение занимает меньше ресурсов ПЛИС, чем традиционный умножитель. Но для формирования нужной последовательности узлов для обработки данных необходимо увеличить цикл чтения для предвыборки и цикл записи для записи последнего слова на один такт частоты относительно типовой схемы.

Таким образом, на каждом матричном умножении, при операциях чтения–записи, предлагаемая схема имеет задержку на два такта больше, чем типовой умножитель, но, благодаря оптимизации логических и трассировочных ресурсов ПЛИС, схема позволяет получить более высокие рабочие тактовые частоты и увеличить количество матричных умножителей для увеличения производительности декодера.

Помимо этого, так как в системе используется всего один генератор адреса, это упрощает формирование адресов при изменении матрицы сдвига и размера базовой матрицы, что делает предлагаемую схему умножителя удобной для декодеров с изменением параметров кодирования на лету.

4. Результаты сравнения традиционной и предлагаемой архитектур

Проведем сравнение архитектур по используемым ресурсам ПЛИС. В качестве примера рассмотрим базовый блок размером 1 024 бита, размер слова обработки 4 узла, размер метрики правдоподобия узла 4 бита.

Таблица 1

Сравнение ресурсов ПЛИС, требуемых для реализации классического и предлагаемого матричного умножителя, на логическом уровне

Составные модули	Традиционный умножитель	Предлагаемый умножитель
Память	4 блока памяти размером 256 × 4	1 блок памяти размером 256 × 16
Генератор адреса	8 генераторов адреса (счетчик по модулю 256 с установкой)	2 генератора адреса (счетчик по модулю 256 с установкой)
Сдвигатель 4x4	2 бареловских сдвигателя	2 сдвигателя
Регистры	0	2
Количество тактов на одну итерацию*	256	258

* – без учета латентности.

Для проверки предлагаемой реализации было разработано два варианта умножителя на матрицу сдвига на языке SystemVerilog. Результаты синтеза для ПЛИС Xilinx семейства Artix-7 в ПО Xilinx Vivado 2019.1.3 с настройками по умолчанию приведены в табл. 2, 3.

Таблица 2

Сравнение ресурсов ПЛИС Xilinx семейства Artix, требуемых для реализации классического и предлагаемого матричного умножителя

Ресурсы ПЛИС	Традиционный умножитель	Предлагаемый умножитель	Экономия ресурсов ПЛИС
RAMB18	4	1	75%
Slice LUT	364	167	54%
Slice REG	273	200	26%

Таблица 3

Сравнение показателей пиковой производительности реализаций классического и предлагаемого матричного умножителя

Показатели производительности	Традиционный умножитель	Предлагаемый умножитель	Выигрыш
Количество тактов на одну итерацию	261	263	–0,8%
Тактовая частота, МГц	262	281	8%
Производительность, млн умножений/с	1,007	1,07	6,3%

Из табл. 1–3 следует, что, несмотря на необходимость увеличения количества тактов, необходимых для выполнения матричного умножения, предложенное решение позволяет достичь более высокого значения тактовой частоты. Это позволяет нивелировать потенциальный проигрыш в производительности одного блока по тактовой частоте. С учетом меньшего количества использованных ресурсов ПЛИС для предлагаемого решения можно получить значительный выигрыш по производительности декодера или значительно уменьшить требуемый для декодера ресурс ПЛИС при сохранении той же производительности.

Заключение

В статье рассмотрена задача оптимизации декодера QC-LDPC кодов по критерию оптимального использования ресурсов ПЛИС. Предложена схема умножителя на матрицу сдвига для быстродействующих декодеров с поддержкой переменного размера блока и скорости кодирования. Показан выигрыш в требуемом ресурсе ПЛИС, достигаемый при использовании умножителя предлагаемой архитектуры.

ЛИТЕРАТУРА

1. Lin S., Costello D.J. Error Control Coding. 2nd ed. Upper Saddle River, NJ : Prentice-Hall, 2004. 1272 p.
2. ETSI 3rd Generation Partnership Project; Technical specification 5G; NR; multiplexing and channel coding (3GPP TS 38.212 version 15.8.0 Release 15). 2020. P. 18–25.
3. IEEE Standard for Air Interface for Broadband Wireless Access Systems // IEEE Std 802.16–2012. 2012. P. 1285–1289.
4. Mhaske S. et al. High-Throughput FPGA-Based QC-LDPC Decoder Architecture // 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall). Boston, MA : IEEE, 2015. P. 1–5.
5. IEEE Standard for Information technology. Telecommunications and information exchange between systems. Local and metropolitan area networks. Specific requirements // IEEE Std 802.11–2012. 2012. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. P. 2304–2306.
6. Jinghu C., Fossorier M.P.C. Near optimum universal belief propagation based decoding of low density parity check codes // IEEE Transactions on Communications. 2002. V. 50 (3). P. 406–414.
7. Chavet C., Coussy P. Advanced Hardware Design for Error Correcting Codes. Cham : Springer, 2015. IX, 192 p.

Поступила в редакцию 2 ноября 2020 г.

Shekhalev D.V. (2021) MATRIX MULTIPLIER ARCHITECTURE FOR QC-LDPC CODE WITH MINIMUM FPGA BLOCK RAM RESOURCE USING. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 55. pp. 103–111

DOI: 10.17223/19988605/55/12

Forward error correction coding based upon iterative decoding are widely used in modern communication systems. Low Density Parity Check codes (LDPC) is one of it. These codes have increased requirements for the processing performance of modem equipment, since they require a large number of decoding iterations to obtain a good coding gain. The LDPC check matrix can be arbitrary and vary with other encoding parameters in general. So, decoding such codes is performed sequentially symbol by symbol and does not allow high throughput from the decoder.

There are some classes of LDPC codes that have good capabilities for paralleling processing and changes in code characteristics on fly. Quasi-Cyclic Low Density Parity Check codes (QC-LDPC) is one of it. The check matrix of such codes is a shift matrix based upon base matrices array and has clear structure. It makes possible to increase the performance of the decoder due to its natural parallelism at the level of base matrices.

However, requirements of communication system performance can be so high that despite this feature, decoder performance may not be sufficient. An additional level of parallelism based upon elements of the base matrices is required. It's realized through a matrix multiplier unit based on block memory. The classic solution of it involves the use of a large amount of FPGA memory blocks used in a non-optimal mode. Each memory unit use an own write/read address generator. Multiplication by shift matrix is carried out due to barrel shifter. This architecture provides high throughput but requires a lot of FPGA resources. Especially, the block ram units which is one of most critical FPGA resources.

There is an alternative solution for architecture of matrix multiplier. The proposed solution is based upon one larger-width memory unit with single read/write address generator, bit shifter and registers for intermediate data storing. This solution has a similar bandwidth as classic one and requires a significantly smaller number of FPGA resources. Proposed solution shows reduction of FPGA block memory and LUT using up to 4 and 2 times respectively and obtain increased performance due to clock frequency increase up to 5-6% at same time. Moreover, the advantage of it is the ease of changing base matrix size and the shift value. Therefore, the proposed solution is recommended for constructing high throughput decoders with adaptive coding support.

Keywords: forward error correction; FPGA; LDPC; matrix multiply.

SHEKHALEV Denis Vladimirovich (Senior Engineer, Scientific and Educational Center "Engineering Center of Microwave Technic and Technology", National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: diod2003@list.ru

REFERENCES

1. Lin, S. & Costello, D.J. (2004) *Error Control Coding.* 2nd ed. Upper Saddle River, NJ: Prentice-Hall.
2. ETSI 3rd Generation Partnership Project. (2020) *Technical specification 5G; NR; multiplexing and channel coding (3GPP TS 38.212 version 15.8.0 Release 15)*. pp. 18–25.
3. *IEEE Standard for Air Interface for Broadband Wireless Access Systems (2012)*. IEEE Std 802.16-2012. pp. 1285–1289.
4. Mhaske, S. et al. (2015) High-Throughput FPGA-Based QC-LDPC Decoder Architecture. *2015 IEEE 82nd Vehicular Technology Conference* (VTC2015-Fall). Boston, MA, USA: IEEE. pp. 1–5.

5. IEEE Standard for Information technology. (2012) Telecommunications and information exchange between systems. Local and metropolitan area networks. Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012*. pp. 2304–2306.
6. Jinghu, C. & Fossorier, M.P.C. (2002) Near optimum universal belief propagation based decoding of low density parity check codes. *IEEE Transactions on communications*. 50. pp. 23–31.
7. Chavet, C. & Coussy, P. (2015) *Advanced Hardware Design for Error Correcting Codes*. Heidelberg; New York; Dordrecht; London: Springer.