Маршруты высокоуровневого синтеза

Иван Селиванов (Москва)

Обычно сложные алгоритмы сначала моделируются и тестируются в Matlab или на языке C/C++, а затем отлаженное высокоуровневое описание вручную транслируется в язык описания аппаратуры, такой как VHDL или Verilog, для последующего RTL-синтеза. У этого подхода есть свои достоинства, но есть и недостатки. К ним относятся: большая трудоёмкость этого процесса, отсутствие гибкости, жёсткая привязка к реализации, сложность внесения изменений в проект и т.п. Все это, вкупе с растущей сложностью проектируемых устройств, заставляет искать новые тракты проектирования, переходить к средствам высокоуровневого синтеза.

Традиционный подход

Практически все большие проекты сегодня основаны на множестве сложных алгоритмов. Обычно сначала происходит отладка и доказательство правильности работы алгоритмов либо в программе MATLAB фирмы MathWorks (www.mathworks.com), являющейся стандартом де-факто в данной области, либо с использованием обычного C/C++.

При этом происходит примерная оценка времени выполнения и потребления ресурсов для различных частей проекта. С учётом этих данных проект разбивается на программные и аппаратные блоки, после чего алгоритмическое представление аппаратного блока, использующее операции с плавающей точкой, вручную преобразуется в HDL-описание на RTL-уровне, использующее только битовые операции. Затем из этого RTL-кода синтезируется описание на вентильном уровне при помощи обычных средств синтеза, таких как Altera Quartus, Xilinx ISE или Mentor Graphics Precision Synthesis (рис. 1).

Несмотря на то что в прошлом этот метод был достаточно эффективен, у него есть ряд серьёзных недостатков, которые в совокупности с постоянным ростом объёмов и сложности проектов делают его дальнейшее использование всё более и более проблематичным:

- *разница в подходах к проектированию* между алгоритмистом, работающим с непривязанными ко времени алгоритмами, и инженером, создающим описание устройства на VHDL/Verilog. По сути они разговаривают на разных языках, и один не всегда понимает, что хотел сказать другой;
- *процесс создания RTL-кода очень трудоёмок*: поскольку описание алгоритма для MATLAB (или на C/C++) радикально отличается от RTL-кода, самостоятельное преобразование одного в другое – процесс очень долгий, трудоёмкий, и вероятность появления ошибок на этой стадии очень велика;
- *долгая верификация RTL-кода*: моделирование больших проектов, представленных на RTL-уровне, происходит крайне медленно и требует много вычислительных ресурсов;
- *сложно проанализировать различные варианты реализации проекта*. Один и тот же алгоритм может иметь множество аппаратных реализаций. Некоторые его части могут выполняться как параллельно, так и последовательно. Отдельные блоки или даже весь проект можно реализовать и в виде конвейера. Причём результат применения той или иной реализации не всегда предсказуем, а первоначальная оценка занимаемой площади и времени выполнения, мягко говоря, весьма приблизительна, поэтому может оказаться, что проект выполняется слишком медленно или занимает слишком большую площадь кристалла. Чтобы проанализировать несколько вариантов реализации одного и того же алгоритма, нужно каждый раз модифицировать, а затем и верифицировать RTL-код, что весьма непросто и требует огромного количества времени. Из-за ограничений во времени разработчики могут опробовать ограниченное количество вариантов, и далеко не всегда оптимальное решение будет среди них;
- *сложно вносить изменения*. Если во время работы над проектом в ТЗ вносятся какие-то изменения, то их достаточно легко внести в MATLAB, но для передачи на уровень ниже, в RTL-код, может потребоваться много времени и усилий. Это особенно важно в таких областях, как беспроводная связь, т.к. стандарты и протоколы всё время развиваются и меняются;
- *код зависит от реализации*. Например, RTL для ПЛИС может отличаться от RTL для реализации

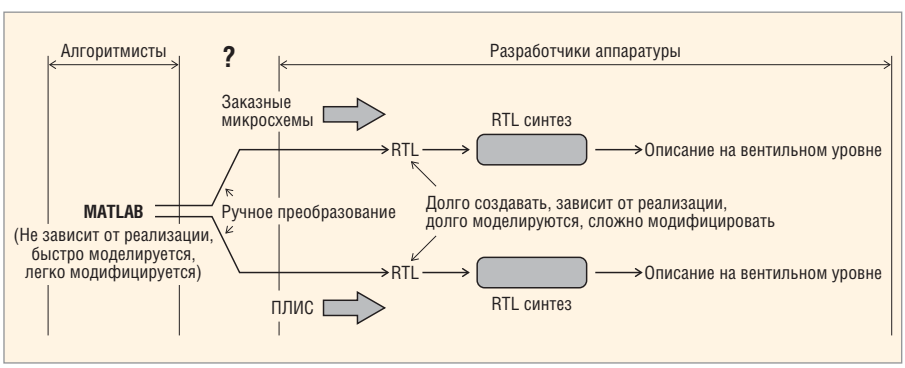


Рис. 1. Традиционный маршрут проектирования

на заказной микросхеме. Другими словами, если использовать один и тот же код и для ПЛИС, и для заказных схем, то, скорее всего, реализация на заказной схеме будет неполноценной из-за ограничений по скорости, присущих ПЛИС. И наоборот, на ПЛИС можно достичь необходимого быстродействия путём параллельного выполнения большого количества операций, однако на заказных схемах такое распараллеливание может и не понадобиться. Из-за этого становится крайне сложным, если не невозможным, изменить реализацию RTL-описания сложного проекта. А теперь представим себе организацию, занимающуюся прототипированием заказных микросхем на ПЛИС. Они могут сделать превосходный прототип, удовлетворяющий всем требованиям, однако чтобы реализовать всё то же самое на заказных схемах, нужно вернуться обратно на уровень MATLAB.

Важно понимать, что привязка к реализации выходит за границы простого «ПЛИС или заказные микросхемы». Даже для одного и того же кристалла, если в различных применениях устройства для обработки данных используется разная последовательность алгоритмов, то могут потребоваться разные микроархитектуры для каждого конкретного применения.

Всё это происходит от слишком большой разницы в уровнях абстракции между MATLAB и RTL. Многие из вышеперечисленных проблем не были бы так существенны, если бы появился некоторый промежуточный уровень, не абстрагированный от технологии изготовления, но не зависящий от конкретной реализации.

Проанализировав всё вышесказанное, можно выделить в традиционном методе проектирования три основные стадии:

- разработка алгоритма в MATLAB;
- преобразование нетактированного алгоритма MATLAB в тактированное RTL-описание (включая верификацию и анализ различных вариантов реализации);
- преобразование RTL-уровня в вентильный с использованием уже знакомой технологии RTL-синтеза.

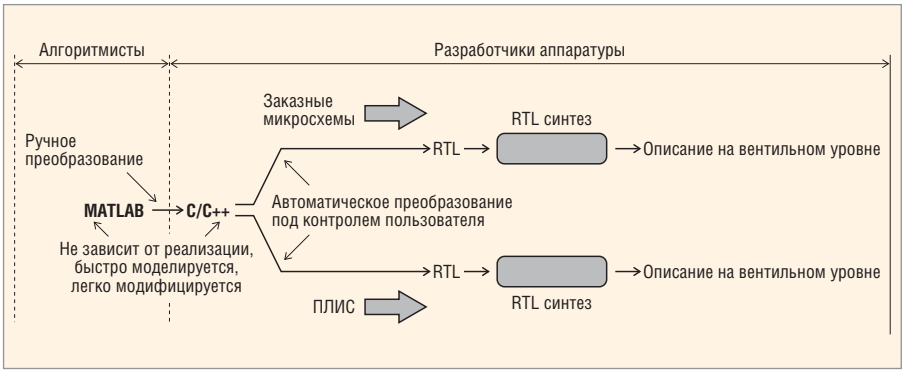


Рис. 2. Идеальный маршрут проектирования

Первая и последняя стадии всем хорошо знакомы и отлично работают. «Узкое место» в процессе – вторая стадия, ручное преобразование нетактированного алгоритма MATLAB в тактированное RTL-описание, включая верификацию, анализ различных вариантов реализации и внесение изменений в алгоритм на RTL уровне.

Идеальный подход

В идеале процесс проектирования должен основываться на языке C, который используется для разработки встроенного программного обеспечения вот уже более 10 лет и поэтому знаком инженерам, проектирующим аппаратуру, гораздо лучше, чем MATLAB. Поскольку разработка и отладка алгоритма происходит всё же в основном в MATLAB, а не на C, всё-таки придётся проводить преобразование из одного кода в другой, но поскольку концептуально MATLAB и C очень близки, такую операцию несложно проделать даже вручную.

После верификации описание на языке C можно было бы использовать для автоматической генерации RTL-кода, который затем предавать в существующие средства RTL-синтеза (рис. 2).

Можно было бы, конечно, из C-описания создавать сразу вентильный уровень, но RTL-уровень является некоторой промежуточной ступенью, которая:

- отлично вписывается в уже существующий и знакомый всем процесс проектирования;
- используется для верификации и оценки решений, принятых в ходе синтеза. RTL-код синтезируется и моделируется быстрее, чем вентильное описание, поэтому если потребуется вернуться на шаг обратно, то с этого уровня это будет сделать проще;

и самое главное:

- является тем уровнем абстракции, на котором разные функциональные блоки одного проекта собираются в единое целое. Это касается не только блоков, разработанных инженерами конкретного предприятия, но и блоков, разработанных сторонними фирмами, так называемых IP-блоков.

Всё это означает, что промежуточный RTL-уровень по крайней мере сегодня нельзя исключать из процесса проектирования, так как это важная точка для объединения и верификации компонентов электронного устройства. Инженеры могут в полной мере воспользоваться такими преимуществами существующего процесса проектирования, как встраивание тестовой логики, анализ потребляемой мощности и т.п.

Данный метод позволит избежать всех существующих на сегодняшний день проблем в этой области:

- *Возникает понимание между алгоритмистом и разработчиком аппаратуры.* Поскольку и те, и другие знают и понимают C, две эти области становятся концептуально гораздо ближе друг к другу;
- *Создавать C-описание быстро.* Поскольку исторически MATLAB и C близки друг к другу и C-описания гораздо компактнее их RTL-эквивалентов, преобразование кода MATLAB в код C происходит быстро и безболезненно. Кроме того, благодаря среде MATLAB Simulink появляется дополнительные возможности для интеграции и верификации системы;
- *Верифицировать C быстро и легко.* C-представление моделируется на несколько порядков быстрее, чем

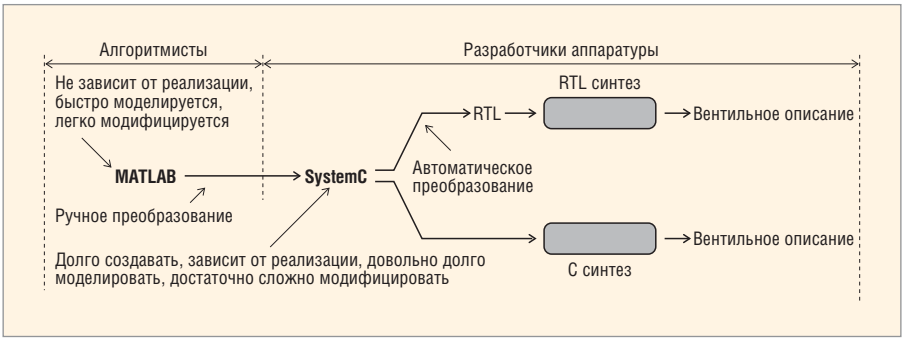


Рис. 3. Проектирование в SystemC

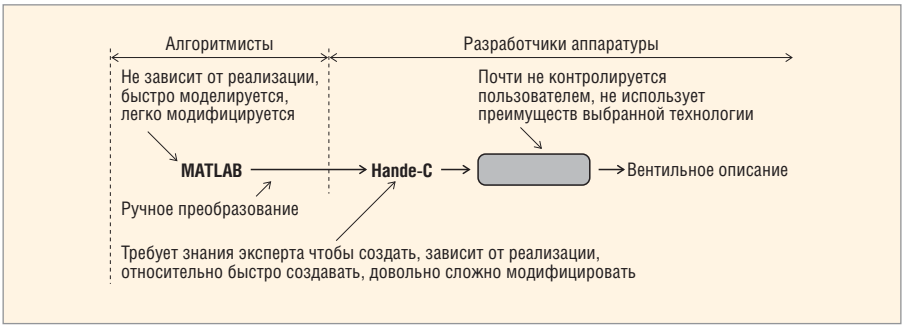


Рис. 4. Проектирование в Handel-C

RTL-описание, причём чем больше проект, тем больше становится разница в скорости;

- *Можно легко опробовать несколько вариантов реализации*, поскольку изменение и повторная верификация С-кода не занимает очень много времени. Это значит, что можно проверить гораздо больше альтернативных вариантов и возрастают шансы найти оптимальный;
- *Несложно реализовать изменения в спецификации*. Если во время работы над проектом в спецификацию необходимо внести какие-то изменения, то их несложно реализовать в MATLAB-модели и затем перенести и в С-описание;
- *С не зависит от реализации*. Ключевая особенность этого процесса проектирования состоит в том, что С-представление полностью абстрагировано от конечной реализации. Это значит, что все данные о способах воплощения кода содержатся не в самом коде, а задаются в процессе преобразования С в RTL посредством некоторых команд или опций в среде трансляции. В свою очередь это означает, что одно и то же С-описание может использоваться как для ПЛИС, так и для заказных схем, а также и для различных вариантов микроархитектуры.

SystemC

Существуют два процесса проектирования, основанных на одном из наиболее известных языков описания аппаратуры, который является расширением C++, – SystemC. В обоих вариантах описание из MATLAB транслируется вручную в SystemC, а затем верифицируется в симуляторе. Далее, используя существующие средства, описание автоматически детализируется либо до вентильного, либо до RTL-уровня (рис. 3). Поскольку язык SystemC создавался специально для описания аппаратуры, в нём есть множество полезных типов и возможностей. Например, возможность легко указать, что будет происходить при переполнении, или задать режим округления, а также поддержка различных типов передачи данных между процессами (например, FIFO).

Главным преимуществом SystemC является то, что он моделируется примерно на порядок быстрее, чем описание на VHDL или Verilog с тем же уровнем абстракции. Однако для написания кода, пригодного для моделирования и синтеза, требуется примерно столько же усилий, как и для написания собственно RTL-кода. Кроме того, при таком подходе микроархитектура устройства жёстко задана и для её изменения нужно переписывать код, то есть налицо практически все недостатки традиционного метода.

Handel-C

Альтернативный подход – использование собственных языков и компиляторов. Примером может служить Handel-C, унаследовавший синтаксис и управляющие конструкции языка С, что упрощает его понимание программистами и инженерами. В дополнение к типам данных, предназначенных специально для описания аппаратуры, в этом языке есть специальные ключевые слова для описания передачи данных и поддержки параллельного программирования. MATLAB-описание нужно вручную транслировать в Handel-C, затем верифицировать в симуляторе и синтезировать вентильное описание (рис. 4). Для моделирования и синтеза необходимо использовать пакет DK Design Suite компании Celoxica.

Теоретически преобразование из MATLAB в Handel-C должно проходить легко, но на практике для создания корректного описания, пригодного для программы синтеза, требуется большая работа опытного пользователя.

Так же как и с SystemC, псевдо-временные конструкции, необходимые для моделирования и синтеза Handel-C, незнакомы как алгоритмистам, так и разработчикам аппаратуры. И опять же, вся микроархитектура проекта описана в коде. Более того, пользователь практически не может контролировать процесс синтеза и не имеет возможности использовать преимущества той или иной технологии, как то: использование встроенных умножителей, RAM, и т.п. Всё это означает, что пользователю нужно проделывать нетривиальные манипуляции с кодом, чтобы выполнить требования по скорости и площади устройства.

Catapult C

Как уже упоминалось выше, основной проблемой существующих методов проектирования с использованием С является то, что микроархитектура жёстко задана в коде, в результате чего он должен изменяться для каждого конкретного варианта реализации устройства. Основным отличием пакета Catapult C фирмы Mentor Graphics является то,

что C-описание, используемое Catapult, практически не отличается от того, которое алгоритмист написал бы просто для описания поведения алгоритма, даже не задумываясь о том, каким образом этот алгоритм будет реализован в аппаратуре. Вместо того чтобы описывать микроархитектуру в исходном коде, тем самым привязываясь к конкретному варианту реализации, все данные задаются в самом средстве синтеза через простой и интуитивно понятный интерфейс.

В Catapult используется стандартный C++ с добавлением типов SystemC, чтобы переменным и константам можно было присваивать требуемую разрядность.

Огромным преимуществом использования Catapult C является то, что множество компаний уже описывают свои алгоритмы на C, потому что его легко написать и быстро промоделировать. Единственное изменение, которое обычно требуется внести в такой код для использования его в Catapult C, – это прагма, обозначающая верхний уровень проекта (всё, что по иерархии находится уровнем выше, считается частью тестирующей программы).

Другое важное преимущество – простой и интуитивно понятный интерфейс. Некоторые средства настолько сложны, что нужны месяцы, чтобы освоить их на должном уровне, а для ознакомления с Catapult нужно всего несколько дней.

После загрузки кода в Catapult C пользователь может попробовать различные варианты микроархитектуры устройства и сравнить их быстродействие, площадь и другие параметры. Можно легко ассоциировать порты с регистрами или RAM. Программа автоматически находит такие конструкции, как различные типы циклов, и позволяет для каждого отдельно взятого цикла указать, что с ним нужно сделать – выполнить все операции цикла параллельно, частично параллельно или же выполнять цикл последовательно. Кроме того, пользователь по своему желанию может на основе циклов создавать конвейеры с различным интервалом запуска, организовывать совместное использование одного ресурса несколькими операциями, управлять использованием различных ресурсов и т.п.

Результат всех этих манипуляций можно увидеть практически моментально – от нескольких секунд до нескольких минут, в зависимости от размера и сложности проекта. Различные варианты настроек проекта можно сохранять, сравнивать их параметры, задавать им уникальные имена и затем возвращаться к наиболее интересному варианту для, возможно, дальнейших экспериментов с целью поиска оптимального решения. Всё это абсолютно не реально осуществить путём самостоятельной трансляции MATLAB-описания в RTL-код.

Ну и, наконец, самое важное отличие подхода, проповедуемого Mentor Graphics, состоит в том, что исходный код не содержит абсолютно никакой информации о микроархитектуре устройства, вся она задаётся уже изнутри пакета, что позволяет быстро и легко переключаться между различными технологиями и микроархитектурами.

Теперь несколько слов о том, как всё это работает. Основная единица

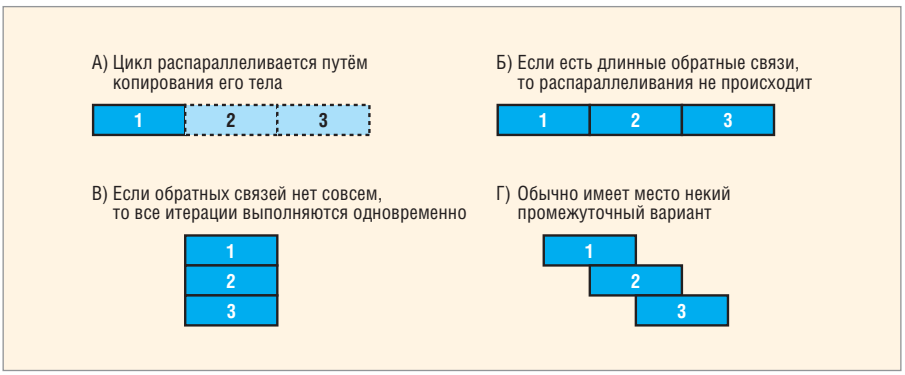


Рис. 5. Проектирование в Catapult C

проекта для Catapult C – циклы. И большинство операций (но не все) по оптимизации проекта по скорости либо по площади обычно производится с ними. Обычно в алгоритме присутствует множество циклов, самый главный из которых – тело основной функции. В него уже вложены все остальные циклы и операторы. Программа анализирует циклы и определяет, какие из них зависят друг от друга, а какие могут выполняться параллельно. Параллельные циклы автоматически «склеиваются», хотя пользователь может включить опцию, запрещающую эту операцию. Далее пользователь анализирует проект и решает, какие оптимизации и к каким циклам можно применить.

По умолчанию все итерации цикла выполняются последовательно, одна за другой. Однако, если количество итераций невелико, а в теле цикла небольшое количество операций и обрабатываются только внутренние данные (т.е. нет операций ввода-вывода), то для увеличения быстродействия обычно применяется распараллеливание цикла.

Если в цикле нет обратных связей, т.е. каждая последующая итерация не использует данные предыдущих, то в результате распараллеливания тело цикла просто копируется столько раз, сколько в нем итераций. Если же обратные связи есть, то каждая последующая итерация цикла сдвигается во времени относительно предыдущей таким образом, чтобы необходимые результаты предыдущих итераций были готовы к тому моменту, когда они понадобятся (рис. 5).

Если количество операций велико или не определено, то можно использовать частичное распараллеливание – тело цикла копируется столько раз, сколько будет указано пользователем, и выполняется параллельно.

Кроме обратных связей есть и другой фактор, препятствующий распараллеливанию циклов, – ресурсы кристалла. Часто также узким местом является память, т.к. количество обращений к одному блоку памяти ограничено. У пользователя есть возможность по-другому организовать хранение массивов внутри блока памяти или увеличить ширину слова, что при некоторых условиях позволит увеличить количество одновременных обращений к одному и тому же блоку. Можно также использовать двухпортовую память вместо однопортовой, однако эти меры не всегда оказываются достаточными.

Когда распараллеливание цикла по каким-либо причинам невозможно, можно прибегнуть к созданию конвейера – каждая последующая итерация запускается через некоторое количество тактов (определяется пользователем) после запуска предыдущей.

По мере работы над проектом пользователь может анализировать полученные решения, оценивать занимаемую площадь, быстродействие, видеть, какая часть проекта выполняется дольше всего, а какая потребляет много ресурсов. Таким образом можно находить проблемные места в самом алгоритме и по возможности улучшать их.

В конечном итоге пользователь получает на выходе следующее:

- код на SystemC, предназначенный для совместного моделирования исходного C++-описания алгоритма и его RTL-варианта с использованием той же самой тестирующей программы, что использовалась и для отладки самого алгоритма на C++. Программа получает тестовые воздействия из программы-тестера и передаёт их в C- и в RTL-описания, а затем сравнивает ответы. Тем самым можно легко проверифицировать синтезированное описание;
- собственно RTL-код на VHDL или Verilog для последующей передачи в программу RTL-синтеза;
- отчёт о площади и использованных элементах, о быстродействии, набор скриптов для программы RTL-синтеза и т.п.

Заключение

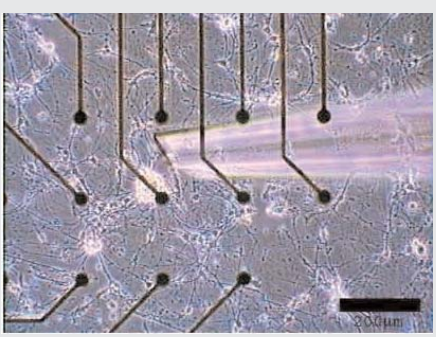
Основная разница между различными процессами проектирования заключается в уровне абстракции синтезируемого описания алгоритма. Например, несмотря на то, что SystemC располагает большими возможностями для моделирования на системном уровне, его синтезируемое подмножество имеет гораздо более низкий уровень абстракции. Также, хотя Handel-C-описания больше похожи на простой C/C++, чем SystemC, что значит, что они моделируются гораздо быстрее, уровень абстракции синтезируемого подмножества всё-таки ниже, чем хотелось бы. Недостаточный уровень абстрагированности от аппаратуры делает описания на SystemC и Handel-C зависящими от конкретного варианта исполнения. Как следствие, их сложнее создавать и модифицировать, уменьшается гибкость и у пользователя становится меньше возможностей для сравнения различных микроархитектур и сложнее переход на другие технологии.

У описаний, создаваемых для Catapult C, уровень абстракции выше, что избавляет его от всех вышеперечисленных недостатков. Эти не зависящие от реализации модели просты и компактны, их легко писать и модифицировать. Благодаря тому, что микроархитектура определяется внутри оболочки синтеза, пользователю проще анализировать и сравнивать различные реализации и использовать различные технологии и интерфейсы. В конечном результате Catapult C ускоряет процесс проектирования и повышает гибкость проектов по сравнению с другими методами высокоуровневого синтеза.

Новости мира News of the World Новости мира

Продемонстрировано хранение данных в живых нейронах

Двое израильских учёных из Университета Тель-Авива продемонстрировали возможность сохранения информации в живых нейронах. Результаты этого исследования могут стать новым шагом на пути к пониманию принципов функционирования человеческого мозга и даже намекают на возможность создания уже в обозримом будущем киборгоподобных систем, объединяющих живые материалы и чипы памяти. Для ввода информационной последовательности учёные в определённом порядке воздействовали на нейроны химическими соединениями, а образовавшиеся в результате возбуждения в нейронных сетях регистрировали при помощи матрицы электродов.



В результате опыта было продемонстрировано, что три введённые информационные последовательности продолжали сохраняться в нейронных сетях на протяжении более чем 40 ч, не накладываясь друг на друга. Подобные исследования с нейронами уже проводились, но израильским учёным впервые удалось добиться сохранения информации в течение относительно продолжительного времени, используя в качестве носителя культуру нейронов.

tgdaily.com

Intel представила первые одночиповые двухканальные TV-демодуляторы

Компания Intel известна, прежде всего, как производитель процессоров, но сфера её деятельности на этом вовсе не ограничивается. С одной стороны, компания внимательно отслеживает современные тенденции в индустрии цифровых устройств, предлагая собственные решения для наиболее перспективных, на её взгляд, направлений, а с другой – выпуском таких решений, многие из которых становятся стандартами, активно влияет на рынок. Одно из относительно новых направлений, осваиваемых Intel, – цифровое телевидение, и для включения соответствующих функций в разнообразные потребительские устройства компания выпустила первые одночиповые двухканальные модуляторы.

Согласно данным компании, демодуляторы Intel CE 6250 Dual Channel COFDM и Intel CE 6251 Dual Channel Diversity-Enabled COFDM отличаются небольшими размерами и пониженным энергопотреблением, что позволяет применять их как в стационарных, так и в портативных мобильных устройствах, оснащённых функциями приёма цифрового телевидения. Это могут быть, например, настольные цифровые видеомагнитофоны, встроенные цифровые телевизоры или компьютеры с поддержкой приёма цифрового TV. Наличие двух каналов позволяет расширить набор функций устройств – скажем, возможностью записи одного канала при одновременном просмотре другого или выводом «картинки в картинке».

Демодуляторы выполнены по КМОП 0,13-мкм технологии в 80-контактных низкопрофильных корпусах размерами 10 × 10 мм, заявленная потребляемая мощность – не более 280 мВт, с поддержкой энергосберегающих режимов «сна». Кроме собственно чипов, предлагаются также наборы для разработчиков, включающие платы с референсным дизайном, ПО и интерфейс к компьютеру, доступны также исходники драйверов.

intel.com

3D-камера размером с пуговицу

Японцы вновь удивили весь мир невероятным изобретением. Исследователи Университета в Осаке разработали ультратонкую камеру, способную делать трёхмерные снимки. Секрет в том, что в миниатюрном устройстве размещается девять линз, каждая из которых делает снимок с небольшим смещением. Полученные данные обрабатываются специальной программой, которая определяет расстояние между объектами сцены, их цвет и другие особенности. В результате при помощи такой камеры можно получить настоящее трёхмерное изображение.

Интересно, что для разработки технологии использовались данные о зрении насекомых. Программа, анализирующая

сцену, имитирует процесс распознавания положения, формы и цвета объектов насекомыми.

Сама камера размером не больше пуговицы от рубашки, что даст возможность использовать её в современных портативных устройствах, например, в мобильных телефонах или системах навигации для автомобилей. Правда, пока что качество получаемых при помощи камеры изображений совсем невелико – 1,1 мегапиксела, однако исследователи говорят, что его можно улучшить, увеличив число линз. Однако для этого нужно чётко знать, где будет использоваться система. Например, если речь идёт о камере наблюдения на автостоянке, то тут вполне можно ограничиться невысоким качеством.

www.technologyreview.com

Самый маленький голубой диод от Sharp

Японская корпорация Sharp, отметившаяся прошедшей зимой выпуском самой крупной ЖК-панели, решила попробовать свои силы на ниве миниатюризации, объявив о завершении разработки самого маленького в мире голубого диода, пригодного для использования в считывающей головке оптических приводов Blu-ray и HD DVD следующего поколения.

Представленная новинка, GH04020A4G, обладает диаметром всего 3,3 мм, мощностью в 10 мВт и запасом долговечности в 10 тыс. ч. Отгрузка тестовых образцов начнётся 13 июня 2007 г. Старт массового производства намечен на июль. Цена GH04020A4G составляет $100.

sharp.co.jp