Создание защищенных пользовательских приложений на базе СнК SmartFusion2 компании Microsemi. Часть 6. Криптографическая защита IP-ядер в Libero SoC

В предыдущих статьях [7–11] были рассмотрены вопросы взаимодействия между микроконтроллерной подсистемой (MSS) и массивом ПЛИС (FPGA Fabric) через контроллер FIC, а также технология создания IP-ядер на стандартные интерфейсные шины AHB-Light и APB компании ARM. В предлагаемой и последующих публикациях проанализированы аспекты криптографической защиты IP-ядер с использованием программных средств.

Андрей САМОДЕЛОВ

Защита приложений для СнК SmartFusion2, проектируемых в среде разработки Libero SoC, может осуществляться двумя способами: разделением приложения на выполняемую в микроконтроллерной подсистеме (MSS) бизнес-логику и реализуемую в массиве ПЛИС (FPGA Fabric) функциональную часть. Функциональная часть может выглядеть как один или несколько законченных модулей (IP-ядер). При этом она оказывается отделенной от бизнес-логики либо только с помощью контроллера интерфейса к массиву ПЛИС (FIC), либо с помощью контроллера FIC и процессорного ядра, размещенного в массиве ПЛИС и скрывающего за предоставляемыми микроконтроллерной подсистеме (MSS) функциями API детали реализации целевых функций IP-ядрами. Еще большая степень защиты может быть реализована при использовании технологии шифрования IP-ядер в соответствии со стандартом IEEE 1735, поддерживаемой средой разработки Libero SoC начиная с версии 11.3.

Введение

Остановимся на некоторых предпосылках, приведших к широкому применению IP-ядер при проектировании устройств на ПЛИС.

Современные ПЛИС (FPGA) и СнК ПЛИС (SoC FPGA) предоставляют пользователям большое количество конфигурируемых ресурсов, позволяющих создавать законченные устройства в едином кристалле. В таких сценариях предварительно спроектированные и верифицированные законченные функциональные блоки, или IP-ядра (Intellectual Property, IP), помогают значительно уменьшить время и стоимость разработки. В результате снижается наиболее рискованная и непредсказуемая фаза верификации и отладки системы и, как следствие, общий риск, связанный с проектированием нового устройства.

Использование IP-ядер не только сокращает время разработки, но и предоставляет проверенные и надежные компоненты для повторного размещения во множестве приложений. Применение IP-ядра при разработке в автоматизированной среде проектирования EDA накладывает два противоречивых требования, которые необходимо решить: защита IP-ядра и его совместимость с различными средами проектирования (EDA).

Методология проектирования на основе IP-ядер предусматривает поддержку со стороны расширенных средств разработки, призванных упрощать интеграцию, конфигурирование и тестирование системы, основанной на таких ядрах. Например, при конфигурировании IP-ядер важно убедиться, что конфликтующие определения не смогут пройти через набор инструментов разработки без уведомления об этом проектировщика. Кроме того, большое количество связей IP-ядра с другими объектами может вызвать трудности при ручном управлении подключениями. Таким образом, некоторый упрощенный подход к выполнению подключений, устраняющий общие недочеты в соединениях, способен устранить многие источники ошибок в проекте.

Подход Microsemi к упрощению разработок на основе IP-ядер

Компания Microsemi имеет опыт эффективной реализации основанных на IP-ядрах разработок с двумя основными стратегиями.

Первая стратегия представляет собой поддержку промышленных стандартов, которые повышают эффективность создания, управления, защиты и взаимодействия различных IP-ядер. В основанной на IP-ядрах технологии разработки обычно используется множество данных ядер, полученных из различных источников. Соответственно, важным моментом становится простота соединения этих блоков между собой и совместимость с различными технологиями проектирования. Компания MicroSemi поддерживает следующие промышленные стандарты для IP-ядер:

- стандарт шифрования и управления правами IEEE Std 1735-2014 [5];
- стандартные интерфейсы компании ARM, такие как AHB, AXI и APB3;
- схему IP XACT для XML-спецификации соединений и метаданных для IP-ядра в соответствии со стандартом IEEE Std 1685-2014 [6].

Вторая стратегия — это создание простых в применении автоматизированных схем разработки (design flow) для интеграции в проект и конфигурирования IP-ядер. Без эффективных схем проектирования (development flow) время и усилия будут впустую потрачены на «борьбу» с инструментарием, вместо того чтобы использовать их на простое подключение предопределенных блоков подобных

модулей. Среда визуального проектирования Libero SoC компании Microsemi предоставляет ключевые возможности для упрощения интеграции и конфигурирования IP-ядер:

- Выпадающие меню и другие направленные подходы к определению конфигураций, которые предотвращают дорогостоящее и подверженное ошибкам ручное конфигурирование.
- IP-ядра со стандартными шинами, такими как различные интерфейсные шины компании ARM, можно подключать с помощью интеллектуального инструментария, способного точно выполнять соединения во избежание ошибок интеграции.

Оба подхода предназначены для ускорения цикла проектирования и не только обеспечивают функциональную совместимость между IP-ядрами и инструментами разработки, но и увеличивают эффективность такого набора. Далее каждый из этих подходов будет рассмотрен более подробно.

Шифрование IP-ядер по IEEE 1735 и управление правами доступа для IP-ядер от Microsemi

Поскольку в современных разработках на ПЛИС (FPGA) и СнК ПЛИС (SoC FPGA) возрастает использование IP-ядер, наиболее вероятно, что в новом проекте появится не одно такое ядро от различных поставщиков. В своей деятельности разработчики данных модулей применяют разные методы защиты и управления, поэтому стоимость поддержки различных потоков проектирования значительно возрастает.

К счастью, сегодня доступен стандарт IEEE 1735 шифрования IP-ядер и поставщики таких блоков могут обратиться к этому документу, чтобы зашифровать и, следовательно, защитить свои ценные IP-ядра от инженерного анализа (reverse engineering). Производители ПЛИС и сторонние разработчики инструментов проектирования поддерживают стандарт IEEE 1735, а потому пользователи могут свободно применять IP-ядра, защищенные по стандарту IEEE 1735.

Компания Microsemi приняла стандарт IEEE 1735-2014 и поддерживает процесс разработки зашифрованных IP-ядер для микросхем семейств SmartFusion2, IGLOO2, RTG4 и PolarFire. Совместно с инструментарием Synplify Pro от Synopsys (Synplify Pro ME версия I2013.09MSP1 или более поздняя) и ModelSim (ModelSim версия 10.2c или более поздняя) от Mentor Graphics (поддерживающие стандарт IEEE 1735-2014) среда разработки Libero SoC (v11.3 или выше) позволяет реализовать бесшовный процесс проектирования систем на микросхемах SmartFusion2, IGLOO2, RTG4 и PolarFire с помощью зашифрованных IP-ядер.

Стандарт IEEE 1735-2014 для разработчиков IP-ядер и сред проектирования

Стандарт IEEE 1735-2014 представляет собой схему шифрования, принятую большинством специалистов, и сред проектирования (EDA), чтобы обеспечить совместимость IP-ядер от разных разработчиков, а также различных сред проектирования (EDA). Цель стандарта IEEE 1735-2014 — обслуживание проектировщиков IP-ядер и EDA-сообщества в следующих направлениях:

- для разработчиков IP-ядер: защита безопасности IP-ядер в процессе их создания в различных средах проектирования (EDA);
- для пользователей IP-ядер и разработчиков сред проектирования: гарантировать совместимость IP-ядра между различными EDA-средами.

Алгоритмы шифрования

Среда проектирования Libero SoC поддерживает следующие алгоритмы шифрования:

- DES-CBC;
- Triple DES-CBC;
- AES128-CBC;
- AES256-CBC.

Имеется два основных класса методов шифрования: симметричное и асимметричное шифрование.

Симметричное шифрование

В этой схеме шифрования в качестве ключа для зашифрования данных используется специальная строка. Тот же самый ключ предназначен и для расшифрования данных (рис. 1). Примеры этого типа алгоритмов шифрования включают:

- алгоритмы Data Encryption Standard (DES), такие как DES-CBC;
- алгоритмы Triple DES, или TDES, или TDEA (Triple Data Encryption Algorithm), в которых алгоритм DES используется три раза подряд (с 1, 2 или 3 различными ключами), например Triple DES-CBC;
- алгоритмы Advanced Encryption Standard (AES), в частности AES128-CBC и AES256-CBC.

Для надежной защиты данных ключ шифрования в симметричной схеме должен сохраняться в секрете.

Асимметричное шифрование

В этой схеме шифрования предусмотрено два ключа: один для зашифрования и один для расшифрования. Пользователь генерирует два ключа: закрытый, или секретный, ключ (private key) и затем по нему открытый, или публичный, ключ (public key). После этого пользователь распространяет открытый ключ (public key) среди всех желающих для зашифрования сообщений и оставляет закрытый ключ (private key), чтобы применить его для расшифрования сообщений, зашифрованных на парном к нему открытом ключе (рис. 2).

В качестве примеров алгоритмов асимметричного шифрования можно привести:

- алгоритм генерации общих сеансовых ключей Диффи — Хедмана (Diffie — Hellman, DH);
- алгоритм асимметричного шифрования RSA (Rivest, Shamir, and Adelman).

В асимметричной схеме шифрования в секрете остается только закрытый ключ (private key), а открытый ключ (public key) может быть публично доступен. Открытые ключи распространяются, как правило, в виде сертификатов открытого ключа (X.509), подписанных одним из удостоверяющих центров (УЦ).

Два уровня шифрования

При создании зашифрованного IP-ядра имеется два уровня шифрования. Вначале разработчик IP-ядра применяет сессионный (случайный) ключ для зашифрования содержимого IP-ядра. Это первый уровень шифрования (рис. 1). Затем разработчик IP-ядра использует открытые ключи (Public Key) от разработчиков различных автоматизиро-

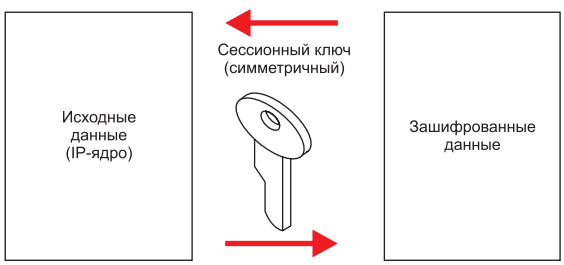

Рис. 1. Зашифрование исходных данных IP-ядра

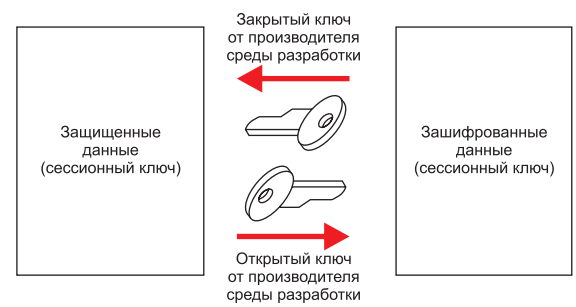

Рис. 2. Зашифрование сессионного ключа

ванных сред проектирования (EDA), чтобы зашифровать сессионный ключ. Это второй уровень шифрования (рис. 2).

Открытые ключи (Public Key) должны передаваться каждым из разработчиков сред проектирования (EDA) разработчику IP-ядер. Для пользователей Libero SoC, которые в своих проектах обращаются к IP-ядрам сторонних производителей, в качестве разработчиков сред проектирования (EDA) выступают:

- Synopsys для инструментария Synplify Pro;
- Mentor Graphics для инструментария ModelSim;
- Microsemi для среды разработки Libero SoC.

Результатом первого уровня шифрования является зашифрованный блок данных. Случайный сессионный ключ (session key), необходимый для симметричного зашифрования блока данных, генерируется скриптом *encryptP1735.pl*.

Итогом второго уровня шифрования является зашифрованный сессионный ключ.

Конверты зашифрования

Конверт зашифрования (encryption envelope) — это начальная часть, или заголовок, IP-ядра в HDL-файле. Поставщики IP-ядер должны подготовить конверты зашифрования (encryption envelope) для всех видов инструментов проектирования (EDA tools), где предполагается использовать IP-ядро. Конверт зашифрования состоит из ключевых слов Pragma (описаны в разделе «Ключевые слова Pragma»), после которых размещается следующая информация:

- версия шифрования (Encryption Version);
- тип кодирования (Encoding Type);
- агент шифрования (Encryption Agent);
- владелец ключа (Key Owner);
- имя ключа (Key Name)
- метод ключа (Key Method).

Ниже приведен пример конверта зашифрования (encryption envelope). Следует отметить, что данный конверт идентифицирует троих поставщиков средств разработки (EDA tool vendor)/владельцев ключей (key owner).

```
module secret (a, b, sum, clk, rstn);
input[7:0]a, b;
input clk, rstn;
output[8:0]sum;
reg[8:0]sum;

/*Encryption Envelope*/
`pragma protect version=1
`pragma protect encoding=(enctype="base64")
`pragma protect author="author-a", author_info="author-a-details"
`pragma protect encrypt_agent="encryptP1735.pl", encrypt_agent_info="Synplify encryption scripts"
`pragma protect key_keyowner="Synplicity", key_keyname="SYNP05_001", key_method="rsa", key_block
`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIMRSA-1",
key_method="rsa", key_block
`pragma protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEYRSA",
key_method="rsa", key_block
`pragma protect data_keyowner="ip-vendor-a", data_keyname="fpga-ip", data_method="aes128-cbc"
/*Ends Encryption Envelope*/

`pragma protect begin
   always @(posedge clk or negedge rstn) begin
      if (!rstn)
sum <= 9'b0;
      else
sum <= a + b;
   end
`pragma protect end
endmodule
```

Конверты расшифрования

Конверт расшифрования (decryption envelope) представляет собой заголовок (преамбулу) в составе зашифрованного IP-ядра. Конверт расшифрования состоит из ключевых слов Pragma (раздел «Ключевые слова Pragma»), после которых располагается следующая информация:

- версия шифрования (Encryption Version);
- тип кодирования (Encoding Type);
- агент шифрования (Encryption Agent);
- владелец ключа (Key Owner);
- имя ключа (Key Name);
- метод ключа (Key Method).

Ниже представлен пример конверта расшифрования (decryption envelope) на языке Verilog.

```
module secret (a, b, sum, clk, rstn);
input[7:0]a, b;
input clk, rstn;
output[8:0]sum;
reg[8:0]sum;

//Decryption Envelope begins
`pragma protect begin_protected
`pragma protect version=1
`pragma protect author="author-a", author_info="author-a-details"
`pragma protect encrypt_agent="encryptP1735.pl", encrypt_agent_info="Synplify encryption scripts"

`pragma protect key_keyowner="Synplicity", key_keyname="SYNP05_001", key_method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=256)
`pragma protect key_block
NfR8W3gmxwh3Bj4QxA+Qi+BhD1CTnQv7KO4UGOOS27KzF4jtejZxAewyFaShFSqRn9tRNx+u7Ivw1m
2BydGyW7MAQx2ePgbrKQbRLaN8XF/iiUFUX0QXnWDZrxtgcVHULOsPXpwd25wNyeWQkTekAslnub
KiFDfNySxaP5W3SboZE0pMLqH+mpZlcvKljlE30uOAQQLjECEBGj1KxMZQ2hhUKLrXz34+9p68tVzbM/
u1TbsXvdPcN23UItAxNPSH5ND75rAviq7ACIVawH87/m2RshSDSVcmz7ndMpSJRQOFe2pdusuHdCFJm
1YaEaCZYfqReV7RjCzbV48d3LPtoA==

`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIMRSA-1",
key_method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=128)
`pragma protect key_block
boN+vsIsOJ/Ihy7BF0MM2ZdaeYl2zoepUP9xdDVnlME3q5lgqZtPjMtPqTQDvwbree7NngmOUGVm
WbggEEW/UWYWajwld641fsggKfu7kcFcMhLLBu0WHUVFvQjRhdiqcBWbEKM39O0SCYTJnhQFPs0B
RZgdCwOPvZ4IEAUqx4U=
`pragma protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEY-RSA", key_method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=960)
`pragma protect key_block
MIID4jANBgkqhkiG9w0BAQEFAAOCA88AMIIDygKCA8EAxvOR7+3o0rtdoggobQ7e3LQ5Bhjfcudaf
ujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrWxballfktti2lBBdU/SDV83mLYKzAqe20/
SaZR5FAZH8cyuUPxYOviHQ/fpqNwUaoU/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8OpN
WDY4fZ4g9vQFBhmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1OZvaxh
5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6NBc4EiyOT2iofDDtqoxdLZPb4
L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtSry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj2145x9vn
XXINiuOIuIj1K/a2djkP+2A3Jvt53z8gv9Jij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp/ZSoVRg
MfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuXIPIODGoHEdFC4feve5uuucMbH
w8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKSF3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQOrQw
0pBvWm0DyUFRTJ53GZAfbEz+1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeUvbmc
uRL8wVMHvXo586qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0dM+
QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2CqbSB58Ykljvoiu
70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY1JekLYeGaSX0WzVcxQcA3flpGL+4Sd
jdRWDYK3wXv6QoQ9YVag78nMIYUECtz+YtRevision 4 7py8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutI
ZZXgP0PhR1smfcEFeUTwe56nDDpBJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqo
viDZ1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GlIMW8KNBNtbn6qFYaMq2uG6AmwTZ
AVfhru0yjnIELj3k3t/OS/YbA6wRFpg0GddNNRAgMBAAE=

`pragma protect data_keyowner="ip-vendor-a", data_keyname="fpga-ip", data_method="aes128-cbc"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=128)
//Decryption Envelope ends
`pragma protect data_block
RgKC7i4hx7zh3MLd50RYrZoCwPWFEyLwISIXDLkpkL6qFgFm1WmZEwFvZjNfQCNUgoSHeIRpxg
9ilXnvMiBjQCiQVvMp32UtfSX625K8+yvJLMPdHQ8G/2qxa6ViHAhBhRcsSUl0XGskRmU3JvNuNf
Ak0IoB1HpFEJ0Vv6vEI5g=

`pragma protect end_protected

endmodule
```

Защита IP-ядер

Для большинства поставщиков IP-ядер одной из важнейших задач является защита собственной интеллектуальной собственности (intellectual property) и упаковка данных модулей таким образом, чтобы они были совместимы с основными средствами разработки (EDA tool) без ущерба для защиты. Шифрование осуществляется на двух уровнях:

- шифрование IP-ядра (IP Core Encryption);
- шифрование ключа шифрования IP-ядра (Data Key Encryption).

Разработчики IP-ядер защищают свои продукты с помощью алгоритмов шифрования, как показано на рис. 3. Для безопасности IP-ядра создается симметричный ключ данных (symmetric data key), зашифровывающий на нем исходный код самого ядра. Затем разработчик получает открытые ключи (public key) от всех поставщиков средств разработки (EDA vendor), в которых планируется использовать защищаемое IP-ядро, и зашифровывает на них симметричный ключ данных (symmetric data key). Затем оба блока данных объединяются в соответствии со стандартом IEEE Std 1735-2014

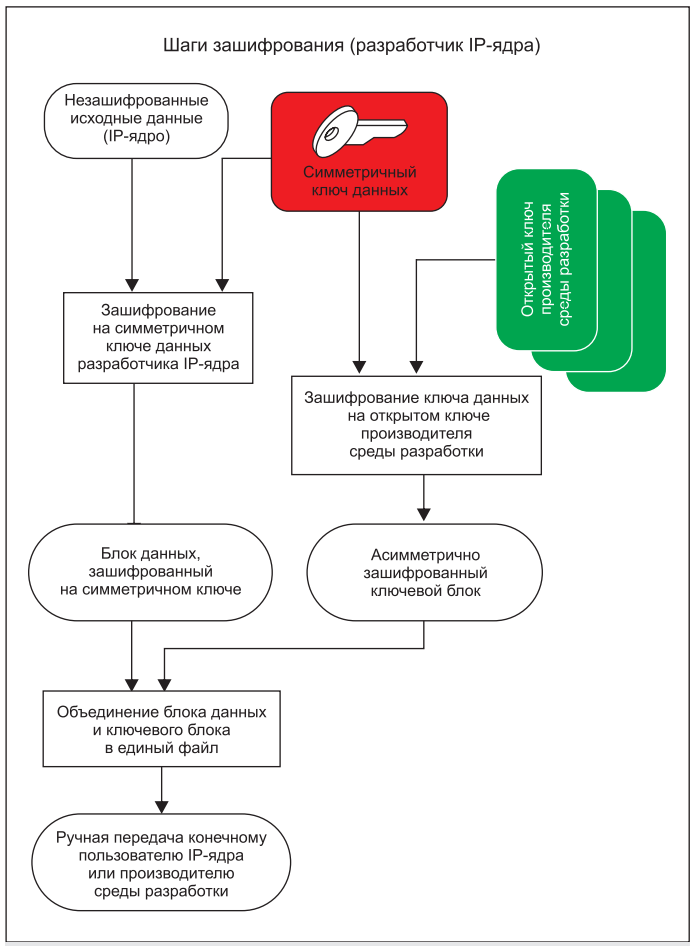

Рис. 3. Последовательность шифрования, используемая разработчиками IP-ядер для их защиты

и передаются конечному пользователю IP-ядра или поставщику средств разработки (CAE vendor). Таким образом, конечный пользователь может применять IP-ядро в своих проектах, но сама разработка оказывается защищенной от инженерного анализа (reverse engineering).

Для того чтобы использовать зашифрованное IP-ядро, конечный пользователь может включить его в свой проект, как показано на рис. 4. Инструментарий для ПЛИС (FPGA tools) от производителя самих ПЛИС (FPGA vendor) или средств разработки (CAE vendor) будет работать в нормальном режиме, но детали реализации IP-ядра не отобразятся, следовательно, будут недоступны для отладки внутри микросхемы (on-chip debugging) или при просмотре синтезированной схемы. Поскольку IP-ядра являются предварительно верифицированными, нет необходимости в отладке их внутренней структуры или просмотра схемы соединений — они просто могут устанавливаться как законченные блоки, в которых наружу выставлены только сигналы интерфейсов и больше ничего.

Защита IP-ядра по стандарту IEEE 1735-2014

Для шифрования IP-ядра по схеме IEEE 1735-2014 следует:

1. Получить открытый ключ (public key) от каждого из производителей средств разработки (EDA tool vendor), с которыми планируется использовать IP-ядро (раздел «Открытые ключи (public key) производителей сред разработки (EDA vendor)»).
2. Добавить конверт зашифрования (encryption envelopes) к коду RTL (как в разделе «Конверты зашифрования»). Убедиться, что в конверт включена информация обо всех необходимых производителях средств разработки (EDA tool vendors).
3. Выполнить скрипт encryptP1735, написанный на языке Perl Script.

Разберем подробнее каждый из этих шагов.

Открытые ключи производителей сред разработки

Получите открытые ключи (public key) от каждого из производителей средств разработки (EDA tool vendor). Объедините полученные открытые ключи (public key) в единый файл хранилища открытых ключей (public keys repository file). Далее приведен пример[1] такого файла.

1 Ключи, приведенные в следующих примерах, являются фиктивными ключами и не будут работать в реальных условиях. Для запроса файла открытого ключа (public key file), который содержит все три открытых ключа (public keys) вместе с файлом скрипта на языке Perl, для зашифрования файлов необходимо отправить запрос на адрес электронной почты soc_marketing@microsemi.com.

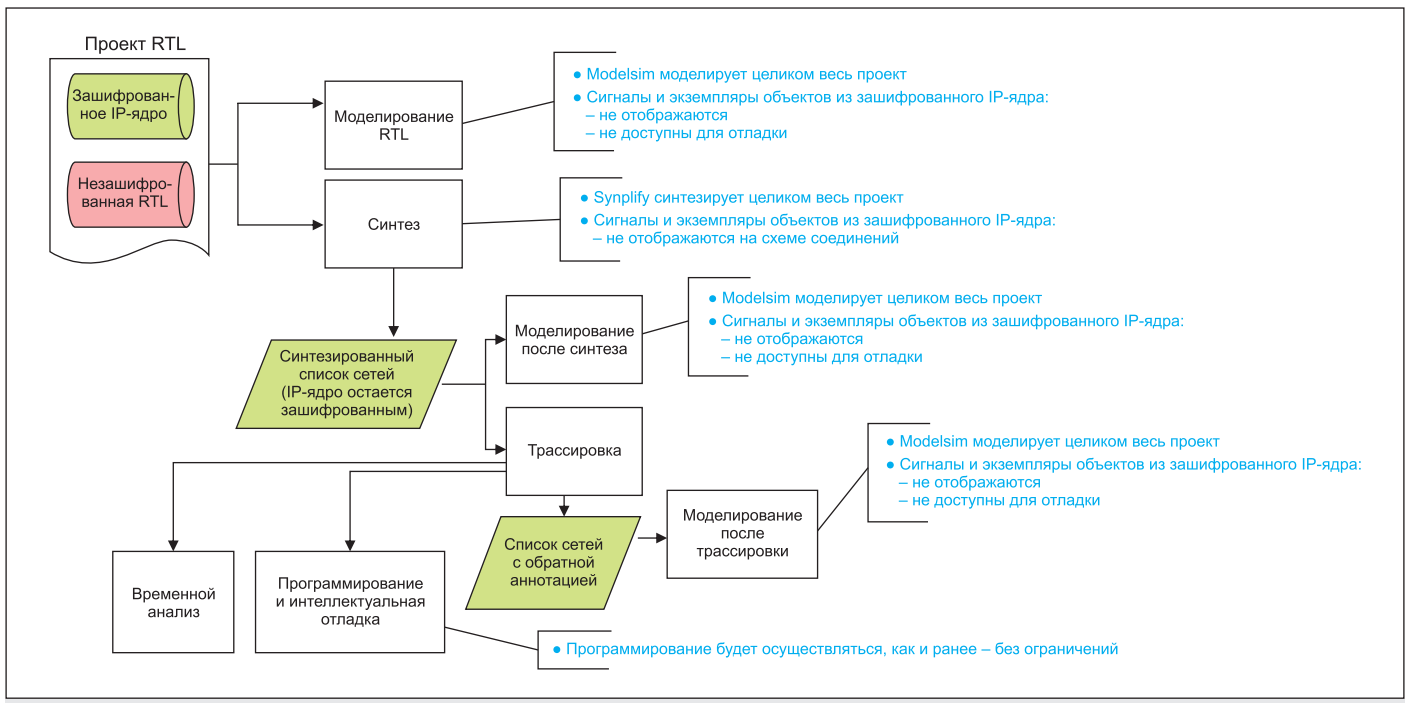

Рис. 4. Последовательность проектирования зашифрованного IP-ядра в среде разработки Libero SoC

```
`pragma protect key_keyowner="Synplicity", key_keyname="SYNP05_001", key_public_key
-----BEGIN PUBLIC KEY-----                    Public Key from Synopsys
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAybsQaMidiCHZyh14wbXnUpP8lK+jJY5oLp
GqDfSW5PMXBVp0WFd1d32onXEpRkwxEJLlK4RgS43d0FG2ZQ1lirdimRKNnUtPxsrJzbMr74MQkwmG/
X7SEe/lEqwK9Uk77cMEncLycI5yX4f/K9Q9WS5nLD+Nh6BL7kwR0vSevfePC1fkOa1uC7b7Mwb1mcqCL
BBRP9/eF0wUIoxVRzjA+pJvORwhYtZEhnwvTblBJsnyneT1LfDi/D5WZoikTP/0KBiP87QHMSuVBydMA7
J7g6sxKB92hx2Dpv1ojds1Y5ywjxFxOAA93nFjmLsJq3i/P0lv5TmtnCYX3Wkryw4BeQIDAQAB
-----END PUBLIC KEY-----
`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIFSIM-RSA-1",
key_public_key
-----BEGIN PUBLIC KEY-----                    Public Key from Mentor Graphics
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCnJfQb+LLzTMX3NRARsv7A8+LV5SgMEJCvIf
9Tif2emi4z0qtp8E+nX7QFzocTlClC6Dcq2qIvEJcpqUgTTD+mJ6grJSJ+R4AxxCgvHYUwoT80Xs0QgRqkr
GYxW1RUnNBcJm4ZULexYz8972Oj6rQ99n5e1kDa/eBcszMJyOkcGQIDAQAB
-----END PUBLIC KEY-----
`pragma protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEYRSA", key_public_key
-----BEGIN PUBLIC KEY-----                    Public Key from Microsemi
MIID4jANBgkqhkiG9w0BAQEFAAOCA88AMIIDygKCA8EAxvOR7+3o0rtdoggobQ7e3LQ5Bhjfcudaf
ujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrWxballfktti2lBBdU/SDV83mLYKzAqe20/
SaZR5FAZH8cyuUPxYOviHQ/fpqNwUaoU/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8Op
NWDY4fZ4g9vQFBhmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1OZv
axh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6NBc4EiyOT2iofDDtqoxdL
ZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtSry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj214
5x9vnXXINiuOIuIj1K/a2djkP+2A3Jvt53z8gv9Iij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp/ZS
oVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuXIPIODGoHEdFC4feve5uuu
cMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKSF3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQ
OrQw0pBvWm0DyUFRTJ53GZAfbEz+1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeU
vbmcuRL8wVMHvXo586qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0
dM+QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2CqbSB58Yk
ljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY1JekLYeGaSX0WzVcxQcA3flpG
L+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Ytpy8dTljdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXg
P0PhR1smfcFFeUTwe56nDDpBJJsyaybQhj76+tz1346gymRTEasBTlklnmu6Xaf YJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GllMW8KNBNtbn6qFYaMq2uG6AmwTZAVfh
ru0yjnIELj3k3t/OS/YbA6wRFpg0GddNNRAgMBAAE=
-----END PUBLIC KEY-----
```

Таблица 1. Ключевые слова Pragma

Ключевое слово Pragma Keyword	Описание
begin	Открывает новый конверт зашифрования (encryption envelope)
end	Закрывает конверт зашифрования (encryption envelope)
begin_protected	Открывает новый конверт расшифрования (decryption envelope)
end_protected	Закрывает конверт расшифрования (decryption envelope)
author	Идентифицирует автора конверта
author_info	Задает дополнительную информацию об авторе конверта
encoding	Задает схему кодирования для зашифрованных данных
data_keyowner	Задает владельца ключа зашифрования данных (data encryption key)
data_method	Задает алгоритм зашифрования данных (data encryption algorithm)
data_keyname	Задает имя ключа зашифрования данных (data encryption key)
data_public_key	Задает открытый ключ (public key) для зашифрования данных (data encryption)
data_decrypt_key	Задает сессионный ключ (data session key)
key_keyowner	Задает владельца (owner) ключа зашифрования ключа (key encryption key)
key_method	Задает алгоритм зашифрования ключа (key encryption algorithm)
key_keyname	Задает имя ключа зашифрования ключа (key encryption key)
key_public_key	Задает открытый ключ (public key) для зашифрования ключа (key encryption)
key_block	Начало закодированного блока ключевых данных
version	Версия зашифрования P1735

Добавление конверта зашифрования к RTL-коду

К кодам RTL необходимо добавить конверт зашифрования (encryption envelopes), как описано в разделе «Конверты зашифрования (encryption envelopes)». В состав конверта зашифрования должны быть включены и идентифицированы как владельцы ключей все средства разработки (EDA tools), которые должны иметь доступ к зашифрованным блокам данных.

Далее приведены примеры IP-ядра на языке Verilog (Листинг 1) и IP-ядра на языке VHDL (Листинг 2), в чей состав входит конверт зашифрования (encryption envelope). В конверте компании Microsemi, Synopsys и Mentor Graphics идентифицированы как владельцы ключей (key owner).

2 Перед выполнением скрипта убедитесь, что на компьютере установлена криптографическая библиотека Open SSL. Библиотека Open SSL необходима для нормальной работы скрипта.

3 Для ОС Windows рекомендуется, чтобы скрипт выполнялся в программном окружении Cygwin Environment on Windows.

```
module secret (a, b, sum, clk, rstn);
input[7:0]a, b;
input clk, rstn;
output[8:0]sum;
reg[8:0]sum;

/*Encryption Envelope*/
`pragma protect version=1
`pragma protect encoding=(enctype="base64")
`pragma protect author="author-a", author_info="author-a-details"
`pragma protect encrypt_agent="encryptP1735.pl", encrypt_agent_info="Synplify encryption scripts"
`pragma protect
key_keyowner="Synplicity", key_keyname="SYNP05_001", key_method="rsa", key_block
`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIFSIM-RSA-1",
key_method="rsa",key_block
`pragma protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEYRSA",
key_method="rsa", key_block
`pragma protect data_keyowner="ip-vendor-a", data_keyname="fpga-ip", data_method="aes128-cbc"
/*Ends Encryption Envelope*/

`pragma protect begin
   always @(posedge clk or negedge rstn) begin
      if (!rstn)
sum <= 9'b0;
      else
sum <= a + b;
   end
`pragma protect end
endmodule
```

Листинг 1. IP-ядро на языке Verilog с конвертом зашифрования (encryption envelope)

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--------------------------------------------------
entity counter is
generic(n: natural :=2);
port(clock:in std_logic;
clear:in std_logic;
count:in std_logic;
Q:out std_logic_vector(n-1 downto 0)
);
end counter;
--------------------------------------------------
architecture behv of counter is
Revision 4 12
signal Pre_Q: std_logic_vector(n-1 downto 0);
begin
`protect version=1
`protect encoding=(enctype="base64")
`protect author="author-a", author_info="author-a-details"
`protect encrypt_agent="encryptP1735.pl", encrypt_agent_info="Synplify encryption scripts"
`protect key_keyowner="Synplicity", key_keyname="SYNP05_001", key_method="rsa", key_block
`protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIM-RSA-1",
key_method="rsa", key_block
`protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEYRSA", key_method="rsa",
key_block
`protect data_keyowner="ip-vendor-a", data_keyname="fpga-ip", data_method="aes128-cbc"
`protect begin

   process(clock, count, clear)
   begin
if clear = '1' then
      Pre_Q <= Pre_Q - Pre_Q;
elsif (clock='1' and clock'event) then
      if count = '1' then
Pre_Q <= Pre_Q + 1;
      end if;
end if;
      end process;

      Q <= Pre_Q;
   `protect end

end behv;

--------------------------------------------------
```

Листинг 2. IP-ядро на языке VHDL с конвертом зашифрования (encryption envelope)

Ключевые слова Pragma

В таблице 1 описаны ключевые слова Pragma (Pragma keywords), используемые в конверте зашифрования (encryption envelope).

Скрипт encryptP1735.pl

Для зашифрования IP-ядра необходимо выполнить[2,3] скрипт ***encryptP1735.pl***. Это скрипт на языке Perl, который компания Synopsys сделала доступным разработчикам для зашифрования их IP-ядер.

```
module secret (a, b, sum, clk, rstn);
input[7:0]a, b;
input clk, rstn;
output[8:0]sum;
reg[8:0]sum

`pragma protect begin_protected
`pragma protect version=1
`pragma protect author="author-a", author_info="author-a-details"
`pragma protect encrypt_agent="encryptP1735.pl", encrypt_agent_info="Synplify encryption scripts"
`pragma protect key_keyowner="Synplicity", key_keyname="SYNP05_001", key_method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=256)
`pragma protect key_block                                    Synopsys Key Block
NfR8W3gmxwh3Bj4QxA+Qi+BhD1CTnQv7KO4UGOOS27KzF4jtejZxAewyFaShFSqRn9tRNx+u7Ivw1m
2BydGyW7MAQx2ePgbrKQbRLaN8XF/iiUFUX0QXnWDZrxtgcVHULOsPXpwd25wNyeWQkTekAslnub
KiFDfNySxaP5W3SboZE0pMLqH+mpZlcvKljlE30uOAQQLjECEBGj1KxMZQ2hhUKLrXz34+9p68tVzbM/
u1TbsXvdPcN23UItAxNPSH5ND75rAviq7ACIVawH87/m2RshSDSVcmz7ndMpSJRQOFe2pdusuHdCFJm
1YaEaCZYfqReV7RjCzbV48d3LPtoA==

`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIMRSA-1",
key_method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=128)
`pragma protect key_block                                    Mentor Graphics Key Block
boN+vsIsOJ/Ihy7BF0MM2ZdaeYl2zoepUP9xdDVnlME3q5lgqZtPjMtPqTQDvwbree7NngmOUGVmWb
ggEEW/UWYWajwld641fsggKfu7kcFcMhLLBu0WHUVFvQjRhdiqcBWbEKM39O0SCYTJnhQFPs0BRZ
gdCwOPvZ4IEAUqx4U=

`pragma protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEY-RSA", key_
method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=960)
`pragma protect key_block                                    Microsemi Key Block
MIID4jANBgkqhkiG9w0BAQEFAAOCA88AMIIDygKCA8EAxvOR7+3o0rtdoggobQ7e3LQ5Bhjfcudaf
ujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrWxballfktti2lBBdU/SDV83mLYKzAqe20/
SaZR5FAZH8cyuUPxYOviHQ/fpqNwUaoU/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8Op
NWDY4fZ4g9vQFBhmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1OZv
axh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6NBc4EiyOT2iofDDtqoxdL
ZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtSry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj214
5x9vnXXINiuOIuIj1K/a2djkP+2A3Jvt53z8gv9Iij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp/ZS
oVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuXIPIODGoHEdFC4feve5uuu
cMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKSF3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQ
OrQw0pBvWm0DyUFRTJ53GZAfbEz+1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeU
vbmcuRL8wVMHvXo586qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0
dM+QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2CqbSB58Yk
ljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY1JekLYeGaSX0WzVcxQcA3flpG
L+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Ytpy8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXg
P0PhR1smfcEFeUTwe56nDDpBJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GllMW8KNBNtbn6qFYaMq2uG6AmwTZAVfh
ru0yjnIELj3k3t/OS/YbA6wRFpg0GddNNRAgMBAAE=

`pragma protect data_keyowner="ip-vendor-a", data_keyname="fpga-ip",data_method="aes128-cbc"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=128)
`pragma protect data_block                                   Data (IP Core) Block
RgKC7i4hx7zh3MLd50RYrZoCwPWFEyLwISIXDLkpkL6qFgFm1WmZEwFvZjNfQCNUgoSHeIRpxg
9ilXnvMiBjQCiQVvMp32UtfSX625K8+yvJLMPdHQ8G/2qxa6ViHAhBhRcsSUl0XGskRmU3JvNuNf
Ak0IoB1HpFEJ0Vv6vEI5g=

`pragma protect end_protected

endmodule
```

Листинг 3. Выходной зашифрованный файл на языке Verilog

В следующем примере команда запускает скрипт ***encryptP1735.pl*** со случайным ключом для зашифрования блока данных:

```
perl./encryptP1735.pl -input secret.v -output secret_enc.v -pk public_keys.txt -v -om encrypted
```

- *input secret.v* задает *secret.v* в качестве входного файла для скрипта; входной файл представляет собой незашифрованный HDL-файл, содержащий один или более конвертов зашифрования (encryption envelope).
- *output secret_enc.v* задает *secret_enc.v* в качестве имени выходного зашифрованого файла после выполнения скрипта зашифрования.
- *pk public_keys.txt* задает *public_keys.txt* в качестве файла хранилища открытых ключей (public keys repository). В этом файле содержатся открытые ключи (public key) для всех целевых сред разработки (EDA tools). Файл, имеющий открытые ключи (public key), должен содержать открытые ключи (public key) для всех производителей средств разработки (EDA vendor), упоминаемых в конверте зашифрования (encryption envelope).
- *om encrypted* задает, каким образом IP-ядро рассматривается при генерации списка сетей (synthesis netlist); режимом по умолчанию является зашифрованное (encrypted) IP-ядро. В этом режиме некоторые ключевые данные, использованные для зашифрования IP-ядра, применяются и в выходном списке сетей (output synthesis netlist).

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

----------------------------------------------------

entity counter is
generic(n: natural :=2);
port(clock:in std_logic;
clear:in std_logic;
count:in std_logic;
Q:out std_logic_vector(n-1 downto 0)
);
end counter;

----------------------------------------------------

architecture behv of counter is

   signal Pre_Q: std_logic_vector(n-1 downto 0);

begin

`protect begin_protected
`protect version=1
`protect author="author-a", author_info="author-a-details"
`protect encrypt_agent="encryptP1735.pl", encrypt_agent_info="Synplify encryption scripts"

`protect key_keyowner="Synplicity", key_keyname="SYNP05_001", key_method="rsa"
`protect encoding=(enctype="base64", line_length=76, bytes=256)
`protect key_block                                           Synopsys Key Block
EzupxwpLZCgcCoy7O42J4O6TjEXDsFHIEXYIfYKVXIsm/8incqBuPuWZ26osQcaegOtanunB7lPos
TFjlZBLLgsDLE/Pl7j8PhcxhySoKy/8TkZClQf7osKMbfeFAMFtIOAqjGT4Ab2F9DdosbC6QkNYFCVL
JSk5nNBeA6bslznTicV416exZcHTV5tJycz2vkFVlRY+BBtcXlhBrxCZSguf9OwHkr0OcufCjKaHE//
kfF1dlJ1jcuidCnJ5rOtG3BDWFQ7f/ClH6H9IkqikEfDy2qGO4Kz1N8OF6sH2MKCj4O5ye7d1aH+QH3Fr
TmoNgnVg9f7McoZ0Ito4Z1qCQ==

`protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIM-RSA-1",
key_method="rsa"
`protect encoding=(enctype="base64", line_length=76, bytes=128)
`protect key_block                                           Mentor Graphics Key Block
Pfy8Cgmz1tqEDSqqkQ+/HYByVzO7Iq9WSlfEgti2EYSXVTU974UChUeOJwTJUA5z24gL1gI2QF3ISYQ
s6NgHG84V+DMh9s3biK9UDHz4KJqa5Xrsx6QwvD6co3rZ09bzNPL8w9uGaPK40DXWTQbY0T6WpD
dIw9u4pvhII/2L5eY=

`protect key_keyowner="Microsemi Corporation", key_keyname="MSC-IP-KEY-RSA", key_method="rsa"
`protect encoding=(enctype="base64", line_length=76, bytes=960)
`protect key_block                                           Microsemi Key Block
MIID4jANBgkqhkiG9w0BAQEFAAOCA88AMIIDygKCA8EAxvOR7+3o0rtdoggobQ7e3LQ5Bhjfcudaf
ujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrWxballfktti2lBBdU/SDV83mLYKzAqe20/
SaZR5FAZH8cyuUPxYOviHQ/fpqNwUaoU/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8Op
NWDY4fZ4g9vQFBhmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1OZv
axh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6NBc4EiyOT2iofDDtqoxdL
ZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtSry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj214
5x9vnXXINiuOIuIj1K/a2djkP+2A3Jvt53z8gv9Iij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp/ZS
oVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuXIPIODGoHEdFC4feve5uuu
cMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKSF3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQ
OrQw0pBvWm0DyUFRTJ53GZAfbEz+1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeU
vbmcuRL8wVMHvXo586qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0
dM+QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2CqbSB58Yk
ljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY1JekLYeGaSX0WzVcxQcA3flpG
L+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Ytpy8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXg
P0PhR1smfcEFeUTwe56nDDpBJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GllMW8KNBNtbn6qFYaMq2uG6AmwTZAVfh
ru0yjnIELj3k3t/OS/YbA6wRFpg0GddNNRAgMBAAE=

`protect data_keyowner="ip-vendor-a", data_keyname="fpga-ip", data_method="aes128-cbc"
`protect encoding=(enctype="base64", line_length=76, bytes=288)
`protect data_block                                          Data (IP core) Block
+m/P6uHpXWo/2MDE8lnrIGmBHe6DSUtiNm7PkpwC+dMErJ9rG4vuwDcoqErHHk4oToYBn4ZavftYDJ
c1W3U7+dxEN3lVcgRsWveZZ0ePIfkkEKhp7cSgfFt5kFfwPEoMHPDhAPeElMr84o0pYEiFdO6VGwOJgU
LvGsFedDKwWnTn6O9FbtKBKuKyl8NG27C89GRtkr4UhguNgVDJKs/O8E9bHlSlyxSh2sD4GnTPLAVC4N
ONi4HjsBhxVGvq04yjbJwOHohjI/WeY26ZqHJN7jqkKrdOhXTi/DRoCY15vjfvALr1kzErv8zjc9qGqBWucH
hmUgwfKzp6p8XfFPHTZlOnsKigVN9Q8Kmu6ZmN3nYadlK8ASo4A7q3v9mAotx6

`protect end_protected

end behv;
```

Листинг 4. Выходной зашифрованный файл на языке VHDL

- *v* задает выполнение скрипта в режиме подробного журналирования (verbose mode).

Выходной зашифрованный файл

Сгенерированный скриптом выходной файл содержит директивы Pragma для расшифрования зашифрованных данных (IP-ядра) и ключевых данных, использованных для зашифрования данных. В листингах 3 и 4 показаны выходные файлы на языках Verilog и VHDL.

Запаковка и объединение зашифрованного IP-ядра и ключевых данных

При выполнении скрипта ***encryptP1735.pl*** происходит объединение и запаковка зашифрованного IP-ядра и зашифрованных ключевых данных (data key) в один файл, который является выходным файлом скрипта. Этот файл готов для распространения среди клиентов.

Работа в среде разработки Libero SoC с зашифрованными IP-ядрами

Среда разработки Libero SoC начиная с версии v11.3 поддерживает использование в потоке проектирования зашифрованных IP-ядер сторонних производителей для семейств микросхем SmartFusion2, IGLOO2, RTG4 и PolarFire (рис. 4).

Работа с зашифрованными IP-ядрами в Libero SoC активирована по умолчанию. Для внедрения зашифрованного IP-ядра в программное обеспечение Libero SoC необходимо выполнить следующие шаги.

1. Установить настройки проекта (project settings) таким образом, чтобы выходной файл синтезировался в формате Verilog netlist.
2. Импортировать зашифрованное IP-ядро как HDL.
3. Выполнить синтез (synthesis) и моделирование (simulation).
4. Выполнить оставшуюся последовательность разработки в Libero SoC design, как показано на рис. 4.

Последовательность разработки с зашифрованными IP-ядрами

В последовательности разработки (design flow) с зашифрованными IP-ядрами должен использоваться синтез (synthesis) с форматом Verilog выходного файла списка сетей (netlist).

При создании нового проекта необходимо изменить настройки проекта (project settings) для поддержки работы с защищенными IP-ядрами в соответствии с IEEE 1735-2014. Зашифрованное IP-ядро можно импортировать как файлы с исходным кодом на языке Verilog или VHDL.

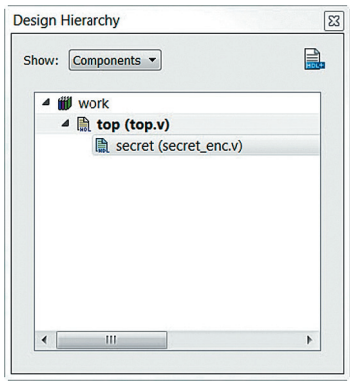

Рис. 6. Окно Design Hierarchy

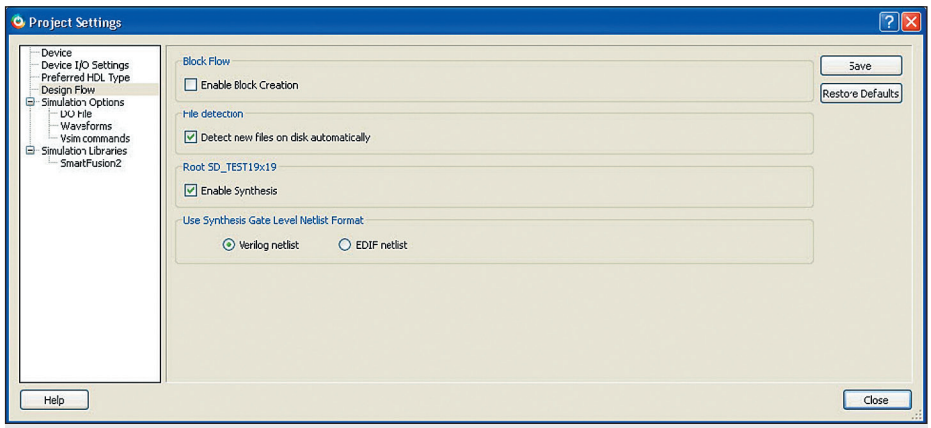

Рис. 5. Настройки формата выходного файла

Схема IEEE 1735-2014 поддерживает только Verilog как формат списка сетей (netlist format); формат EDIF не поддерживается. Поэтому необходимо установить настройки проекта (project settings) в Libero SoC для использования Verilog netlist в качестве формата файла после синтеза (synthesis).

1. Из меню **Project** («Проект») выберите **Project Settings** > **Design Flow**.
2. Выберите **Verilog** в качестве формата Synthesis Gate Level Netlist (рис. 5).
3. Кликните **Save** («Сохранить»), а затем **Close** («Закрыть»).

Импорт зашифрованного IP-ядра как HDL

Импортируйте зашифрованный HDL-файл IP-ядра и незашифрованный HDL-файл как HDL-файлы с исходным кодом (**File** > **Import** > **HDL Source Files**). В окне **Design Hierarchy** отобразится включенный в проект файл (рис. 6).

Рекомендуется, чтобы зашифрованное IP-ядро было представлено в виде одиночного файла. Если в текущий момент IP-ядро организовано в виде иерархии файлов, то предпочтительно, чтобы после зашифрования оно было целиком собрано в единый файл. В настоящее время, если зашифрованное IP-ядро определено во множестве файлов, пользователь должен вручную (на низком уровне) провести файлы через стадии синтеза (synthesis) и моделирования (RTL simulation). Это можно сделать, используя настройку Organize input file инструмента Synthesis/Simulation, как показано ниже. Более подробную информацию об организации файлов с исходным кодом можно получить в разделе Organize Source file справочной системы (пункт меню **Help**) среды разработки Libero SoC. На рис. 7 показана организация входных файлов с исходными кодами для синтеза (synthesis).

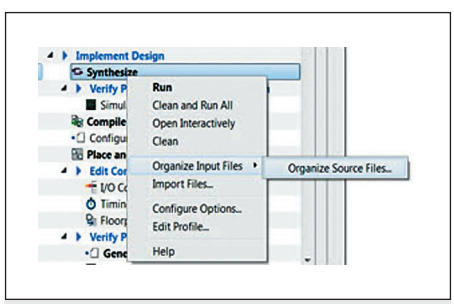

Рис. 7. Организация входных файлов с исходным кодом для синтеза (synthesis)

Поддержка инструмента SmartDesign

Инструмент SmartDesign является визуальным, основанным на блочном принципе инструментом разработки, конфигурирования и соединения между собой IP-ядер от Microsemi, сгенерированных пользователем IP-ядер и HDL-модулей обычной/связующей логики. В SmartDesign можно также создавать экземпляры объекта (instantiate) зашифрованного IP-ядра вместе с другими незашифрованными IP-ядрами. Дополнительную информацию о SmartDesign можно получить в справочной системе Libero SoC.

Выполнение синтеза

После выполнения синтеза в окнах **RTL** и **Technology views** видны только интерфейсные сигналы (входные и выходные порты) защищенного IP-ядра (рис. 8, 9). Сигналы и имена внутренних объектов зашифрованного IP-ядра не видны.

В окнах **RTL** и **Technology views** для блоков проекта, зашифрованных по стандарту IEEE 1735-2014, команды **push** и **pop** запрещены. Невозможно войти (push) в зашифрованный IP-блок «U0» для просмотра внутренних сигналов, сетей или экземпляров других объектов (instance) внутри зашифрованного блока.

Выполнение моделирования в ModelSim

Инструмент ModelSim полностью моделирует[4] весь проект для стадий предварительного синтеза (pre-synthesis), после за-

4 Моделирование поддерживается как для языка Verilog, так и для языка VHDL.

вершения синтеза (post-synthesis) и после завершения трассировки (post-layout). Однако сигналы и экземпляры объектов (instance) внутри зашифрованного IP-ядра не отображаются и недоступны для отладки.

Величины внутренних сигналов не отображаются в окне эпюр напряжений в контрольных точках (waveform window); отображаются только сигналы интерфейсов на границе экземпляра объекта (instance) «U0» зашифрованного IP-ядра (рис. 10).

Среда разработки Libero SoC и зашифрованные IP-ядра

Среда разработки Libero SoC обрабатывает проекты с зашифрованными IP-ядрами целиком через весь проект (design) без ущерба для содержимого самого зашифрованного ядра. Шифрование IP-ядра защищено инструментами синтеза (synthesis) и моделирования (simulation), как упоминалось в предыдущих разделах.

Все экспортированные из Libero SoC списки сетей (netlist) имеют зашифрованные компоненты, связанные с реализацией IP-ядер. В их состав входят:

- обратно аннотированный список сетей (Back annotated netlist) после размещения и трассировки (Place and Route): **_ba.v* или **_ba.vhd*;
- экспортированный после компиляции список сетей (netlist): **.v* или **.vhd*.

На всех этапах проектирования компания Microsemi придерживается принципов шифрования в стандарте IEEE 1735-2014.

Пример

Далее рассмотрим, как в среде разработки Libero SoC реализован зашифрованный модуль с использованием технологии защищенных IP-ядер (Secure IP Flow). Пример содержит файлы, описанные в таблице 2.

Рассмотрим подробнее процедуру создания зашифрованного IP-ядра в среде разработки Libero SoC.

Зашифрование модуля IP-ядра

Зашифрование выполняется с помощью скрипта ***encryptP1735.pl***, который реализует стандарт IEEE 1735-2014 зашифрования модулей IP-ядер (в рассматриваемом примере secret.v). Сегменты кода, которые необходимо зашифровать, включаются в конверты

Таблица 2.
Описание файлов демонстрационного проекта

Имя файла	Описание
Secret.v	Простейший незашифрованный модуль на языке Verilog. Этот модуль содержит конверт зашифрования (encryption envelopes), как было рассмотрено выше
Secret_enc.v	Зашифрованная версия модуля secret.v, которая была зашифрована путем выполнения скрипта encryptP1735.pl для модуля secret.v
Top.v	Модуль верхнего уровня, содержащий экземпляр объекта зашифрованного модуля secret_enc.v
Tb.v	Набор тестов для модуля Top.v
Public_keys.txt	Текстовый файл ключевого хранилища (vault), содержащий открытые ключи (public key) от Synopsys, Mentor и Microsemi

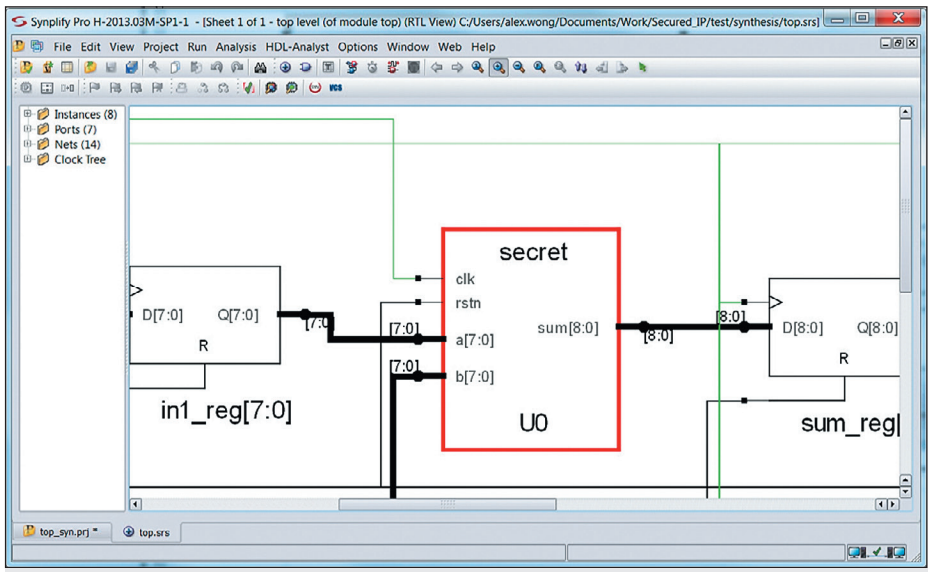

Рис. 8. Вид RTL в SynplifyPro

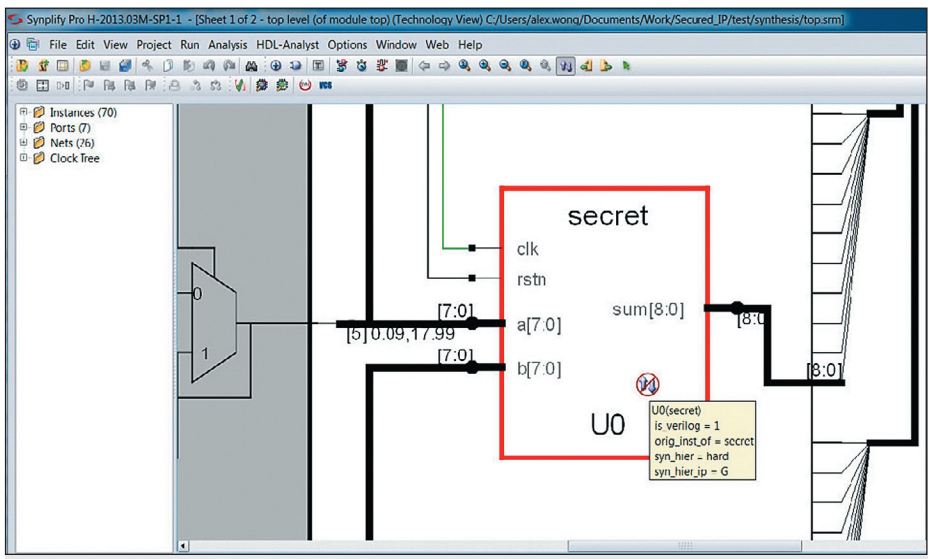

Рис. 9. Technology View в SynplifyPro

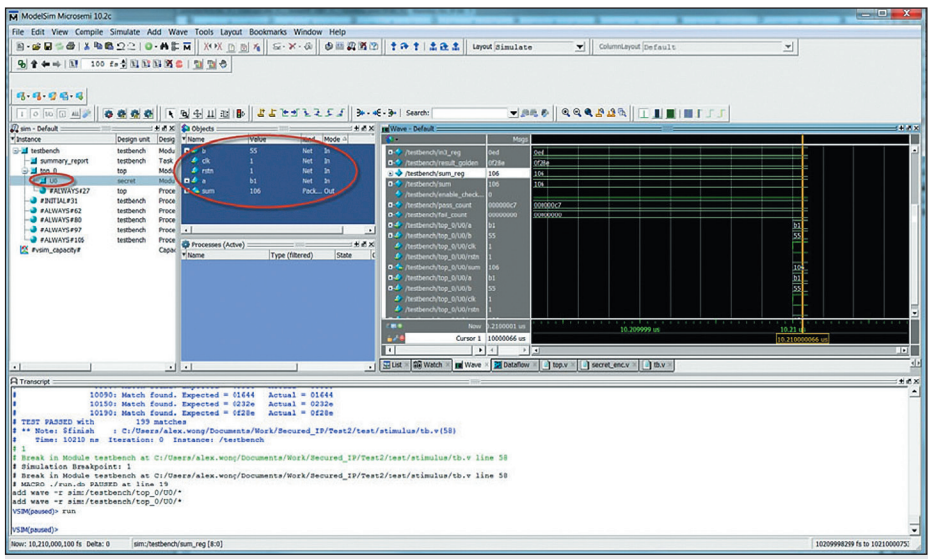

Рис. 10. Modelsim Simulation of Encrypted IP Core

зашифрования (encryption envelopes), как описано в разделе «Добавление конверта зашифрования (encryption envelope) в RTL». Все открытые ключи (public key) от производителей средств разработки, поддерживающих этот стандарт, хранятся в едином файле *Public_Keys.txt*.

Выполним скрипт ***encryptP1735.pl***, используя *secret.v* в качестве входного файла и *secret_enc.v* в качестве выходного файла.

На рис. 11 показан пример выходного кода после выполнения скрипта ***encryptP1735.pl*** над входным файлом модуля *secret.v*. Содержимое выходного файла похоже на код, приведенный ранее в листинге 3.

Отметим, что зашифрованный выходной файл с исходным кодом IP-ядра имеет блок key_blocks, соответствующий всем производителям сред разработки, и блоки Data_blocks с зашифрованной информацией.

Более подробную информацию о параметрах скрипта можно найти в разделе «Скрипт encryptP1735.pl» выше. Скрипт может выполняться как в ОС Linux, так и в ОС Windows с установленными библиотеками openSSL и интерпретатором языка Perl Installed.

Импорт зашифрованного IP-ядра в Libero SoC

Импорт зашифрованного модуля происходит тем же путем, что и импорт любого HDL-файла в проект Libero SoC.

Создайте в Libero SoC проект для кристалла одного из семейств SmartFusion2/IGLOO2/RTG4/PolarFire. Импортируйте файлы *Top.v* и *Secret_enc.v* (меню **File > import > HDL Source Files**) в проект Libero SoC. Кроме того, импортируйте файл *file tb.v* с соответствующим набором тестов (меню **File > Import > HDL Stimulus Files**). После импортирования эти файлы должны появиться в дереве проекта в окнах **Design hierarchy** и **Stimulus hierarchy**, как показано на рис. 6 и 12.

В модуле верхнего уровня или в Smart Design может присутствовать несколько экземпляров объекта зашифрованного модуля. Выберите *top.v* в качестве корневого модуля (Root module), кликнув правой кнопкой мыши и во всплывающем меню указав Set as Root. В настройках проекта Libero SoC (меню **Project > Project Settings > Design Flow**) измените формат списка сетей после синтеза (Synthesis netlist format) на Verilog Netlist, как показано на рис 5.

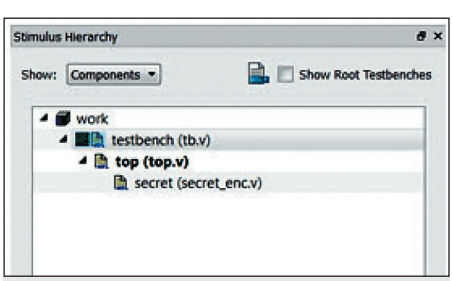

Рис. 12. Окно Stimulus Hierarchy

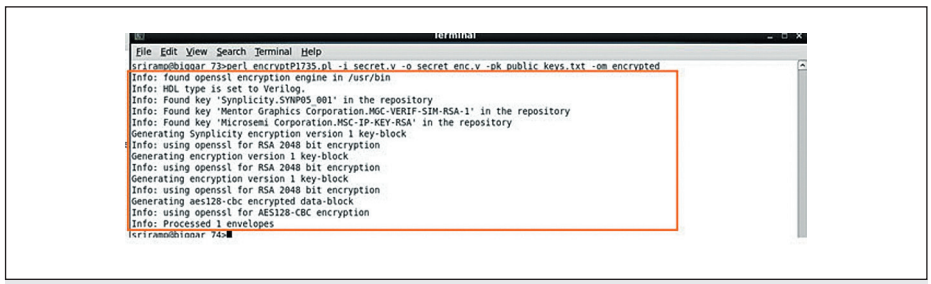

Рис. 11. Выходные данные скрипта EncryptP1735.pl

Синтез

Инструмент синтеза (synthesis tool) Synplify Pro расшифровывает зашифрованное содержимое, используя блок данных Synopsys Key Block, присутствующий в зашифрованном модуле *secret_enc.v*. После выполнения синтеза в окнах **RTL view** и **Technology view** будут видимы только сигналы интерфейсов (входные и выходные порты) защищенного IP-ядра. Подробности можно найти в разделе «Выполнение синтеза». Файл списка сетей (netlist) на языке Verilog (файл с расширением *.vm*), полученный в результате синтеза, не отображает внутренние экземпляры объектов (instance) зашифрованного модуля, и эта информация снова перешифровывается инструментами синтеза (Synthesis tool).

Моделирование

Инструментарий для моделирования (simulation tool) ModelSim расшифровывает защищенное содержимое, используя блок ModelSim Key Block, присутствующий в зашифрованном модуле *secret_enc.v*. Инструментарий ModelSim моделирует целиком весь проект для стадий предварительного синтеза (pre-synthesis), после завершения синтеза (post-synthesis) и после завершения трассировки (post-layout). Однако сигналы и экземпляры объектов (instance) внутри зашифрованного IP-ядра не отображаются и недоступны для отладки. Подробности можно найти в разделе «Выполнение моделирования в ModelSim (ModelSim Simulation)».

Компиляция и размещение

Оставшийся инструментарий, участвующий в последовательности проектирования (design flow) в среде разработки Libero SoC, расшифровывает защищенное содержимое, используя блок Microsemi Key Block, находящийся в зашифрованном модуле *secret_enc.v*.

После завершения синтеза (synthesis) компилятор (compile tool) берет зашифрованные *.vm*-файлы списка сетей (netlist) в качестве входных файлов для дальнейшей обработки инструментарием трассировки (layout tool). В процессе выполнения выходные данные этого инструментария аналогичны таковым для стандартной последовательности проектирования (regular flow).

Последовательность обработки Constraints flow, включая Timing Constraints и Floorplan Constraints, не поддерживается для экземпляров объектов (instance) зашифрованных блоков.

В приведенном выше примере последовательность обработки Constraint flow не поддерживается для модуля *secret_enc.v*. Однако пользователи могут задать ограничения на интерфейс зашифрованного модуля.

Генерация файлов обратной аннотации

После завершения операции трассировки (layout) пользователь может сгенерировать файлы обратной аннотации (back annotated files) для моделирования (post-layout simulation). Сгенерированные файлы **_ba.v* или **_ba.vhd* показывают внутреннюю информацию из модуля of secure_enc.v как зашифрованную.

Этот файл содержит блок Key_Block от Mentor Graphics, предназначенный для расшифрования защищенного содержимого модуля на этапе моделирования после трассировки (post-layout simulations).

Генерация данных для программирования

После завершения моделирования стадий Layout Simulation и Post-Layout Simulation пользователь может приступать к генерации файла для программирования микросхемы (programming file).

На этом завершается описание последовательности работы с зашифрованными IP-ядрами в среде разработки Libero SoC.

Заключение

Применение IP-ядер для ПЛИС (FPGA) и СнК ПЛИС (SoC FPGA) может значительно уменьшить время и стоимость разработки. В то же время процедура распространения IP-ядер должна содержать определенные меры по защите интеллектуальной собственности. Принятый в 2014 году стандарт IEEE Std 1735 позволяет осуществлять криптографическую защиту IP-ядер на уровне не только исходного кода, но и всех промежуточных файлов, возникающих в процессе проектирования. В статье изложены основные принципы, использованные компанией MicroSemi в среде проектирования Libero SoC

по программной криптографической защите IP-ядер для микросхем семейств SmartFusion2, IGLOO2, RTG4 и PolarFire. В следующей статье будет рассказано о решении по защищенному программированию микросхем SmartFusion2/IGLOO2 с помощью аппаратных модулей криптографической защиты (Hardware Security Modules, HSM).

Литература

1. Microsemi IP Cores Accelerate the Development Cycle and Lower Development Costs. **www.microsemi.com/document-portal/doc_view/134435-microsemi-ipcores-accelerate**
2. UG0533. Libero SoC Secure IP Flow for IP Vendors and Libero SoC Users. User Guide **www.microsemi.com/document-portal/doc_download/133573-libero-soc-secure-ip-flow-user-guide**
3. Libero SoC Secure IP User's Guide. **www.microsemi.com/document-portal/doc_download/133573-libero-soc-secure-ip-flow-user-guide**
4. IP Cores. **www.microsemi.com/products/fpga-soc/design-resources/ip-cores**
5. IEEE Std 1735-2014 — IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP). **www.standards.ieee.org/findstds/standard/1735-2014.html**
6. IEEE Std 1685-2014 — IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows. **www.standards.ieee.org/getieee/1685/download/1685–2014.pdf**
7. Самоделов А. Создание защищенных пользовательских приложений на базе SmartFusion2 SoC FPGA компании Microsemi. Часть 1. Контроллер интерфейса между микроконтроллером и ПЛИС // Компоненты и технологии. 2017. № 8.
8. Самоделов А. Создание защищенных пользовательских приложений на базе SmartFusion2 SoC FPGA компании Microsemi. Часть 2. Контроллер прерываний интерфейса между микроконтроллером и ПЛИС // Компоненты и технологии. 2017. № 9.
9. Самоделов А. Создание защищенных пользовательских приложений на базе SmartFusion2 SoC FPGA компании Microsemi. Часть 3. Подключение пользовательской логики к микроконтроллерной подсистеме SmartFusion2 // Компоненты и технологии. 2017. № 10.
10. Самоделов А. Создание защищенных пользовательских приложений на базе SmartFusion2 SoC FPGA компании Microsemi. Часть 4 // Компоненты и технологии. 2017. № 11.
11. Самоделов А. Создание защищенных пользовательских приложений на базе SmartFusion2 SoC FPGA компании Microsemi. Часть 5 // Компоненты и технологии. 2017. № 12.

Часто задаваемые вопросы

Далее приведен краткий список наиболее часто задаваемых вопросов о технологии работы с защищенными IP-ядрами (Secure IP flow) в среде разработки Libero SoC.

1. **Поддерживается ли моделирование VHDL-файлов, поскольку используется список сетей на языке Verilog?**
   Технология Secure IP flow поддерживает как VHDL, так Verilog. Смешанный режим моделирования не требуется, если проект и набор тестов (test bench) написаны на VHDL. Список сетей (netlist) Verilog нужен только для прохождения проекта в Libero SoC — от стадии синтеза (synthesis) до стадии компиляции (compile). Стадия post-synthesis и следующие стадии моделирования используют список сетей (netlist) VHDL, если при создании проекта (Project Creation) для параметра Preferred input HDL type выбрано значение VHDL.
2. **Поддерживает ли Block Flow от Microsemi в технологии Secure IP Flow?**
   Нет. Block Flow не поддерживается для Encrypt IP flow и Secure IP flow.
3. **Поддерживаются ли параметры/обобщения (generic)?**
   Да. Технология Secure IP flow работает с зашифрованными IP-ядрами (Encrypted IP) с параметрами или общими определениями (generic definitions). Однако то, что параметры/обобщения (generics) верхнего уровня и порты остаются незашифрованными, делает RTL удобным для интеграции. Подробности можно найти в рассмотренных ранее примерах на VHDL, в которых имеются общие определения (generic definitions).
4. **Какие версии интерпретатора языка Perl и библиотеки OpenSSL необходимы для выполнения скрипта encryptP1735.pl?**
   Для выполнения скрипта можно использовать любые версии OpenSSL/Perl.
5. **Как установить OpenSSL?**
   Библиотеки OpenSSL являются программным обеспечением с открытыми исходными кодами (Open-Source Software). Во многих вариантах установки ОС Linux имеется предустановленная библиотека OpenSSL. Большинство вариантов установщиков Cygwin на ОС Windows также содержат в своем составе пакет OpenSSL, который можно установить. Для ОС Windows необходимо установить OpenSSL.exe. Необходимую версию OpenSSL можно загрузить со страницы openssl-for-windows [**https://code.google.com/archive/p/openssl-for-windows/downloads**]. После установки OpenSSL на ОС Windows для работы скрипта EncryptP1735.pl следует установить переменную окружения (Environment Variable) PATH в <openssl_installation_dir>\bin.
6. **Можно ли импортировать зашифрованное IP-ядро на языке Verilog в проект на языке VHDL и наоборот?**
   Да. Можно импортировать зашифрованный модуль на языке Verilog (или VHDL) в проект на языке VHDL (или Verilog).