by Andrew Dauman

It is hard to test bought-in cores for chips without being able to access simulation models. But those models need to be protected and, up to now, those systems have been tough to use.

# Opening

# up the black box

Since semiconductor intellectual property (IP) first emerged on the integrated circuit (IC) design scene over a decade ago, it has offered the promise of shaving months off of development time while allowing for the inclusion of many more functions in a design.

In reality, however, it has fallen short of fulfilling its promise. To be sure, very compelling IP has been introduced into the marketplace. Where things have failed to measure up is in how effectively that IP is integrated into a design flow. Designers have simply been unable to consistently use an IP element of choice with their existing design tools.

At the heart of the problem is the need by IP providers to protect their development investment. Few IC designers can afford to pay the often high price for access to IP source code, so IP suppliers provide significantly less costly 'black box', or encrypted, versions for use with specific design tools. The problem with this is not only that the IP vendor is saddled with the cumbersome task of maintaining multiple IP core versions for various tool sets, but, more importantly, that it is an incomplete solution.

Most tool sets in use today consist of a mixture of products from various sources. Because of this, it is often difficult for IC designers to use encrypted IP throughout their flow. A so-called 'black box flow' can impair designers' ability to debug and analyse designs and may impair the ability of design tools to deliver the best result.

To address the need for a means to protect IP from piracy while making it easier for the IP supplier to deploy and the IC designer to use, Synplicity has developed and introduced an open IP encryption/decryption methodology. It takes advantage of features in the most recent release of Verilog and commonly used encryption mechanisms in a new usage model that could become an industry standard.

Until recently, an interoperable and accessible IP encryption methodology was not feasible because critical elements in the methodology were lacking or had not been recognised as such. But key developments in encryption methods and design-automation standards have broken down these technological barriers.

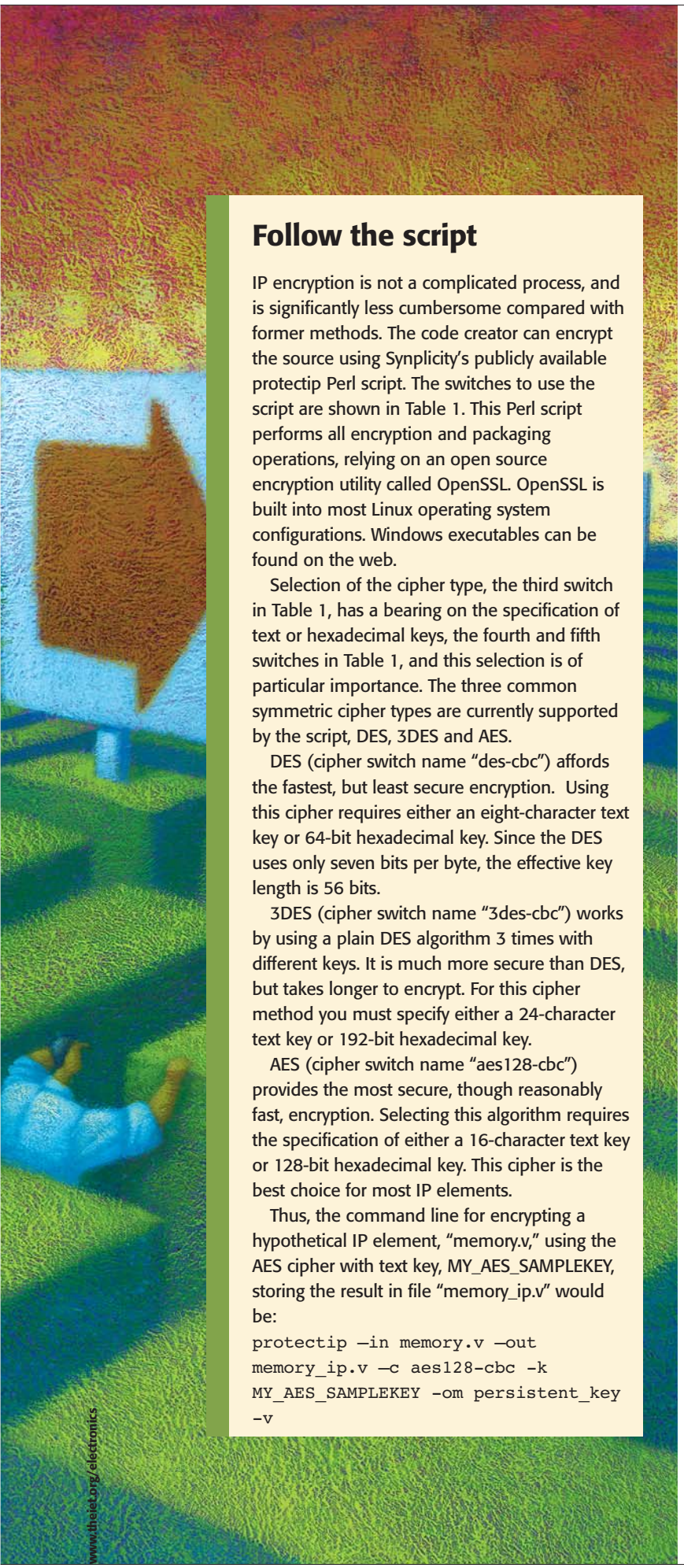
One critical element contributing to this confluence is in the discipline of IP cryptography, or the means by which IP source code is converted into a secure form that is readable by authorised tools. Our approach uses the strengths of the two common cryptographic methods, and overcomes the significant weaknesses of each method when deployed independently.

The two traditional classes of encryption methods are symmetric and asymmetric. Symmetric algorithms employ a secret number, called a 'key', which enables the recipient to decrypt and use an IP element. The same key is used to encrypt and decrypt the code. Symmetric encryption and decryption are generally fast and inexpensive processes. However, security is a concern because the user must have access to the key, and the risk that a key could be compromised or leaked by one of the tens of thousands of users worldwide is one that most IP vendors are unwilling to take.

## KEY CHANGES

IP vendors have attempted to abate this risk by creating a unique key for each design-tool vendor, which then creates unique keys for each of its silicon vendors. But this approach quickly becomes unwieldy, particularly when compounded with new software releases and IP key changes. Not only does this greatly complicate interoperability between design tools but it fails to eliminate the security risk.

Asymmetric algorithms differ in form from their symmetric counterparts in that they employ two different keys. A so-called public key is used for encryption only and is made available to those who need to encrypt the protected information. The two keys are related to each other in that the public key is the product of two very large prime numbers, and the private key is one of those two numbers. To asymmetrically encrypt IP, the design-tool vendor creates public and private keys, supplies a public key to the IP provider for IP encryption, and encrypts and embeds the private key in its tools so that they can decrypt incoming IP. Such algorithms are not only harder to break than symmetric ciphers, but they also eliminate the risk of a leak because the key for decryption is not distributed to users. →



## Follow the script

IP encryption is not a complicated process, and is significantly less cumbersome compared with former methods. The code creator can encrypt the source using Synplicity's publicly available protectip Perl script. The switches to use the script are shown in Table 1. This Perl script performs all encryption and packaging operations, relying on an open source encryption utility called OpenSSL. OpenSSL is built into most Linux operating system configurations. Windows executables can be found on the web.

Selection of the cipher type, the third switch in Table 1, has a bearing on the specification of text or hexadecimal keys, the fourth and fifth switches in Table 1, and this selection is of particular importance. The three common symmetric cipher types are currently supported by the script, DES, 3DES and AES.

DES (cipher switch name "des-cbc") affords the fastest, but least secure encryption. Using this cipher requires either an eight-character text key or 64-bit hexadecimal key. Since the DES uses only seven bits per byte, the effective key length is 56 bits.

3DES (cipher switch name "3des-cbc") works by using a plain DES algorithm 3 times with different keys. It is much more secure than DES, but takes longer to encrypt. For this cipher method you must specify either a 24-character text key or 192-bit hexadecimal key.

AES (cipher switch name "aes128-cbc") provides the most secure, though reasonably fast, encryption. Selecting this algorithm requires the specification of either a 16-character text key or 128-bit hexadecimal key. This cipher is the best choice for most IP elements.

Thus, the command line for encrypting a hypothetical IP element, "memory.v," using the AES cipher with text key, MY_AES_SAMPLEKEY, storing the result in file "memory_ip.v" would be:

```
protectip –in memory.v –out
memory_ip.v –c aes128-cbc -k
MY_AES_SAMPLEKEY -om persistent_key
-v
```



The drawbacks associated with asymmetric encryption, however, are significant. The IP vendor must create encrypted IP versions for each design-tool vendor, creating logistical complications not dissimilar to those of the multi-key symmetric approach. More significantly, decrypting a large IP block using compute-intensive asymmetric algorithms can take hours, making this approach impractical for the user.

Instead, a hybrid encryption approach that exploits the strengths of both symmetric and asymmetric methods delivers the high security and productivity demanded by the various constituencies.

In a hybrid IP cryptosystem, shown in fig 1, the IP vendor generates a symmetric key and uses it to rapidly encrypt the IP. The IP vendor then encrypts this symmetric key itself using an asymmetric algorithm and the design-tool vendor's public key. The process is repeated for each design-tool vendor.

Because the key does not represent a lot of data, even the asymmetric encryption process for multiple vendors is very fast. The IP provider combines the encrypted IP and encrypted keys for all design-tool vendors into a single file and delivers the file to all its customers. The design tools that support this methodology are equipped with sufficient information embedded in the IP envelope to fully use the IP when the user integrates it into a design.

### PRIVATE COPIES

The design tool, thus endowed with the vendor's private key, symmetrically decrypts the keys and IP, integrates the IP with unencrypted elements of the design, processes the design and then encrypts the proprietary portions of the output using the same keys. Each design tool uses its own private key to access the IP vendor's unique symmetric key, and all decryption, data manipulation and encryption activities take place inside the tools. The unencrypted IP is never accessible by the end user. Files are decrypted in memory only and never stored on disk.

In addition to hybrid encryption, another key development that contributes to the confluence of an open IP-based design flow is encryption-embedding mechanisms that are now appearing in standard design languages and data formats. The recently published Verilog standard IEEE 1364-2005 and forthcoming VHDL 2006 standard both are compatible with a hybrid encryption scheme. These standards also allow the IP vendor to specify the type of encryption algorithm used, as well as the flexibility to partially decrypt an element in certain stages of the design process where more visibility is required, such as verification and debugging.

As the keys, encryption and decryption are all handled within the design tools, and are not touched by the end user, the prescribed flow requires no action on the part of the IC

| | |
|---|---|
| -in *input_file_name* | name of the Verilog source to be encrypted |
| -out *output_file_name* | name of the encrypted file |
| -c *symmetric_cipher_name* | currently supported ciphers are “des-cbc” “3des-cbc” “aes128-cbc” |
| -k *text_key* | secret encryption key in text format (do not use together with –kx) |
| -kx *hex_key* | secret encryption key in hexadecimal format (do not use together with –k) |
| -a *string* | string identifying author of the encrypted code (optional) |
| -dkn *string* | string identifying data key name (optional) |
| -dko *string* | string identifying data key owner (optional) |
| -om *output_method* | specification of Synplify output: “none” or “persistent_key” (see Synplify documentation for details) |
| -v | verbose mode – provides detailed report of the encryption progress |

*Table 1: Perl “protectip” script switches for IP encryption*

> **“The tasks at hand for IP providers are to create a symmetric private key and provide access to the design tools by encrypting its symmetric key with each tool vendors’ public key”**

designer. The tasks at hand for IP providers are to create a symmetric private key and provide access to the design tools by encrypting its symmetric key with each tool vendors’ public key. With a single file per IP core, this task is greatly simplified compared the approaches in common use today. The design-tool vendor supplies a public key – half of the public/private key pair – for use by anyone who wants their IP to work with the tool vendors.

The capabilities of this open IP encryption methodology have been tested by a number of early adopters: Synplicity, Aldec and Lattice Semiconductor. Broadening usage is the next important step in enabling this capability to fully realise its potential for unburdening IP and EDA vendors, and empowering IC designers. To facilitate this broadening, Synplicity is actively involved in selecting a standards organisation to continue the deployment of the technology. With the status and support of a standards organisation behind it, the methodology can more rapidly develop a large constellation of EDA suppliers and IP vendors, and serve the user community well.

With this IP encryption flow, the original promise of semiconductor IP can, for the first time, be realised – for the IP provider, for the design-tool vendor and for the IC designer. IP vendors can easily deliver functional components that are interoperable with multiple design-tool sets, without security concerns and without

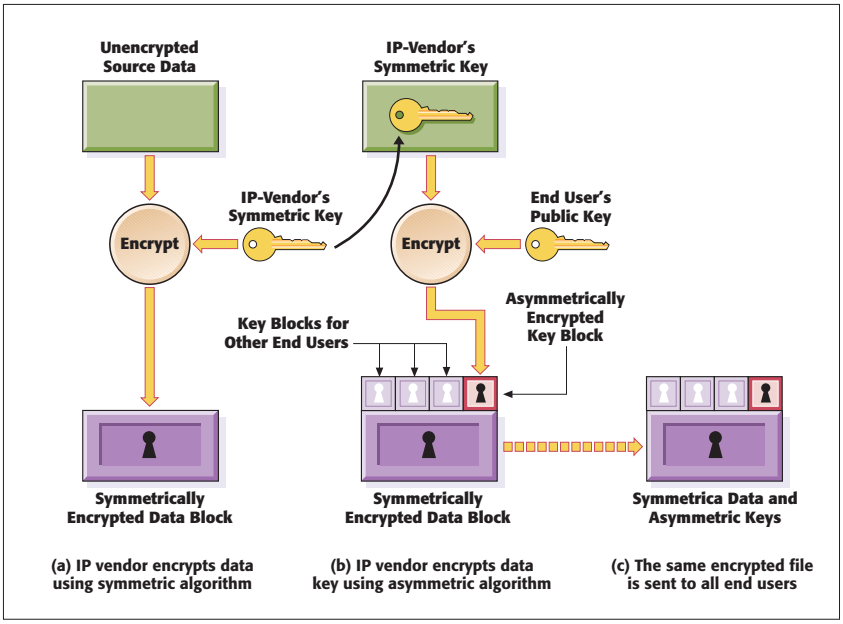



*Fig 1: Hybrid IP encryption scheme*

complicated and cumbersome encryption, management and delivery processes. Similarly, the design-tool vendors can support a wide variety of IP providers without excessive management overhead and tools can achieve maximum functionality instead of being impaired by proprietary IP security schemes.

Finally, the IC designer for whom IP protection has heretofore been a nuisance that inhibits design freedom and tool choice is liberated to select the best IP element for his needs regardless of his design toolset. Moreover, being able to use a single version of the element across the flow greatly reduces management overhead. As more and more vendors adopt this method, IC designers will be increasingly positioned to realise the benefits of IP integration. ■

**Andrew Dauman is senior vice president of worldwide engineering at Synplicity**