UG0533
User Guide
Libero SoC Secure IP Flow
for IP Vendors and Libero SoC Users

Table of Contents

Introduction

Microsemi adopted IEEE 1735-2014 and supports an encrypted IP design flow for the SmartFusion2, IGLOO2, RTG4, and PolarFire silicon families.

See "Securing Your IP Core" on page 8 to get started securing your IP core.

See "Running Libero SoC with Encrypted IP" on page 17 for information on running Libero SoC with encrypted IP.

Together with its OEM tools, Synplify Pro from Synopsys (Synplify Pro ME I2013.09MSP1 or later) and ModelSim (ModelSim 10.2c or later) from Mentor Graphics (both of which support IEEE 1735-2014), Libero SoC (v11.3 or later) enables a seamless design flow for designers targeting SmartFusion2, IGLOO2, RTG4, and PolarFire when they use encrypted IP cores in their design.

Use of IP cores not only shortens the design cycle time but also provides proven and reliable design components for re-use in multiple applications. Using an IP core in the EDA design flow involves two conflicting considerations that must be resolved: IP security and IP interoperability across different EDA tools.

Libero SoC Secure IP Design Flow Requirements

Table 1 lists the software and hardware requirements for Designing with Secure IPs in Libero SoC

Design Requirements	Description
Hardware Requirements	
SmartFusion2 / IGLOO2 Family devices	This feature is supported for SmartFusion2 / IGLOO2 family devices.
Host PC or Laptop	Windows 64-bit Operating System / Linux 64-bit Operating System.
Software Requirements	
Libero System-on-Chip (SoC)	v11.3 or Later
Synplify Version	Synplify Pro ME I2013.09MSP1 or later
ModelSim Version	ModelSim 10.2c or later
Encryption Script Requirements	
OpenSSL	Most Linux and Cygwin have OpenSSL pre-installed. Provide OpenSSL installation location in “PATH” environment variable of system.
Perl	Any version of Perl with following packages installed: FindBin, Math, Getopt, File, MIME
Cygwin (for Windows OS)	For executing Perl Script on Windows OS
Public Keys for encryption	Provided on request

IP Security and IP Interoperability Across Design Tools

Use of IP cores not only shortens the design cycle time but also provides proven and reliable design components for re-use in multiple applications. Using an IP core in the EDA design flow involves two conflicting considerations that must be resolved: IP security and IP interoperability across different EDA tools.

IEEE 1735-2014 Standards for IP and EDA Vendors

IEEE 1735-2014 is an encryption scheme proposal adopted by most IP and EDA vendors to ensure the interoperability of IP cores among the IP vendors and EDA tools. The objective of IEEE 1735-2014 is to serve the IP vendors and the EDA community in the following ways:

- For the IP vendor, protect the security of the IP core in the design flow across different EDA tools.
- For the IP core users and EDA tool vendors, ensure the interoperability of the IP core across different EDA tools.

Encryption Algorithms

Libero SoC supports the following encryption algorithms:

- des-cbc
- 3des-cbc
- aes128-cbc
- aes256-cbc

There are two major classes of encryption methodologies: Symmetric and Asymmetric.

Symmetric Encryption

This encryption scheme uses a special string as a key to encrypt the data. The same key is used to decrypt the data (Figure 1). Examples of this type of encryption algorithms include:

- Data Encryption Standard (DES), such as des-cbc.
- Triple DES or TDES or TDEA (Triple Data Encryption Algorithm) which uses the DES algorithm three times, such as 3des-cbc.
- Advanced Encryption Standard (AES), such as aes128-cbc and aes256-cbc.

Asymmetric Encryption

This encryption scheme uses two different keys: one for encryption and another for decryption. The end user generates two keys, one public and one private. The end user distributes the public key to whoever needs it for encryption and keeps the private key to use for decryption (Figure 2).

Common examples of asymmetric encryption algorithms are:

- DH (Diffie-Hellman)
- RSA (Rivest, Shamir, and Adelman).

Two Levels of Encryption

There are two levels of encryption when producing an encrypted IP core. First, the IP core vendor uses a session (random) key to encrypt the IP content. This is the first level of encryption (Figure 1). Then the IP core vendor uses the Public Keys from EDA vendors to encrypt the session key. This is the second level of encryption (Figure 2).

A Public Key must be provided for each EDA tool to the IP core vendor. For Libero SoC customers who use third-party IP's in their design, the EDA vendors are:

- Synopsys for Synplify Pro
- Mentor Graphics for ModelSim
- Microsemi for Libero SoC

The result of the first level of encryption is the encrypted data block. The Random session key required for symmetric encryption of the data block is generated by the encryptP1735.pl script.

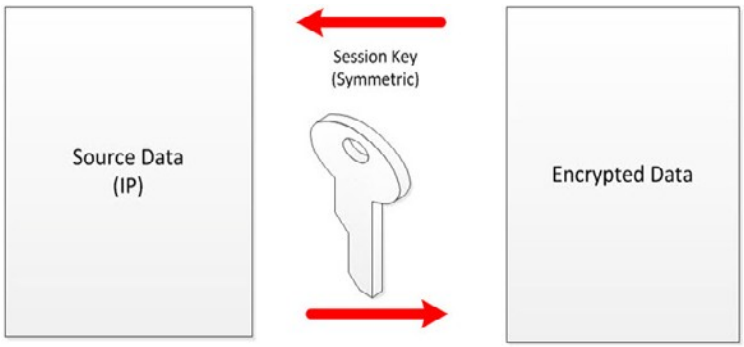

***Figure 1* • Data Encryption of Source Data IP**

The result of the second level of encryption is the encrypted session key.

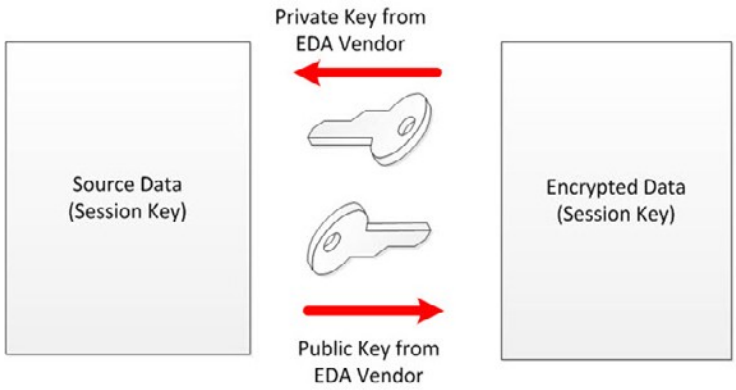

***Figure 2* • Session Key Encryption**

Encryption Envelopes

The encryption envelope is the preamble to the IP in the HDL file. The IP core vendor must prepare an encryption envelope for all EDA tools which are intended to be used with the IP. The encryption envelope consists of pragma keywords (see "Pragma Keywords" on page 12) that provide the following information:

- Encryption Version
- Encoding Type
- Encryption Agent
- Key Owner
- Key Name
- Key Method

An example of an Encryption Envelope appears below. Note that the encryption envelope identifies three EDA tool vendors/key owners.

```
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-SIM-
RSA-1",key_method="rsa",key_block
`pragma protect key_keyowner="Microsemi Corporation",key_keyname="MSC-IP-KEY-
RSA",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
```

Decryption Envelopes

The Decryption Envelope is the preamble to the encrypted IP. The Decryption Envelope consists of pragma keywords (see "Pragma Keywords" on page 12) that provide the following information:

- Encryption Version
- Encoding Type
- Encryption Agent
- Key Owner
- Key Name
- Key Method

A Verilog example of the Decryption Envelope appears below.

```
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+u7Ivw
1m2BydGyW7MAQx2ePgbrKQbRLaN8XF/iiUFUX0QXnWDZrxtgcVHULOsPXpwd25wNyeWQkTekAsln
ubKiFDfNySxaP5W3SboZE0pMLqH+mpZlcvKljlE30uOAQQLjECEBGj1KxMZQ2hhUKLrXz34+9p68
tVzbM/u1TbsXvdPcN23UItAxNPSH5ND75rAviq7ACIVawH87/m2RshSDSVcmz7ndMpSJRQOFe2pd
usuHdCFJm1YaEaCZYfqReV7RjCzbV48d3LPtoA==

`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIM-
RSA-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+3o0rtdoggobQ7e
3LQ5Bhjfcudafujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrW
xbaIlfktti2lBBdU/SDV83mLYKzAqe20/SaZR5FAZH8cyuUPxYOviHQ/fpqNwUao
U/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8OpNWDY4fZ4g9vQFB
hmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1O
Zvaxh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6N
Bc4EiyOT2iofDDtqoxdLZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtS
ry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj2145x9vnXXINiuOIuIj1K/a2dj
kP+2A3Jvt53z8gv9Jij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp
/ZSoVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuX
IPIODGoHEdFC4feve5uuucMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKS
F3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQOrQw0pBvWm0DyUFRTJ53GZAfbEz+
1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeUvbmcuRL8wVMHvXo58
6qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0dM+
QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2C
qbSB58Ykljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY
1JekLYeGaSX0WzVcxQcA3flpGL+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Yt
```

```
py8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXgP0PhR1smfcEFeUTwe56nDDp
BJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GlIMW8KNBNtbn6qFYaMq2uG
6AmwTZAVfhru0yjnIELj3k3t/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
RgKC7i4hx7zh3MLd50RYrZoCwPWFEyLwISIXDLkpkL6qFgFm1WmZEwFvZjNfQCNUgoSHeIRpxg9i
lXnvMiBjQCiQVvMp32UtfSX625K8+yvJLMPdHQ8G/2qxa6ViHAhBhRcsSUl0XGskRmU3JvNuNfAk
0IoB1HpFEJ0Vv6vEI5g=

`pragma protect end_protected

endmodule
```

1 – Securing Your IP Core

As an IP vendor, you must protect your Intellectual Property and package your IP core such that it is inter-operable with EDA tools without compromising security. Encryption is managed on two levels (Figure 1-1):

- IP Core Encryption
- Data Key Encryption

To encrypt an IP core with the IEEE 1735-2014 scheme:

1. Obtain the Public Key (see "Public Key from EDA Vendors" on page 9) from each downstream EDA tool vendor.
2. Add the Encryption Envelopes (see "Encryption Envelopes" on page 5) to the RTL code. Ensure that all required EDA tool vendors are included.

3. Execute the encryptP1735 Perl Script.

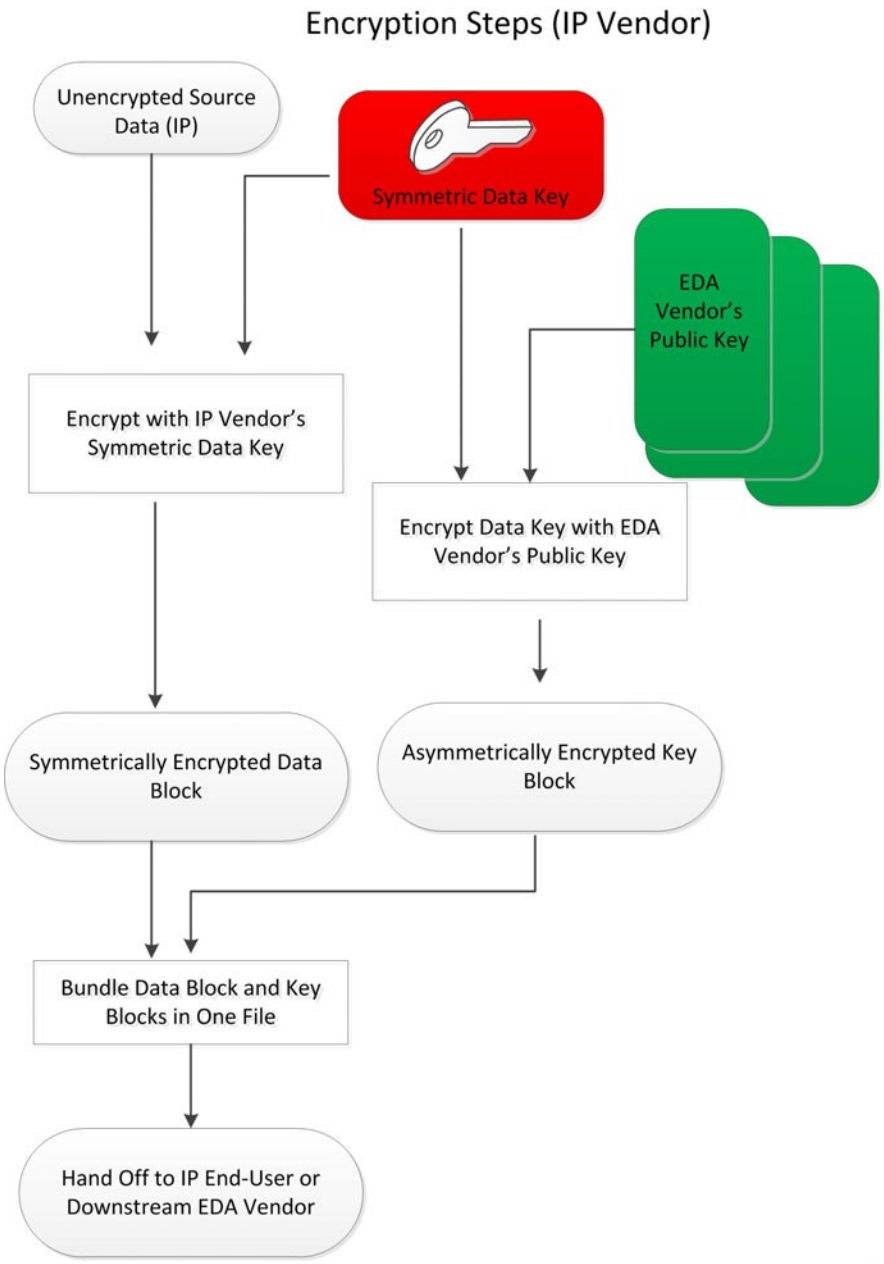

Figure 1-1 **• IP Encryption**

Public Key from EDA Vendors

Obtain a Public Key from each downstream EDA tool vendor. Aggregate the Public Keys from the vendors into a single Public Keys Repository File. An example of the file is shown below.

Note: **The keys shown below are dummy keys and will not work. To request the public key file that contains all 3 public keys along with the Perl script to encrypt your files, please email soc_marketing@microsemi.com**.

```
`pragma protect key_keyowner="Synplicity",key_keyname="SYNP05_001",key_public_key
-----BEGIN PUBLIC KEY-----                    Public Key from Synopsys
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAybsQaMidiCHZyh14wbXn
UpP8lK+jJY5oLpGqDfSW5PMXBVp0WFd1d32onXEpRkwxEJLlK4RgS43d0FG2ZQ1l
irdimRKNnUtPxsrJzbMr74MQkwmG/X7SEe/lEqwK9Uk77cMEncLycI5yX4f/K9Q9
WS5nLD+Nh6BL7kwR0vSevfePC1fkOa1uC7b7Mwb1mcqCLBBRP9/eF0wUIoxVRzjA
+pJvORwhYtZEhnwvTblBJsnyneT1LfDi/D5WZoikTP/0KBiP87QHMSuVBydMA7J7
g6sxKB92hx2Dpv1ojds1Y5ywjxFxOAA93nFjmLsJq3i/P0lv5TmtnCYX3Wkryw4B
eQIDAQAB
-----END PUBLIC KEY-----
`pragma protect key_keyowner="Mentor Graphics Corporation",key_keyname="MGC-VERIF-
SIM-RSA-1",key_public_key
-----BEGIN PUBLIC KEY-----                    Public Key from Mentor Graphics
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCnJfQb+LLzTMX3NRARsv7A8+LV
5SgMEJCvIf9Tif2emi4z0qtp8E+nX7QFzocTlClC6Dcq2qIvEJcpqUgTTD+mJ6gr
JSJ+R4AxxCgvHYUwoT80Xs0QgRqkrGYxW1RUnNBcJm4ZULexYz8972Oj6rQ99n5e
1kDa/eBcszMJyOkcGQIDAQAB
-----END PUBLIC KEY-----
`pragma protect key_keyowner="Microsemi Corporation",key_keyname="MSC-IP-KEY-
RSA",key_public_key
-----BEGIN PUBLIC KEY-----                    Public Key from Microsemi
MIID4jANBgkqhkiG9w0BAQEFAAOCA88AMIIDygKCA8EAxvOR7+3o0rtdoggobQ7e
3LQ5Bhjfcudafujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrW
xbaIlfktti2lBBdU/SDV83mLYKzAqe20/SaZR5FAZH8cyuUPxYOviHQ/fpqNwUao
U/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8OpNWDY4fZ4g9vQFB
hmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1O
Zvaxh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6N
Bc4EiyOT2iofDDtqoxdLZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtS
ry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj2145x9vnXXINiuOIuIj1K/a2dj
kP+2A3Jvt53z8gv9Jij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp
/ZSoVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuX
IPIODGoHEdFC4feve5uuucMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKS
F3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQOrQw0pBvWm0DyUFRTJ53GZAfbEz+
1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeUvbmcuRL8wVMHvXo58
6qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0dM+
QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2C
qbSB58Ykljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY
1JekLYeGaSX0WzVcxQcA3flpGL+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Yt
py8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXgP0PhR1smfcEFeUTwe56nDDp
BJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GlIMW8KNBNtbn6qFYaMq2uG
6AmwTZAVfhru0yjnIELj3k3t/OS/YbA6wRFpg0GddNNRAgMBAAE=
-----END PUBLIC KEY-----
```

Adding an Encryption Envelope to Your RTL

You must add the Encryption Envelopes (see "Encryption Envelopes" on page 5) to the RTL codes. All EDA tools that need access to the encrypted data block must be included and identified as a key owner in the Encryption Envelope.

An example of a Verilog IP core and a VHDL IP core with an Encryption Envelope is provided below. The envelope identifies Microsemi, Synopsys and Mentor Graphics as key owners.

Verilog IP Core with Encryption Envelope

```
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-
SIM-RSA-1",key_method="rsa",key_block
`pragma protect key_keyowner="Microsemi Corporation",key_keyname="MSC-IP-KEY-
RSA",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
```

VHDL IP Core with 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
```

```
    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-KEY-
RSA",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;

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

Pragma Keywords

Table 1-1 describes the Pragma keywords in the Encryption Envelope.

***Table 1-1* • Pragma Keywords**

Pragma Keywords	Description
begin	Opens a new encryption envelope
end	Closes an encryption envelope
begin_protected	Opens a new decryption envelope
end_protected	Closes a decryption envelope
author	Identifies the author of an envelope
author_info	Specifies additional author information
encoding	Specifies the coding scheme for the encrypted data
data_keyowner	Identifies the owner of the data encryption key
data_method	Identifies the data encryption algorithm
data_keyname	Specifies the name of the data encryption key

***Table 1-1* • Pragma Keywords (continued)**

Pragma Keywords	Description
data_public_key	Specifies the public key for data encryption
data_decrypt_key	Specifies the data session key
key_keyowner	Identifies the owner of the key encryption key
key_method	Specifies the key encryption algorithm
key_keyname	Specifies the name of the key encryption key
key_public_key	Specifies the public key for key encryption
key_block	Begins an encoded block of key data
version	P1735 encryption version

encryptP1735.pl Script

Execute the encryptP1735.pl script to encrypt your IP. The encryptP1735 script is a Perl script that Synopsys makes available to IP vendors for encryption of their IP cores.

Note: Before running the script, make sure that Open SSL is installed on your machine. Open SSL is required for the script to work.

Note: For Windows OS, it is recommended that the script be executed in the Cygwin Environment on Windows.

The example command below invokes the script with a random key to encrypt the data block:

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

where

-input *secret.v*	Specifies secret.v as the input file to the script; the input file is the non-encrypted HDL file containing one or more encryption envelopes.
-output *secret_enc.v*	Specifies secret_enc.v as the name of the encrypted output file after running the encryption script.
-pk *public_keys.txt*	Specifies public_keys.txt as the public keys repository file. This file contains public keys for all downstream EDA tools. The public keys file must include public keys for all EDA vendors mentioned in the encryption envelope.
-om *encrypted*	Specifies how the IP is treated when generating the synthesis netlist; encrypted is the default mode. In this mode, the same data key used for encryption of the IP is used in the output synthesis netlist.
-v	Specifies the script runs in verbose mode

Output Encrypted File

The output file generated by the script contains pragma directives for decrypting the encrypted data (IP core) and the data key used to encrypt the data. The example below shows a Verilog and a VHDL of the output.

Output Encrypted Verilog

```
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+u7Ivw
1m2BydGyW7MAQx2ePgbrKQbRLaN8XF/iiUFUX0QXnWDZrxtgcVHULOsPXpwd25wNyeWQkTekAsln
ubKiFDfNySxaP5W3SboZE0pMLqH+mpZlcvKljlE30uOAQQLjECEBGj1KxMZQ2hhUKLrXz34+9p68
tVzbM/u1TbsXvdPcN23UItAxNPSH5ND75rAviq7ACIVawH87/m2RshSDSVcmz7ndMpSJRQOFe2pd
usuHdCFJm1YaEaCZYfqReV7RjCzbV48d3LPtoA==
`pragma protect key_keyowner="Mentor Graphics Corporation", key_keyname="MGC-VERIF-SIM-
RSA-1", key_method="rsa"
`pragma protect encoding=(enctype="base64", line_length=76, bytes=128)
`pragma protect key_block                          Mentor Graphics 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                                  Microsemi Key Block
MIID4jANBgkqhkiG9w0BAQEFAAOCA88AMIIDygKCA8EAxvOR7+3o0rtdoggobQ7e
3LQ5Bhjfcudafujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrW
xbaIlfktti2lBBdU/SDV83mLYKzAqe20/SaZR5FAZH8cyuUPxYOviHQ/fpqNwUao
U/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8OpNWDY4fZ4g9vQFB
hmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1O
Zvaxh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6N
Bc4EiyOT2iofDDtqoxdLZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtS
ry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj2145x9vnXXINiuOIuIj1K/a2dj
kP+2A3Jvt53z8gv9Jij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp
/ZSoVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuX
IPIODGoHEdFC4feve5uuucMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKS
F3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQOrQw0pBvWm0DyUFRTJ53GZAfbEz+
1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeUvbmcuRL8wVMHvXo58
6qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0dM+
QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2C
qbSB58Ykljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY
1JekLYeGaSX0WzVcxQcA3flpGL+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Yt
py8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXgP0PhR1smfcEFeUTwe56nDDp
BJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GlIMW8KNBNtbn6qFYaMq2uG
6AmwTZAVfhru0yjnIELj3k3t/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
RgKC7i4hx7zh3MLd50RYrZoCwPWFEyLwISIXDLkpkL6qFgFm1WmZEwFvZjNfQCNUgoSHeIRpxg9i
```

```
lXnvMiBjQCiQVvMp32UtfSX625K8+yvJLMPdHQ8G/2qxa6ViHAhBhRcsSUl0XGskRmU3JvNuNfAk
0IoB1HpFEJ0Vv6vEI5g=

`pragma protect end_protected

endmodule
```

Output Encrypted VHDL

```
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
EzupxwpLZCgcCoy7O42J4O6TjEXDsFHlEXYIfYKVXIsm/8incqBuPuWZ26osQcaegOtanunB7lPo
sTFjlZBLLgsDLE/Pl7j8PhcxhySoKy/8TkZClQf7osKMbfeFAMFtIOAqjGT4Ab2F9DdosbC6QkNY
FCVLJSk5nNBeA6bslznTicV416exZcHTV5tJycz2vkFVlRY+BBtcXlhBrxCZSguf9OwHkr0OcufC
jKaHE//kfF1dlJ1jjcuidCnJ5rOtG3BDWFQ7f/ClH6H9IkqikEfDy2qGO4Kz1N8OF6sH2MKCj4O5
ye7d1aH+QH3FrTmoNgnVg9f7McoZ0Ito4Z1qCQ==

`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+/HYByVzO7Iq9WSlfEgti2EYSXVTU974UChUeOJwTJUA5z24gL1gI2QF3I
SYQs6NgHG84V+DMh9s3biK9UDHz4KJqa5Xrsx6QwvD6co3rZ09bzNPL8w9uGaPK40DXWTQbY0T6W
pDdIw9u4pvhII/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+3o0rtdoggobQ7e
3LQ5Bhjfcudafujkinm+213ui89cvxjkaYKRDadsklgfklDGTFyiYUIKasKv3MrW
xbaIlfktti2lBBdU/SDV83mLYKzAqe20/SaZR5FAZH8cyuUPxYOviHQ/fpqNwUao
U/3jp4nvc76K/FO14W56I/hXb23/0s8zzyny3gHfqcEu8Dn8OpNWDY4fZ4g9vQFB
hmv71HjJl0NRvvJHrXYmCEwlWPQjzru+8lj4JhBx/9ChKskTpVB6vkV//IX5Od1O
```

```
Zvaxh5x+xPCSKEgbmjv0uxaXtvnBJQa4xdMM7eHglGDSbZ2A13cg1qtxrCn05f6N
Bc4EiyOT2iofDDtqoxdLZPb4L6UDIR+EY1o+1l1mDBrBvqn6hQtpUoi+bgWe+xtS
ry30qmJkjjkejkJKJk+258uUI622kjlCCVGijj2145x9vnXXINiuOIuIj1K/a2dj
kP+2A3Jvt53z8gv9Jij9xC90725pCl5Cziw4XsBsg+jJJEn4IpqvwgoA/7SkDpZp
/ZSoVRgMfDvn60mzc/0Y6dtaX4FTsyJiduQBtKNtssGSVQGajOKEcfUOVgslkwuX
IPIODGoHEdFC4feve5uuucMbHw8pmjI0dYGz0XIcU5dZNW1yVvNaPXC7cKvIeuKS
F3bogXenDzZ40/6+n9kRRS74vzdOMv5CSoxQOrQw0pBvWm0DyUFRTJ53GZAfbEz+
1IU+cwAMmQR7FMpbJtaKJeNdccHe/nOm4kdnW6W00FxUVeUvbmcuRL8wVMHvXo58
6qDuHOk0LPXK+KLRr5P1QyD7b78t4PJOmbKgT0xQd8h1Oun2j61ZfQsvaguF0dM+
QOO+EWoUU0+1I4eCzMG38R927w9kT8jJCPmIF2DT5tSB0JWIMC+Md6u0HFKUPG2C
qbSB58Ykljvoiu70Avay79vAAREvjkjlVWYKLJMjiuvaweRGPWtKdeBXwOHNSFRY
1JekLYeGaSX0WzVcxQcA3flpGL+4SdjdRWDYK3wXv6QoQ9YVag78nMIYUECtz+Yt
py8dTIjdp3d+KDsJ8t0dYkvHETiv8QoDNeutIZZXgP0PhR1smfcEFeUTwe56nDDp
BJJsyaybQhj76+tz1346gymRTEasBTlklnmu6XafYJ290fklsfdjkYjklaqoviDZ
1OphMGkNCqUa0JslpPBuPbVAgEBB4R3MUNQZpR9W7GlIMW8KNBNtbn6qFYaMq2uG
6AmwTZAVfhru0yjnIELj3k3t/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+dMErJ9rG4vuwDcoqErHHk4oToYBn4ZavftY
DJc1W3U7+dxEN3lVcgRsWveZZ0ePIfkkEKhp7cSgfFt5kFfwPEoMHPDhAPeElMr84o0pYEiFdO6V
GwOJgULvGsFedDKwWnTn6O9FbtKBKuKyl8NG27C89GRtkr4UhguNgVDJKs/O8E9bHlSlyxSh2sD4
GnTPLAVC4NONi4HjsBhxVGvq04yjbJwOHohjI/WeY26ZqHJN7jqkKrdOhXTi/DRoCY15vjfvALr1
kzErv8zjc9qGqBWucHhmUgwfKzp6p8XfFPHTZlOnsKigVN9Q8Kmu6ZmN3nYadlK8ASo4A7q3v9mA
otx6

`protect end_protected

end behv;
```

Packaging and Bundling the Encrypted IP and the Data Key

When you execute the encryptP1735.pl script, you bundle and package the Encrypted IP and the Encrypted Data Key together in one single file, which is the output of the script. This file is ready for delivery to your customers.

2 – Running Libero SoC with Encrypted IP

Libero System-on-Chip (SoC) software v11.3 or later supports the use of third-party encrypted IP cores in the design flow for SmartFusion2, IGLOO2, RTG4, and PolarFire families (Figure 2-1).

To run Libero SoC with Encrypted IP:

Libero SoC software support for encrypted IP is enabled by default. Follow the steps below to incorporate an encrypted IP inside the Libero software.

1. Set your Project Settings so that the synthesized output is a Verilog netlist.
2. Import the encrypted IP core as HDL (page 19).
3. Run synthesis (page 20) and simulation (page 22).
4. Run the rest of the Libero SoC design flow as shown in Figure 2-1.

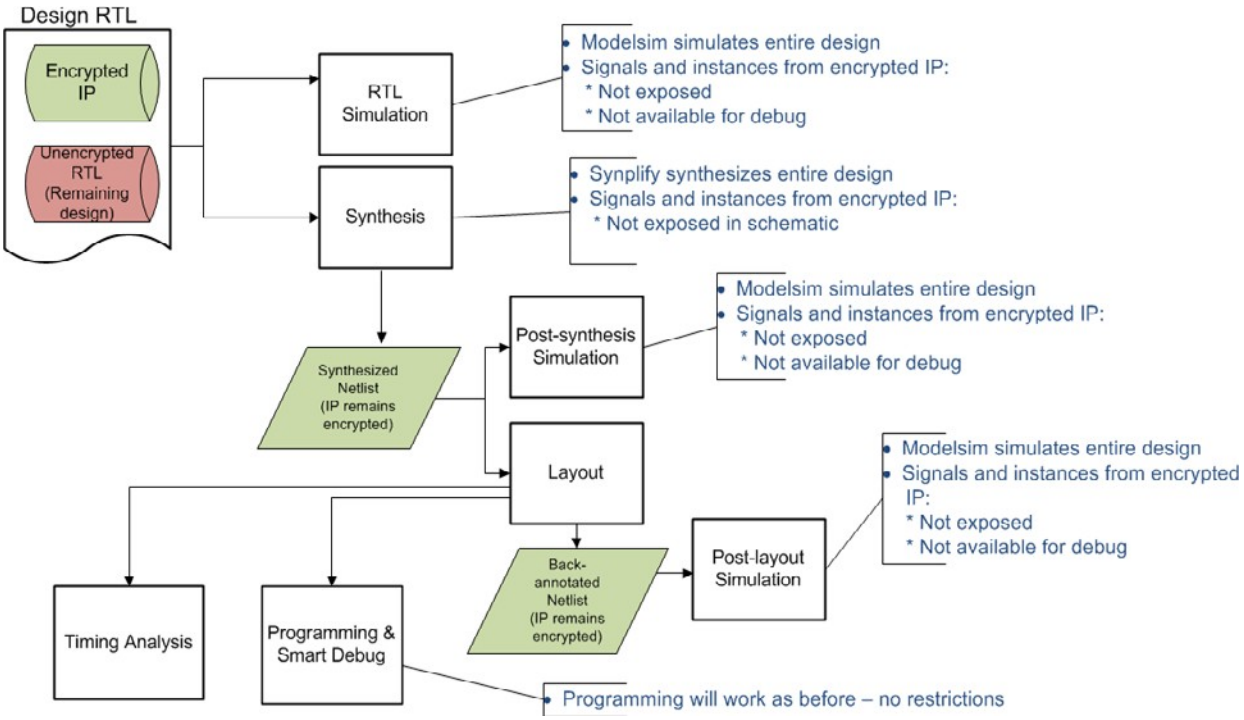

Figure 2-1 **• Encrypted IP Design Flow**

Encrypted IP Design Flow Must Use a Verilog Netlist from Synthesis

When creating a new project, you must change Project Settings to support the IEEE 1735-2014 secure IP flow. You can then import the encrypted IP core as Verilog or VHDL source files.

The IEEE 1735-2014 scheme supports only Verilog as the netlist format; EDIF format is not supported. You must set Libero SoC Project Settings to use the Verilog netlist from Synthesis.

1. From the **Project** menu, choose **Project Settings > Design Flow**.
2. Select **Verilog Netlist** as the Synthesis Gate Level Netlist format (Figure 2-2).
3. Click **Save** and then **Close.**

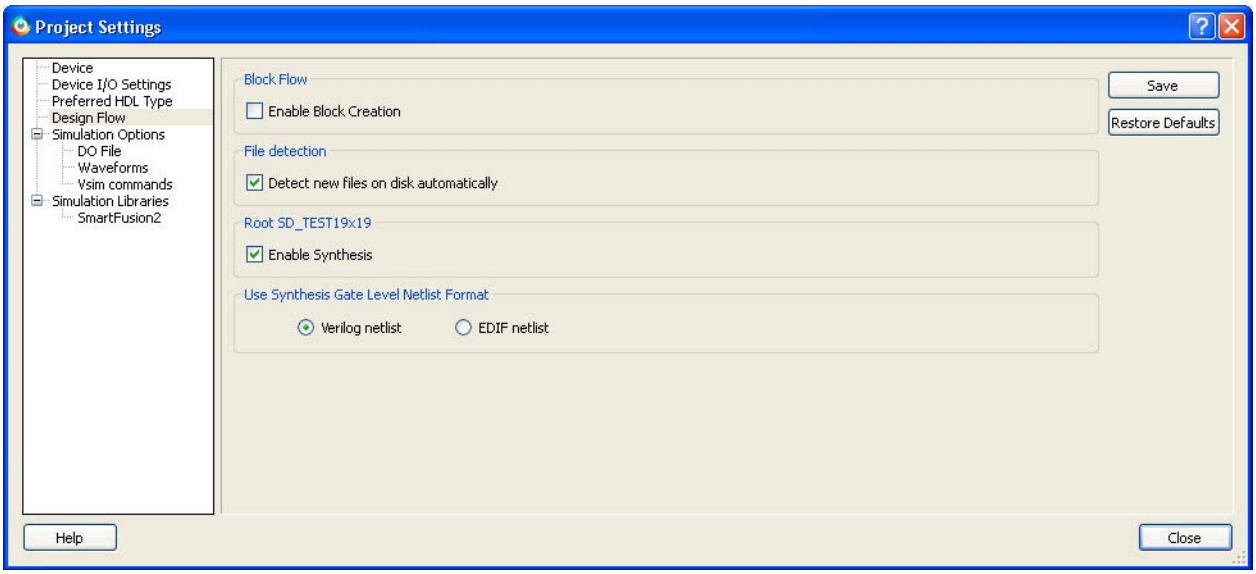

***Figure 2-2* • Project Settings**

Import Encrypted IP Core as HDL

Import the encrypted IP HDL and the non-encrypted HDL file as HDL source files (**File > Import > HDL Source Files**). The Design Hierarchy displays the imported file in your design (Figure 2-3).

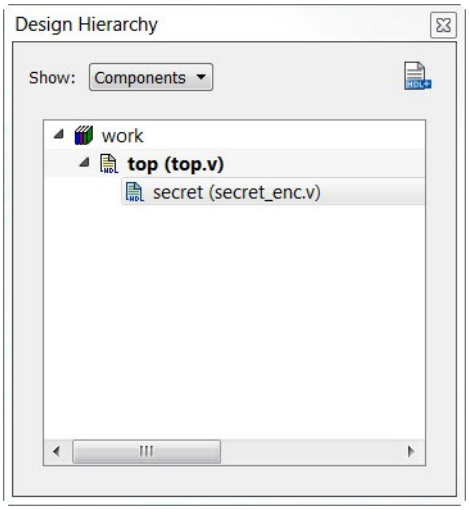

***Figure 2-3* • Design Hierarchy**

Note: It is recommended that the encrypted IP be presented as a single file. If the IP is currently organized in a hierarchy of files, it is recommended that the entire IP be concatenated into a single file after encryption. Currently, if the encrypted IP is defined in multiple files, the user needs to pass the (lower level) files manually to synthesis & RTL simulation steps. This is done from Organize input file option Synthesis / Simulation tool, as shown below. Refer to Organize Source file in Libero Help for more information on how to organize source files.

Figure 2-4 shows how to organize input source files for Synthesis..

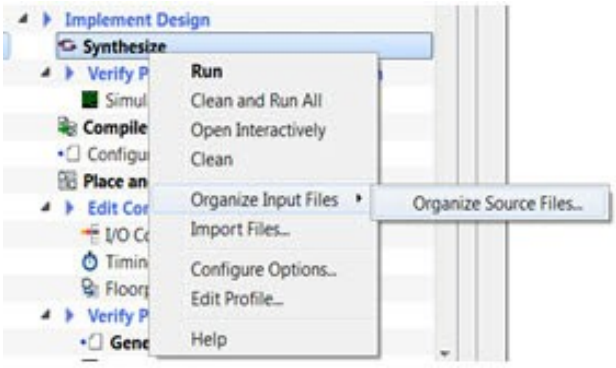

***Figure 2-4* • Organizing input source files for Synthesis**

Smart Design Support

SmartDesign is a visual block-based design creation tool for instantiation, configuration and connection of Microsemi IP, user-generated IP, custom/glue-logic HDL modules. Encrypted IP can also be instantiated in a SmartDesign, along with other non-encrypted IP. Refer to About SmartDesign in Libero Help for more information.

Run Synthesis

After synthesis, only the interface signals (inputs and output ports) of the Secure IP core are visible in the RTL and Technology views (Figure 2-5 and Figure 2-6). Signals and instance names internal to the Encrypted IP are not visible.

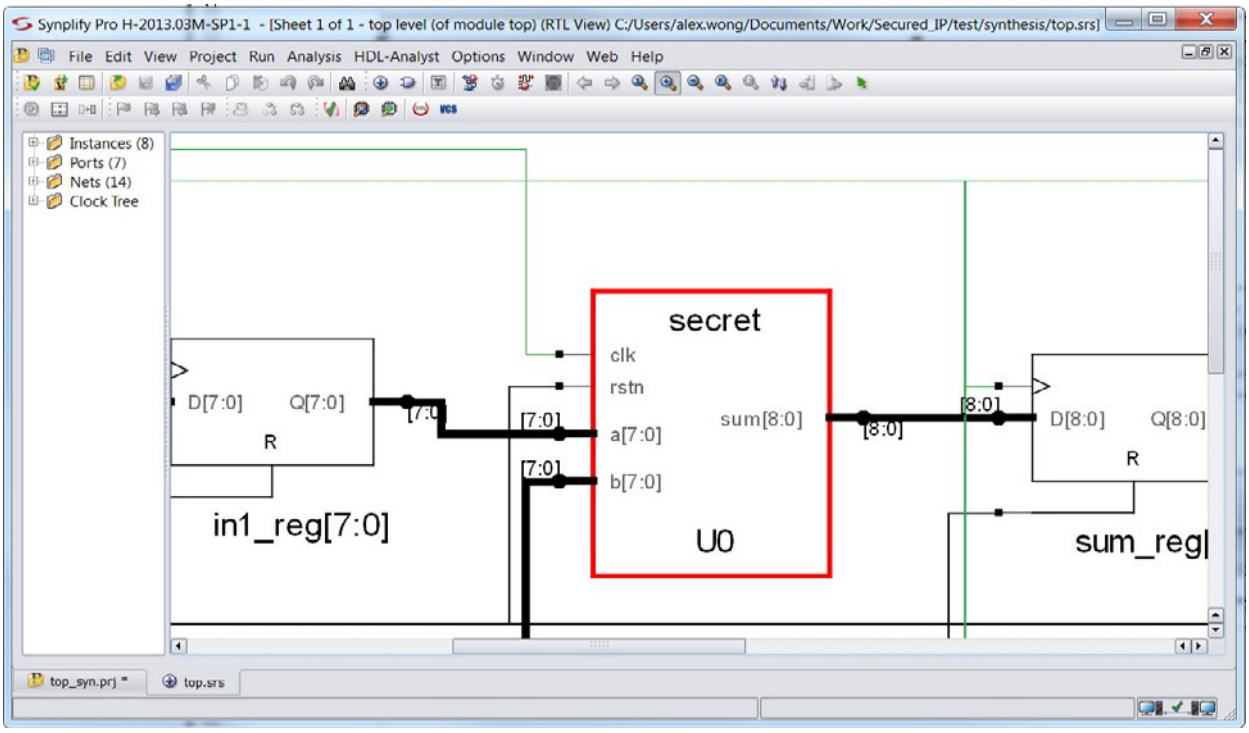

***Figure 2-5* • SynplifyPro RTL View**

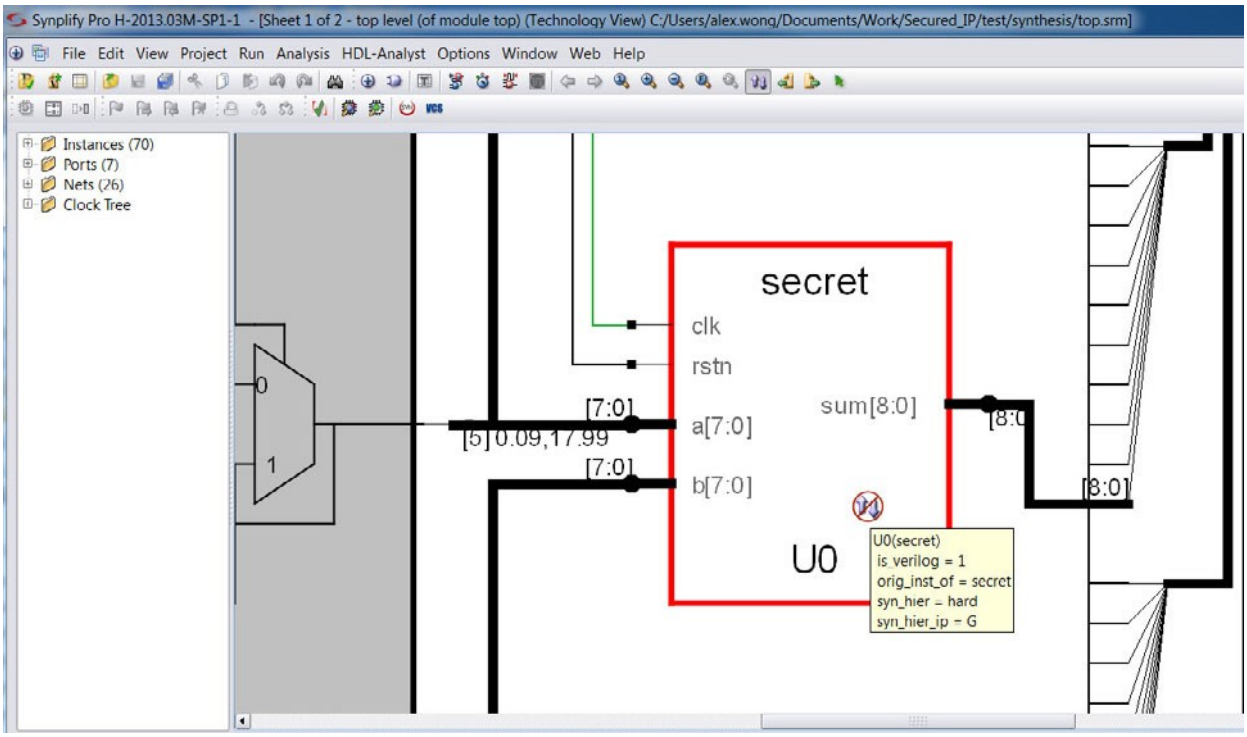

Figure 2-6 • **SynplifyPro Technology View**

In the RTL and Technology Views, the push and pop commands are disabled for design blocks encrypted with IEEE 1735-2014. You cannot push into the encrypted IP block ‘U0’ to look at the internal signals, nets or instances inside the encrypted block.

Run ModelSim Simulation

ModelSim simulates the entire design for pre-synthesis, post-synthesis and post-layout simulations. However, the signals and instances internal to the encrypted IP are not exposed and are not available for debug.

The values of the internal signals are not displayed in the waveform window; only the interface signals at the boundary of the encrypted IP instance ‘U0’ are displayed (Figure 2-7).

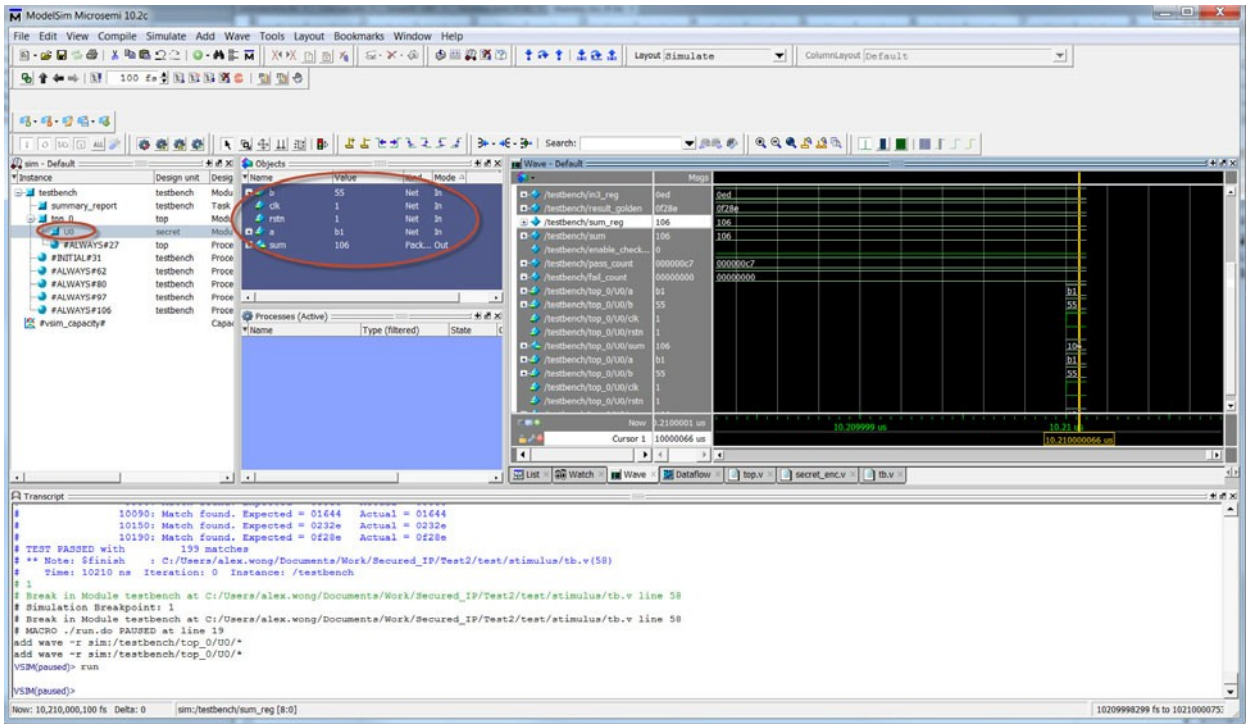

Figure 2-7 • **Modelsim Simulation of Encrypted IP Core**

Note: Simulation is supported for both Verilog and VHDL.

Libero SoC and Encrypted IPs

Libero SoC Software processes designs with encrypted IP through the entire Design without compromising the encrypted IP content. The encryption of IP is protected in Synthesis and Simulation tools, as mentioned in earlier sections.

All netlists exported from Libero SoC have the IP component encrypted. These include:

- Back annotated netlist after Place and Route: *_ba.v or *_ba.vhd
- Exported netlist after compile: *.v or *.vhd

Microsemi adheres to the Encryption Guidelines in the IEEE 1735-2014 standard throughout the design flow.

Example

This section shows an example in which an Encrypted Module is implemented using the Libero SoC Secure IP Flow.

The example consists of the following files:

Secret.v:- This is a simple Non-Encrypted Verilog module. This module has encryption envelopes as mentioned on page 5.

Secret_enc.v:- This is the encrypted version of Secret.v module which has been encrypted by executing encryptP1735.pl script on Secret.v module.

Top.v:- This is a top level module instantiating encrypted secret_enc.v module.

Tb.v:- Test Bench for Top.v module.

Public_keys.txt:- This is Text File containing Public Keys from Synopsys, Mentor and Microsemi as mentioned on page 9.

Encryption of IP Module

We are going to use to encryptP1735.pl script which implements IEEE 1735-2014 standard for encryption of IP Module (secret.v in this example). The segment of the code that needs to be encrypted have to be included within Encryption Envelopes. (Refer to "Adding an Encryption Envelope to Your RTL" on page 11).

All Public Keys from vendors supporting this standard are stored in a single file Public_Keys.txt.

Execute encryptP1735.pl script with the secret.v as input file and secret_enc.v as output file.

Figure 2-8 shows an example of output after encryptP1735.pl has been executed on the secret.v module.

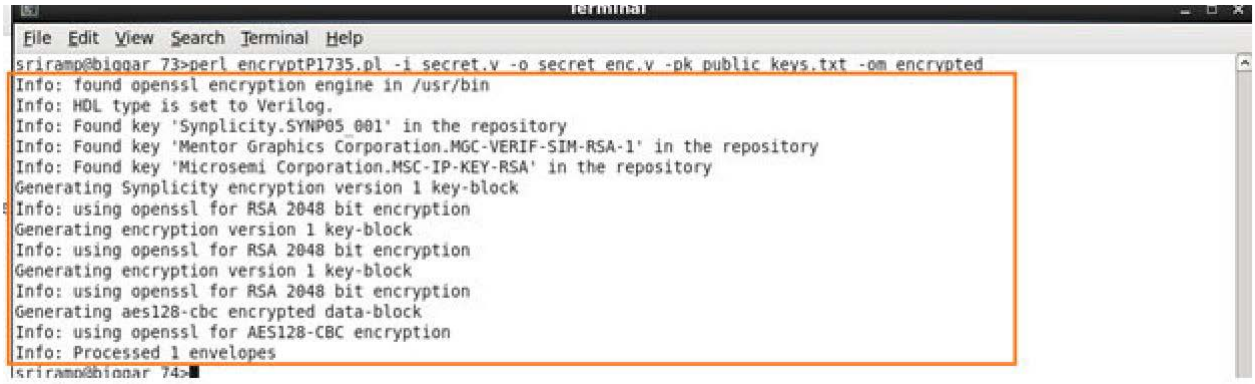

Figure 2-8 • **Output of EncryptP1735.pl script**

The output file is similar to "Output Encrypted Verilog" on page 14.

Observe that the encrypted output source file of the IP have key_blocks corresponding to all the Vendors and Data_blocks with encrypted information.

Note: Refer to "encryptP1735.pl Script" on page 13 for more information about different parameters of the script. The script can be executed on both Windows and Linux OS with openSSL and Perl Installed.

Importing Encrypted IP in Libero SoC

Import an Encrypted module in the same way you import any HDL file into a Libero Project.

Create a Libero Project with SmartFusion2 / IGLOO2 / RTG4 / PolarFire family die. Import Top.v and Secret_enc.v files (**File > import > HDL Source Files**) into the Libero Project. Also import the corresponding Test bench file tb.v (**File > Import > HDL Stimulus Files**). On importing these files, your design hierarchy and stimulus hierarchy appear as shown in Figure 2-9 and Figure 2-10.

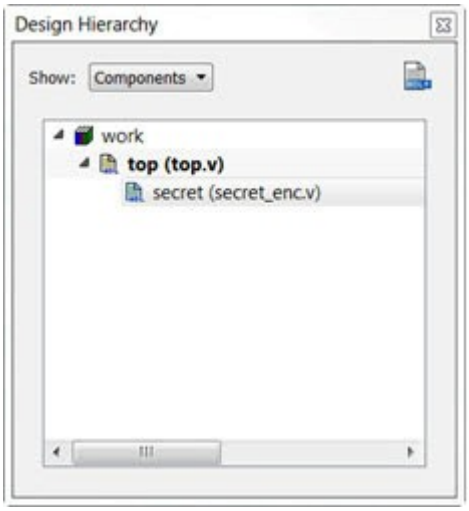

Figure 2-9 • **Design Hierarchy**

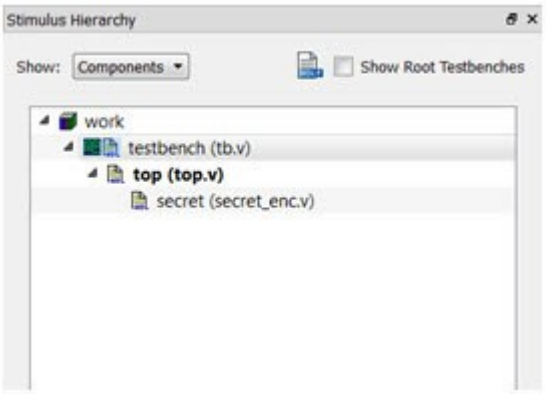

Figure 2-10 • **Stimulus Hierarchy**

There can be multiple instantiations of an encrypted module in a Top Level module or Smart Design.

Select top.v as the Root module (Right-click > **Set as Root)**. Change Synthesis netlist format to Verilog Netlist from the Libero Project Settings Menu (**Project > Project Settings > Design Flow**) as shown in Figure 2-11.

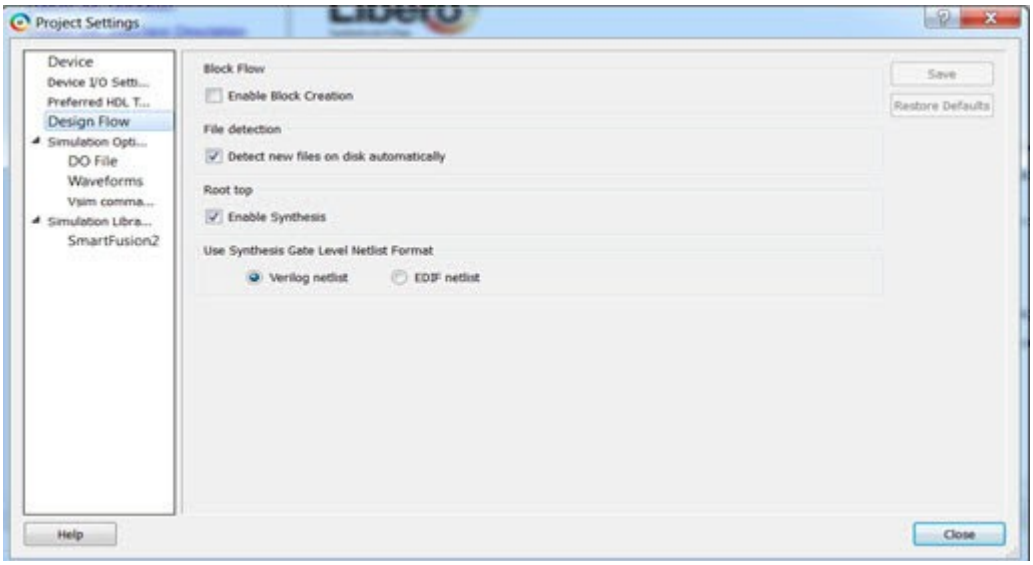

Figure 2-11 • **Project Settings for Netlist Format**

Synthesis

The Synthesis tool (Synplify Pro) decrypts the protected content using Synopsys Key Block present in Encrypted Module secret_enc.v. After synthesis, only the interface signals (inputs and output ports) of secure IP core are visible in the RTL and Technology views. Refer to "Run Synthesis" on page 20 for more information. The Verilog netlist file (.vm file) obtained after synthesis does not show internal instances of encrypted module and this information is again re-encrypted by synthesis tool.

Simulations

The Simulation tool (ModelSim) decrypts the protected content using the ModelSim Key Block present in Encrypted Module secret_enc.v. ModelSim simulates the entire design for pre-synthesis, post-synthesis and post-layout simulations. However, the signals and instances internal to the encrypted IP are not exposed and are not available for debug. Refer to "Run ModelSim Simulation" on page 22 for more information.

Compile and Layout

The rest of the tools in the Libero SoC Design Flow decrypt the protected content using Microsemi Key Block present in Encrypted Module secret_enc.v.

Once synthesis is completed, the Compile tool takes the encrypted .vm netlist file as input for further processing by the Layout Tool. The execution and output of these tools are similar to the Regular Flow.

Note: Constraints flow, including Timing Constraints and Floorplan Constraints, are not supported for instances inside encrypted Blocks. In the above example, Constraint flow is not supported for secret_enc.v module. However users can provide constraints to the interface of the encrypted module.

Generate Back Annotated Files

Once Layout is complete, users can generate the Back Annotated Files for Post-layout simulations. The *_ba.v or *_ba.vhd files generated shows the internal information of secure_enc.v module as encrypted. These file incorporate Key_Block from Mentor which is used for decryption while running Post-Layout Simulations.

Generate Programming Data

Once the design has completed Layout and Post-Layout Simulations, users can generate the Programming file.

Frequently Asked Questions

Below are some Frequently Asked Questions about Secure IP flow and its support in Libero SoC.

1. Are VHDL simulations supported, since we are using a Verilog Netlist?

   Secure IP flow is supported for both VHDL and Verilog. Mixed mode simulation is not required if the design and test bench are both in VHDL. The Verilog netlist is only required for passing the design from the synthesis to compile step in Libero. Post-synthesis and other simulation steps still use VHDL netlist if the Preferred input HDL type is VHDL at Project Creation.

2. Is Microsemi Block Flow Supported in Secure IP Flow?

   No. Block Flow is not supported for Encrypt IP and Secure IP flow.

3. Are parameters/generics supported?

   Yes. Secure IP flow works on an Encrypted IP with parameters or generic definitions.

   However, leaving top level parameters/generics and ports unencrypted makes the RTL easier to integrate.

   Refer to the VHDL example in this document, which has a generic definition.

4. Which Versions of Perl and OpenSSL are required for encryptP1735.pl script?

   Any version of OpenSSL/Perl can be used for the script to execute.

5. Installing OpenSSL?

   OpenSSL is Open-Source Software. Most Linux Installations have OpenSSL pre-installed. Most Cygwin installations on Windows also have the OpenSSL Package, which can be installed.

   For Windows, you must install OpenSSL.exe.

   You can download a version of OpenSSL from https://code.google.com/p/openssl-for-windows/downloads/detail?name=openssl-0.9.8k_X64.zip. Once you install OpenSSL on Windows, you need to set the PATH Environment Variable to <openssl_installation_dir>\bin for the EncryptP1735.pl to work.

6. Can we import an encrypted Verilog core into a VHDL design, and vice versa?

   Yes. You can import an Encrypted Verilog (or VHDL) module in a VHDL (or Verilog) Design.

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world, **650.318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

For Microsemi SoC Products Support, visit http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office.

Visit About Us for sales office listings and corporate contacts.

Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

5-02-00533-4/03.20