## Synplify Pro ME L201609MSP1-5

## **Release Notes**

2/2018





Microsemi Corporate Headquarters One Enterprise, Aliso Viejo, CA 92656 USA Within the USA: +1 (800) 713-4113 Outside the USA: +1 (949) 380-6100 Fax: +1 (949) 215-4996 Email: sales.support@microsemi.com www.microsemi.com

©2018 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners. Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

#### About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center 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, California, and has approximately 4,800 employees globally. Learn more at <u>www.microsemi.com</u>.

51300194-1/2.18



## **Revision History**

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

#### **Revision 1.0**

Revision 1.0 is the first publication of this document.



## Contents

	Revisio	n 1.0
1	Insta	llation5
2	Fami	lies Supported6
3	Enha	ncements7
	3.1	RTG4 - Disable safe implementation for FSMs7
	3.2	RTG4 - Write Byte-Enable Support for RAM7
	Exam	ple 1: RTL coding style for Single port RAM with write byte-enables7
	Exam	nple 2: RTL coding style for Two- port RAM with write byte-enable8
Exam		pple 3: VHDL RTL coding style for Two- port RAM with write byte-enable
	3.3	RTG4 - Updated Timing Models11
	3.4	SmartFusion2, IGLOO2, RTG4 - Soft JTAG Controller feature in Identify Instrumentor11
4 Resolved Issues		lved Issues14
	4.1 blocks	SmartFusion2, IGLOO2, and RTG4 - Logical Bug with inference of pipelined wide multipliers into MATH
	4.2	Other resolved issues
5	Knov	vn Issues16
	5.1	Synplify Pro error: library fusion not found16
	5.2	High reliability option in Synplify Pro is not inferring the state machine correctly16
	5.3	Warning message: "@W: CL269 : State error detection not built"16



## 1 Installation

- The Synplify Pro ME L201609MSP1-5 version can be downloaded from
   <u>https://www.microsemi.com/products/fpga-soc/design-resources/design-software/synplify-pro-me#downloads</u>
- Change the Libero SoC Tool Profile for Synthesis (Libero SoC -> Project -> Tool Profiles -> Synthesis) and point to the location of the newly-installed Synplify Pro executable.



## 2 Families Supported

Synplify Pro ME L201609MSP1-5 supports the following families:

- Fusion
- IGLOO
- IGLOOE
- IGLOO PLUS
- IGLOO2
- ProASIC3
- ProASIC3E
- ProASIC3L
- RTG4
- SmartFusion
- SmartFusion2
- PolarFire



## 3 Enhancements

The Synplify Pro ME (L2016.09MSP1-5) version has following enhancements for SmartFusion2, IGLOO2, and RTG4 families.

#### 3.1 RTG4 - Disable safe implementation for FSMs

The following options can be used to insert safe implementation logic:

- Attribute *syn\_encoding* = safe
- Attribute *syn\_safe\_case*

Enable the options under Implementation Options -> High Reliability -> Preserve and Decode unreachable states.

Synplify Pro L-2016.09M-SP1-5 issues the following error: "Safe state machine option is not recommended for Microsemi RTG4 technology. To continue with safe state machine implementation, downgrade this error to warning".

You have the option to downgrade this error to a warning message, and Synplify Pro implements the safe logic for FSMs if these options are present.

#### 3.2 RTG4 - Write Byte-Enable Support for RAM

This feature is supported for RAMs inferred in non-low power (speed) mode.

Coding style examples for RAM Write Byte-Enable are provided below.

#### Example 1: RTL coding style for Single port RAM with write byte-enables

module ram (din, dout, addra, clk, wen1, wen2);
input [9:0] din;
input wen1;
input wen2;
input [9:0] addra;
input clk;
output reg [9:0] dout;
localparam max\_depth=1024;
localparam min\_width=10;
reg [9:0] taddra;
reg [min\_width-1:0] mem\_ram[max\_depth-1:0];
always @(posedge clk) begin
 taddra<=addra;
 if(wen1) mem\_ram[taddra][4:0]<=din[4:0];
 if(wen2) mem ram[taddra][9:5]<=din[9:5];</pre>



end always @(posedge clk) begin dout <= mem\_ram[taddra]; end endmodule

#### **Resource Usage Report:**

SLE 10 uses

Total Block RAMs (RAM1K18\_RT): 1 of 209 (0%)

Total LUTs: 0

#### Example 2: RTL coding style for Two- port RAM with write byte-enable

module ram\_wb\_wen\_2addr(din ,dout, addra, addrb, clk, wen); input [17:0] din; input [1:0] wen; input [9:0] addra; input [9:0] addrb; input clk; output reg [17:0] dout; localparam max\_depth=1024; localparam min\_width=18; reg [9:0] taddra; reg [9:0] taddrb; reg [min width-1:0] mem ram[max depth-1:0]; always @(posedge clk) begin taddra<=addra; *taddrb*<*=addrb*; if(wen[0]) mem\_ram[taddra][8:0]<=din[8:0];</pre> if(wen[1]) mem\_ram[taddra][17:9]<=din[17:9];



end always @(posedge clk) begin dout <= mem\_ram[taddrb]; end endmodule Resource Usage Report: SLE 10 uses Total Block RAMs (RAM1K18\_RT) : 1 of 209 (0%) Total LUTs: 0

#### Example 3: VHDL RTL coding style for Two- port RAM with write byte-enable

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity test\_LSRAM\_1kx16 is

port	(clk_wr	: in std_logic;	
	clk_rd	: in std_logic;	
	en1_wr	: in std_logic;	

en2\_wr : in std\_logic;

addr_wr	: in std_logic_vector(9 downto 0);
data_wr	: in std_logic_vector(15 downto 0);
addr_rd	: in std_logic_vector(9 downto 0);
data_rd	: out std_logic_vector(15 downto 0)

);

end test\_LSRAM\_1kx16;

architecture behave of test\_LSRAM\_1kx16 is

type mem\_type is array (1023 downto 0) of std\_logic\_vector(15 downto 0);

signal MEM1 : mem\_type;

signal r\_addr\_rd : std\_logic\_vector(9 downto 0);



begin

process(clk\_wr)

begin

*if rising\_edge(clk\_wr) then* 

*if* (*en1\_wr* = '1') *then* 

MEM1(CONV\_INTEGER(addr\_wr(9 downto 0)))(7 downto 0)<= data\_wr(7 downto 0);

end if;

if (en2\_wr = '1') then

MEM1(CONV\_INTEGER(addr\_wr(9 downto 0)))(15 downto 8) <= data\_wr(15 downto 8);

end if;

end if;

end process;

data\_rd <= MEM1(CONV\_INTEGER(r\_addr\_rd));</pre>

process(clk\_rd)

begin

*if rising\_edge(clk\_rd) then* 

r\_addr\_rd <= addr\_rd;

end if;

end process;

end behave;

#### **Resource Usage Report:**

SLE 0 uses

Total Block RAMs (RAM1K18\_RT) : 1 of 209 (0%)

Total LUTs: 0



#### 3.3 RTG4 - Updated Timing Models

Synplify Pro L-2016.09M-SP1-5 updates the timing models for RTG4 – cell delay, net delay models and carry-chain paths.

### 3.4 SmartFusion2, IGLOO2, RTG4 - Soft JTAG Controller feature in Identify Instrumentor

- The "soft" communication interface feature is fixed.
- Users can select through the Identify Instrumentor integrated within Synplify Pro as shown in the following screenshots:

	nstrumentor - J.\syeole\Testing_lib11.8sp3\test_identify_sf2_syn-5\synthesis\synthesis_1\identify.idc]	- 0 >
	n Analysis HDL-Analyst Instrumentor Options Window Web Help	
	19 49 (24)   (24)   (25)   (2	New HDL Analyst (Be
ICE     ICE     ICE       Instrumentations     Select:     ICE       Instrumentations     Select:     ICE	SMARTFUSION2 builtin C Create skew free instrumentation C market and a construction builtin C construction constructicon	RTL (top v)         Instrum (*)           9 module top(         (*)           10         // Inputs           11         dc CLR0 pR0,           12         dd R.           13         dd L.           14         // Outputs           15         dr of o           16);         :           17/         :           22.input         15:01 dc           23.input         (15:01 dc           26//
Added instrumentation 'sy "design_flow" is unrecogn Added instrumentation 'sy "design_flow" is unrecogn Current design is top Loading instrumentation ' Source IDC file J;/sycole Setting IICE sampler (set Instrumentor%	General       Ignore no IICE at startup       Warn if no IICE at startup         Ignore no IICE at startup       Automatically create IICE at startup if none         Show SRS-only warning at startup       Automatically create IICE at startup if none         Show notification when SRS instrumentation is possible       Automatically launch Analyst when SRS instrumentation is possible         OK       Cancel	



ontrol Panel Search		BUFD_0 (BUFD)	RTL (Top.v)	Instrumen
ichnology ommunication Interface	SmartFusion2 soft	B G DSC_0 (Top B G DSC_0 (Top P G pcie_hotreset ♦ SYSRESET_0	11 6/.05 12 6/.28 13 6/.13	VAST_N, IN, ST_IN,
IICE	•		14 // 0 15 & CR 16 & CR	
IICE Type	_EPGA Memory		17 6113	22,
Sample Clock	/pcie_hotreset_D/CLKIN		18 6713 19 6712	ST_OUT.
Sample Only Bits	0		20 61 VO 21 61 VO	
Trigger Only Bits	0		22 );	
Sample & Trigger Bits	5		23 24 // 25 // Imput	
nstrumentations Select: ▶■ synthesis_2			29 input 6 30 // Output 6 33 output 6 34 output 6 35 output 6 36 output 6 39 output 6 39 output 6 40 // Mass 41 // Mass 42 // Mass 43 vire 6 44 vire 6 45 vire 6 45 vire 6	(SND1; (SND2; (LED0; (LED1; (DST_0); (VCC1; (VCC1; (VCC2; (LED1_n); (LED1_n); (LED1_n);

• In Identify Debugger, the communication port selected in Instrumenter shows as "soft"



S Identify Debugger - [J:/syeole/Testing_lib11.8sp3/test_identify	_sf2_syn-5/synthesis/top_syn*]		- 🗆	$\times$
File Edit Debug Options Window Help				
Complex Counter Mode: vents Value: 1	Sample Mode: normal	Cross trigger mode: disabled		
Er E top_syn*		Communication settings		
🕀 🔄 other 🕀 🔄 verilog	Run	Cable type: Microsemi_BuiltinJTAG		
⊕→ ■ synthesis_1		Port settings Show chain		
		Comm check		
		Instrumentation settings		
		Device family: SmartFusion2		
		Communication port: soft		
		Skew Free: off		
•	74 Running	×		
	Connect to hard	ware		
🛐 top_syn*				
J:/syeole/Testing_lib11.8sp3/test_identify_s INFO: run -iice IICE	f2_syn-5/synthesis\$ run -	-iice {IICE}		*
INFO: Info: Attempting to connect to: S201Z7	CK8X Info: Type: FlashPro	o5 Info: ID: S201Z7CK8X Info: Connection: us	b2.0 Info: )	Re
vision: UndefRev INFO: Checking communication with the Micros	emi Builtin.TTAG cable and	the hardware		
INFO: The hardware is responding correctly	cmi_builteinoiko cabie and	i one naraware		
				<b>_</b>
				ton v

For details about this feature, refer to the Synplify Pro and Identify User Guides:

Identify -> doc -> identify\_instrumentor\_user\_guide.pdf

Identify -> doc -> identify\_debugger\_user\_guide.pdf

For using Identify Instrumentor and Identify Debugger with Libero SoC, refer to the tutorial at the following link:

https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#documents



## 4 Resolved Issues

# 4.1 SmartFusion2, IGLOO2, and RTG4 - Logical Bug with inference of pipelined wide multipliers into MATH blocks

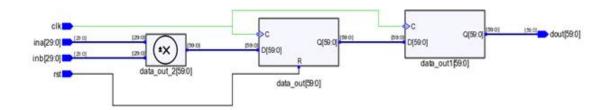
**Testcase Scenario:** 

- Two stages of registers.
- The first stage of registers has an async or sync reset, but the second stage does not.
- Width of ina > 18 or Width of inb > 18.

In the above test scenario, Synplify Pro infers MACC block by packing registers.

#### Issue:

• Reset signal was left dangling and was not connected to the MACC block.



#### 4.2 Other resolved issues

CASE	Description
<u>493642-2121758499</u>	Synplify does not issue warning
<u>493642-2275822814</u>	Synplify Pro ME L2016.09M-2 crash
<u>493642-2329166944</u>	Synplify Pro L-2016.09M-2 prompts warning Pure - Impure Function but J-2015.03M SP1-2 does not give warning
<u>493642-2352648113</u>	Internal Error in m_proasic.exe
-	For the RTG4 family, the async_globalthreshold should be set 12 by default
-	Synplify creates RGRESETs when FSM compiler option is checked for RTG4



-	Synplify Pro infers RAM1K18_RT with unsupported WMODE =01 with ECC=1
-	Synplify Pro crashes with Compiler Error for vhdl design
-	The total global count is "8" for all dies, which is incorrect.
-	Synplify Pro (embedded with Libero SoC v11.7) displays the unsupported Operating Condition for SmartFusion2/ IGLOO2/RTG4 devices
-	Need explanation for SRST_N absorption across hierarchy
-	Synplify Pro Online Help claims register initial value support
-	Synplify suboptimal 32x32 Mult
-	Forward-annotated SDF is not supported for ProASIC3 and other families and needs to be removed from doc.



### 5 Known Issues

#### 5.1 Synplify Pro error: library Fusion not found

**Issue**: There are missing lines related to the Fusion library from the location.map from Synplify Pro installation folder.

Workaround: Updated files can be patched locally upon request.

# 5.2 High reliability option in Synplify Pro is not inferring the state machine correctly

**Issue:** With High reliability option on, Synplify Pro does not infer state machines from the state machine coding in the correct way.

**Workaround:** Use the attribute syn\_state\_machine in the case statement code as below: attribute syn\_state\_machine : boolean; attribute syn\_state\_machine of state : signal is true;

#### 5.3 Warning message: "@W: CL269 : State error detection not built"

**Reason:** Refer to Synplify Pro Help for CL269 warning message.

In safe mode, the compiler generates a state error detection component for all case statements used to synthesize the state machine logic. If the component is removed, this warning is generated for you to confirm the following:

- 1. A state machine was not inferred for a particular case statement.
- 2. A state machine was inferred from a case statement, but the case statement is missing an others clause (VHDL) or default clause (Verilog).

Action: Check for the correct intended behavior. In the first condition above, verify whether the logic should not be a state machine or if the compiler was unable to extract it. For the second scenario, you may need to add an others or default clause to the source code so the compiler knows how to handle a bad state.

**Workaround:** If you believe a state machine was not inferred for a particular case statement, analyze the RTL to understand why the logic should not be a state machine. If the intended behavior is correct, ignore the warning message. Otherwise, modify the RTL so that a state machine is inferred.