PolarFire FPGA and PolarFire SoC FPGA Macro Library User Guide

Libero SoC v2025.2

Introduction (Ask a Question)

This macro library guide supports the PolarFire® FPGA family. See the Microchip website for macro guides for other families.

This guide follows a naming convention for sequential macros that is unambiguous and extensible, making it possible to understand the function of the macros by their name alone.

The first two mandatory characters of the macro name will indicate the basic macro function:

- DF—D-type flip-flop
- DL—D-type latch

The next mandatory character indicates the output polarity:

- I—output inverted (QN with bubble)
- N—output non-inverted (Q without bubble)

The next mandatory number indicates the polarity of the clock or gate:

- 1—rising edge-triggered flip-flop or transparent high latch (non-bubbled)
- 0—falling edge-triggered flip-flop or transparent low latch (bubbled)

The next two optional characters indicate the polarity of the Enable pin, if present:

- E0—active-low enable (bubbled)
- E1—active-high enable (non-bubbled)

The next two optional characters indicate the polarity of the asynchronous Preset pin, if present:

- P0—active-low asynchronous preset (bubbled)
- P1—active-high asynchronous preset (non-bubbled)

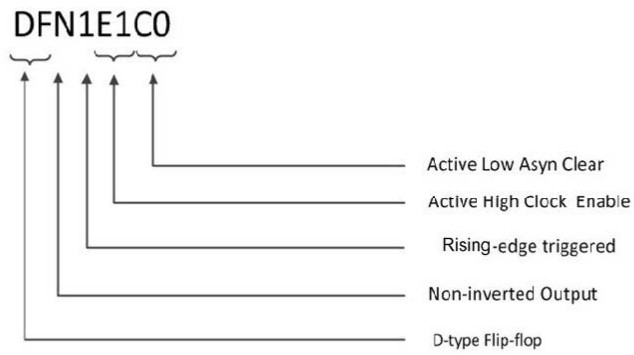
The next two optional characters indicate the polarity of the asynchronous Clear pin, if present:

- C0—active-low asynchronous clear (bubbled)
- C1—active-high asynchronous clear (non-bubbled)

All sequential and combinatorial macros (except MX4 and XOR8) use one logic element in the PolarFire FPGA family.

As an example, the macro DFN1E1C0 indicates a D-type flip-flop (DF) with a non-inverted (N) Q output, positive-edge-triggered (1), with Active-High Clock Enable (E1) and Active-Low Asychronous Clear (C0). See the following table.

Figure 1. Naming Convention



The truth table states in this User Guide are defined as follows:

State	Meaning
0	Logic “0”
1	Logic “1”
X	Do not Care (for Inputs), Unknown (for Outputs)
Z	High Impedance

User Parameter/Generics

WARNING_MSGS_ON

This feature enables you to disable the warning messages display. Default is ON ('True' in VHDL and '1' in Verilog).

Table of Contents

1. Combinatorial Logic (Ask a Question)

1.1. AND2 (Ask a Question)

2-Input AND.

Figure 1-1. AND2

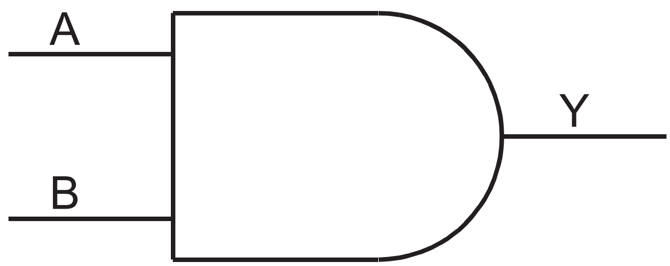


Table 1-1. AND2 I/O

Inputs	Output
A, B	Y

Table 1-2. AND2 Truth Table

A	B	Y
X	0	0
0	X	0
1	1	1

1.2. AND3 (Ask a Question)

3-Input AND.

Figure 1-2. AND3

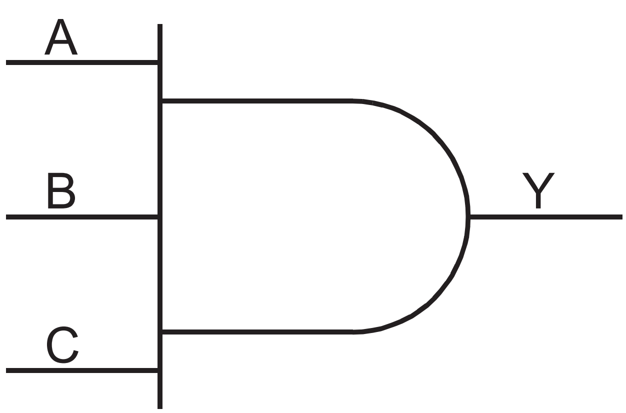


Table 1-3. AND3 I/O

Input	Output
A, B, C	Y

Table 1-4. AND3 Truth Table

A	B	C	Y
X	X	0	0
X	0	X	0
0	X	X	0
1	1	1	1

1.3. AND4 (Ask a Question)

4-Input AND.

Figure 1-3. AND4

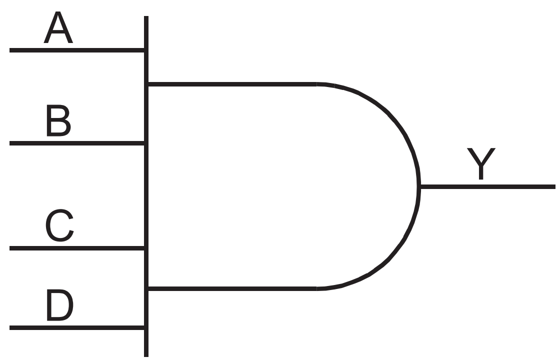


Table 1-5. AND4 I/O

Input	Output
A, B, C, D	Y

Table 1-6. AND4 Truth Table

A	B	C	D	Y
X	X	X	0	0
X	X	0	X	0
X	0	X	X	0
0	X	X	X	0
1	1	1	1	1

1.4. ARI1 (Ask a Question)

The ARI1 macro is responsible for representing all arithmetic operations in the pre-layout phase.

Figure 1-4. ARI1

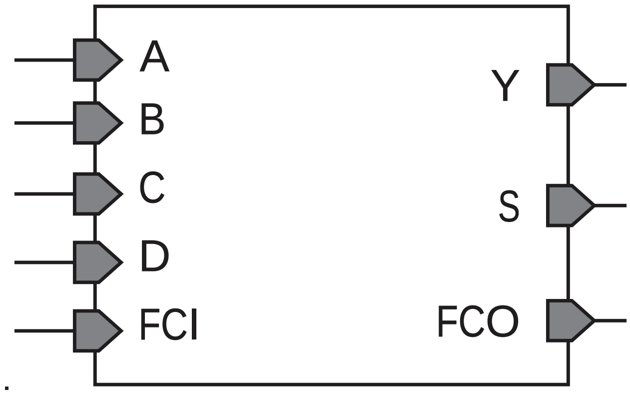


Table 1-7. ARI1 I/O

Input	Output
A, B, C, D, FCI	Y, S, FCO

The ARI1 cell has a 20 bit INIT string parameter that is used to configure its functionality. The interpretation of the 16 LSB of the INIT string is shown in the following table. F0 is the value of Y when A = 0 and F1 is the value of Y when A = 1.

Table 1-8. Interpretation of 16 LSB of the INIT String for ARI1

ADCB	Y	
0000	INIT[0]	F0
0001	INIT[1]	
0010	INIT[2]	
0011	INIT[3]	
0100	INIT[4]	
0101	INIT[5]	
0110	INIT[6]	
0111	INIT[7]	
1000	INIT[8]	F1
1001	INIT[9]	
1010	INIT[10]	
1011	INIT[11]	
1100	INIT[12]	
1101	INIT[13]	
1110	INIT[14]	
1111	INIT[15]	

Table 1-9. ARI1 Truth Table for S

Y	FCI	S
0	0	0
0	1	1
1	0	1
1	1	0

ARI1 LOGIC

The 4 MSB of the INIT string controls the output of the carry bits. The carry is generated using carry propagation and generation bits, which are evaluated according to the following tables.

Table 1-10. ARI1 INIT[17:16] String Interpretation

INIT[17]	INIT[16]	G
0	0	0
0	1	F0
1	0	1
1	1	F1

Table 1-11. ARI1 INIT[19:18] String Interpretation

INIT[19]	INIT[18]	P
0	0	0
0	1	Y
1	X	1

Table 1-12. FCO Truth Table

P	G	FCI	FCO
0	G	X	G
1	X	FCI	FCI

1.5. BUFD (Ask a Question)

Buffer.
Note: The compile optimization does not remove this macro.

Figure 1-5. BUFD

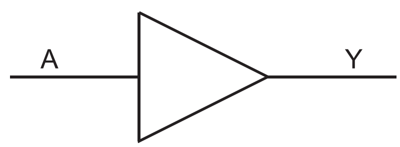


Table 1-13. BUFD I/O

Input	Output
A	Y

Table 1-14. BUFD Truth Table

A	Y
0	0
1	1

1.6. BUFF (Ask a Question)

Buffer.

Figure 1-6. BUFF

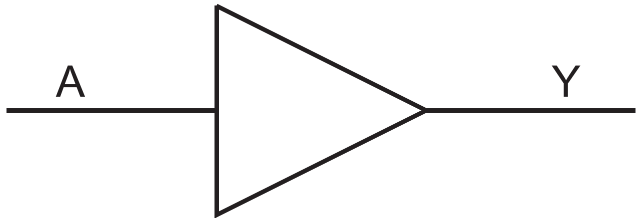


Table 1-15. BUFF I/O

Input	Output
A	Y

Table 1-16. BUFF Truth Table

A	Y
0	0
1	1

1.7. CFG1/2/3/4 and Look-Up Tables (LUTs) (Ask a Question)

CFG1, CFG2, CFG3, and CFG4 are post-layout LUTs that implement any 1-input, 2-input, 3-input, and 4-input combinational logic functions respectively. Each of the CFG1/2/3/4 macros has an INIT string parameter that determines the logic functions of the macro. The output Y is dependent on the INIT string parameter and the values of the inputs.

1.7.1. CFG2 (Ask a Question)

Post-layout macro implements any 2-input combinational logic function. Output Y is dependent on the INIT string parameter and the value of A and B. The INIT string parameter is 4 bits wide.

Figure 1-7. CFG2

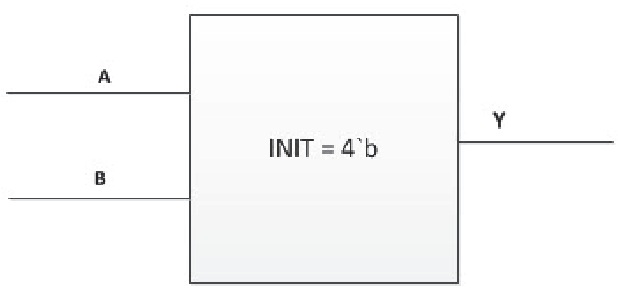


Table 1-17. CFG2 I/O

Input	Output
A,B	Y = f (INIT, A, B)

Table 1-18. CFG2 INIT String Interpretation

BA	Y
00	INIT[0]
01	INIT[1]
10	INIT[2]
11	INIT[3]

1.7.2. CFG3 (Ask a Question)

Post-layout macro used to implement any 3-input combinational logic function. Output Y is dependent on the INIT string parameter and the value of A, B, and C. The INIT string parameter is 8 bits wide.

Figure 1-8. CFG3

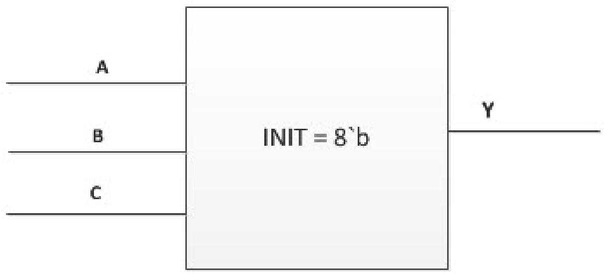


Table 1-19. CFG3 I/O

Input	Output
A, B, C	Y = f (INIT, A,B, C)

Table 1-20. CFG3 INIT String Interpretation

CBA	Y
000	INIT[0]
001	INIT[1]
010	INIT[2]
011	INIT[3]
100	INIT[4]
101	INIT[5]
110	INIT[6]
111	INIT[7]

1.7.3. CFG4 (Ask a Question)

Post-layout macro used to implement any 4-input combinational logic function. Output Y is dependent on the INIT string parameter and the value of A, B, C, and D. The INIT string parameter is 16 bits wide.

Figure 1-9. CFG4

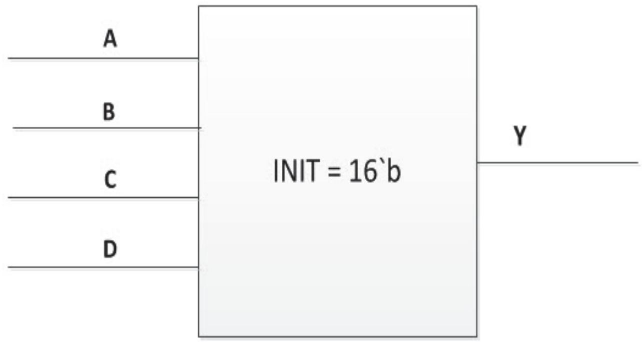


Table 1-21. CFG4 I/O

Input	Output
A, B, C, D	Y = f (INIT, A,B, C, D)

Table 1-22. CFG4 Truth Table

DCBA	Y
0000	INIT[0]
0001	INIT[1]
0010	INIT[2]
0011	INIT[3]
0100	INIT[4]
0101	INIT[5]
0110	INIT[6]
0111	INIT[7]
1000	INIT[8]
1001	INIT[9]
1010	INIT[10]
1011	INIT[11]
1100	INIT[12]
1101	INIT[13]
1110	INIT[14]
1111	INIT[15]

1.8. INV (Ask a Question)

Inverter.

Figure 1-10. INV

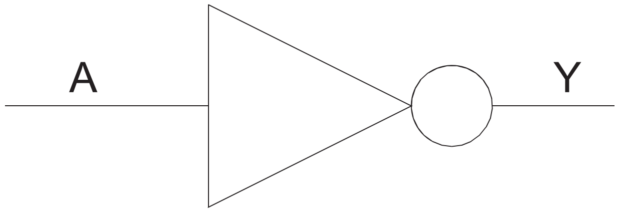


Table 1-23. INV I/O

Input	Output
A	Y

Table 1-24. INV Truth Table

A	Y
0	1
1	0

1.9. INVD (Ask a Question)

Inverter.
Note: Compile optimization does not remove this macro.

Figure 1-11. INVD

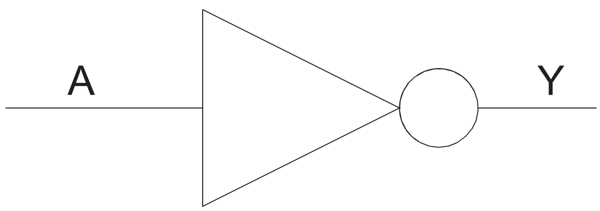


Table 1-25. INVD I/O

Input	Output
A	Y

Table 1-26. INVD Truth Table

A	Y
0	1
1	0

1.10. MX2 (Ask a Question)

2 to 1 Multiplexer.

Figure 1-12. MX2

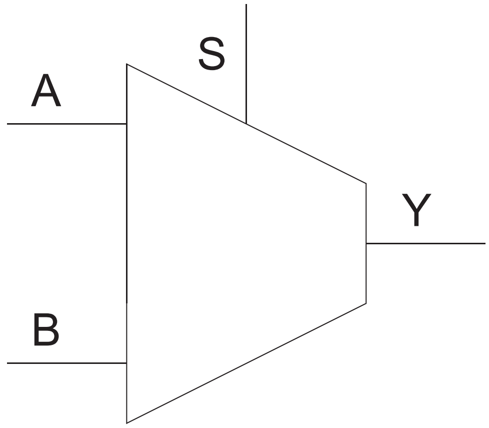


Table 1-27. MX2 I/O

Input	Output
A, B, S	Y

Table 1-28. MX2 Truth Table

A	B	S	Y
A	X	0	A
X	B	1	B

1.11. MX4 (Ask a Question)

4 to 1 Multiplexer.

This macro uses two logic modules.

Figure 1-13. MX4

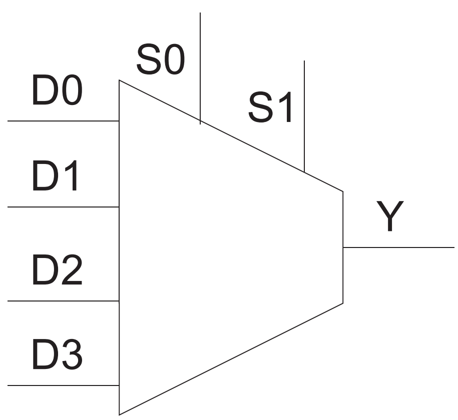


Table 1-29. MX4 I/O

Input	Output
D0, D1, D2, D3, S0, S1	Y

Table 1-30. MX4 Truth Table

D3	D2	D1	D0	S1	S0	Y
X	X	X	D0	0	0	D0
X	X	D1	X	0	1	D1
X	D2	X	X	1	0	D2
D3	X	X	X	1	1	D3

1.12. NAND2 (Ask a Question)

2-Input NAND.

Figure 1-14. NAND2

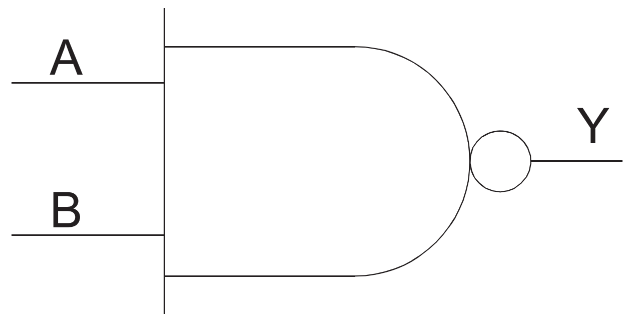


Table 1-31. NAND2 I/O

Input	Output
A, B	Y

Table 1-32. NAND2 Truth Table

A	B	Y
X	0	1
0	X	1
1	1	0

1.13. NAND3 (Ask a Question)

3-Input NAND.

Figure 1-15. NAND3

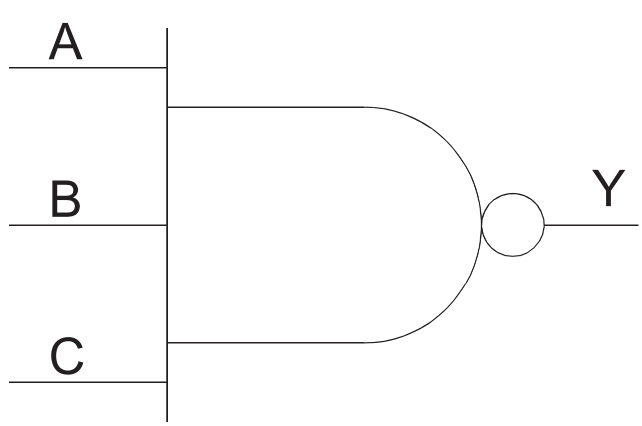


Table 1-33. NAND3 I/O

Input	Output
A, B, C	Y

Table 1-34. NAND3 Truth Table

A	B	C	Y
X	X	0	1
X	0	X	1
0	X	X	1
1	1	1	0

1.14. NAND4 (Ask a Question)

4-input NAND.

Figure 1-16. NAND4

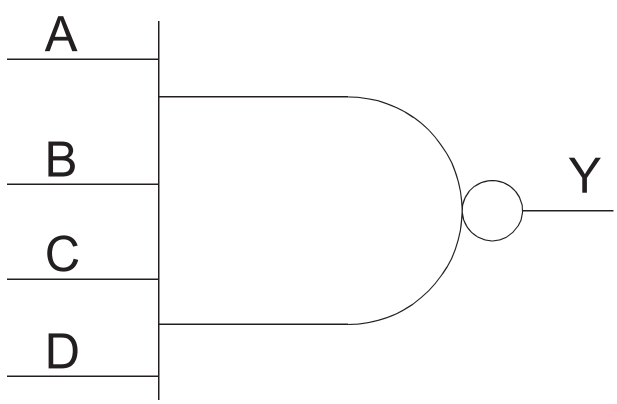


Table 1-35. NAND4 I/O

Input	Output
A, B, C, D	Y

Table 1-36. NAND4 Truth Table

A	B	C	D	Y
X	X	X	0	1
X	X	0	X	1
X	0	X	X	1
0	X	X	X	1
1	1	1	1	0

1.15. NOR2 (Ask a Question)

2-input NOR.

Figure 1-17. NOR2

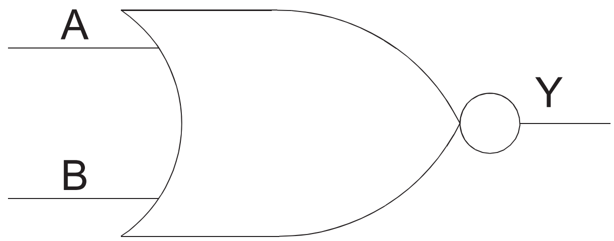


Table 1-37. NOR2 I/O

Input	Output
A, B	Y

Table 1-38. NOR2 Truth Table

A	B	Y
0	0	1
X	1	0
1	X	0

1.16. NOR3 (Ask a Question)

3-input NOR.

Figure 1-18. NOR3

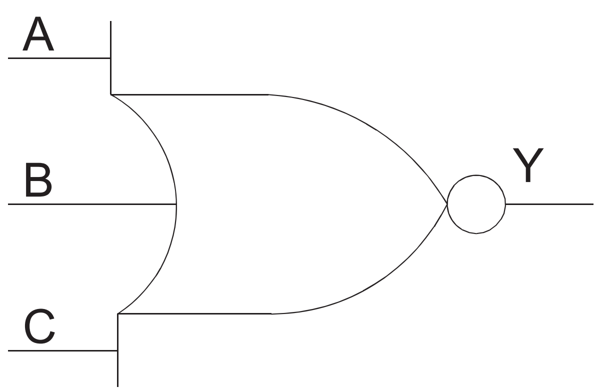


Table 1-39. NOR3 I/O

Input	Output
A, B, C	Y

Table 1-40. NOR3 Truth Table

A	B	C	Y
0	0	0	1
X	X	1	0
X	1	X	0
1	X	X	0

1.17. NOR4 (Ask a Question)

4-input NOR.

Figure 1-19. NOR3

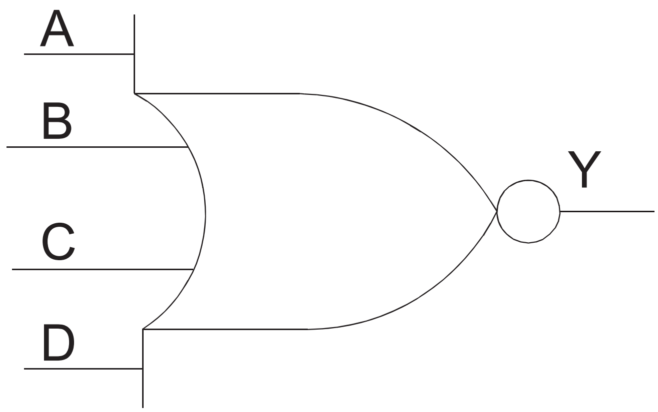


Table 1-41. NOR4 I/O

Input	Output
A, B, C, D	Y

Table 1-42. NOR4 Truth Table

A	B	C	D	Y
0	0	0	0	1
1	X	X	X	0
X	1	X	X	0
X	X	1	X	0
X	X	X	1	0

1.18. OR2 (Ask a Question)

2-input OR.

Figure 1-20. OR2

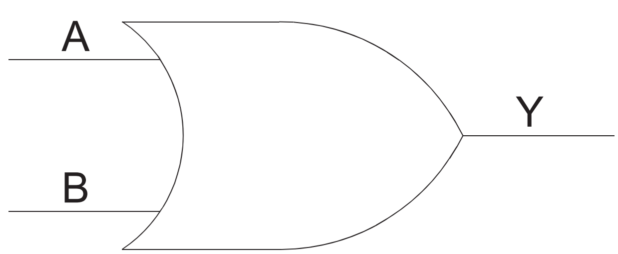


Table 1-43. OR2 I/O

Input	Output
A, B	Y

Table 1-44. OR2 Truth Table

A	B	Y
0	0	0
X	1	1

Table 1-44. OR2 Truth Table (continued)

A	B	Y
1	X	1

1.19. OR3 (Ask a Question)

3-input OR.

Figure 1-21. OR3

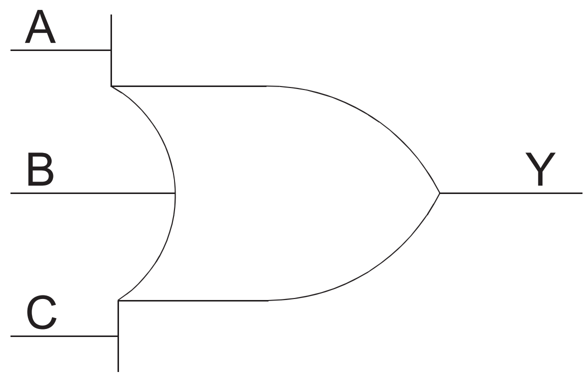


Table 1-45. OR3 I/O

Input	Output
A, B, C	Y

Table 1-46. OR3 Truth Table

A	B	C	Y
0	0	0	0
X	X	1	1
X	1	X	1
1	X	X	1

1.20. OR4 (Ask a Question)

4-input OR.

Figure 1-22. OR4

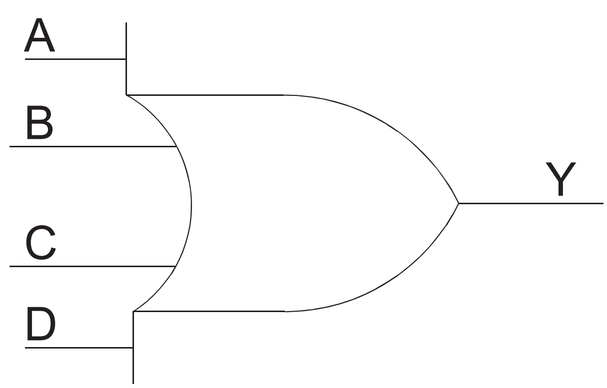


Table 1-47. OR4 I/O

Input	Output
A, B, C, D	Y

Table 1-48. OR4 Truth Table

A	B	C	D	Y
0	0	0	0	0
1	X	X	X	1
X	1	X	X	1
X	X	1	X	1
X	X	X	1	1

1.21. XOR2 (Ask a Question)

2-input XOR.

Figure 1-23. XOR2

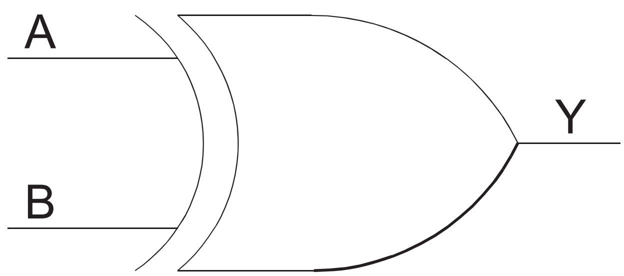


Table 1-49. XOR2 I/O

Input	Output
A, B	Y

Table 1-50. XOR2 Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

1.22. XOR3 (Ask a Question)

3-input XOR.

Figure 1-24. XOR3

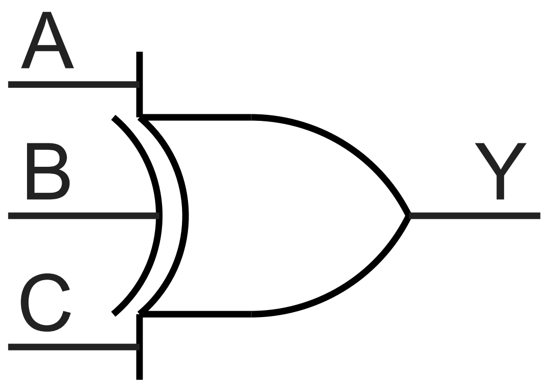


Table 1-51. XOR3 I/O

Input	Output
A, B, C	Y

Table 1-52. XOR3 Truth Table

A	B	C	Y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

1.23. XOR4 (Ask a Question)

4-input XOR.

Figure 1-25. XOR4

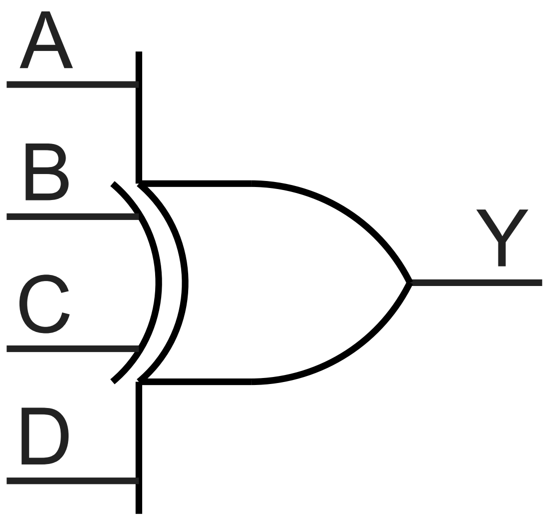


Table 1-53. XOR4 I/O

Input	Output
A, B, C, D	Y

Table 1-54. XOR4 Truth Table

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

1.24. XOR8 (Ask a Question)

8-input XOR.

This macro uses two logic modules.

Figure 1-26. XOR8

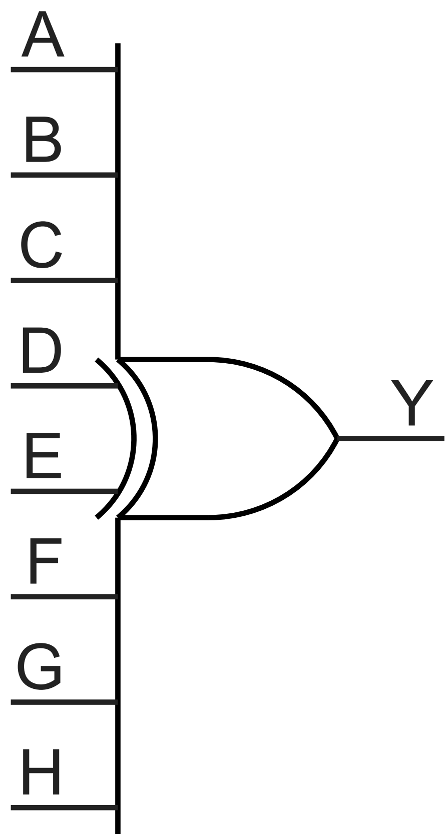


Table 1-55. XOR8 I/O

Input	Output
A, B, C, D, E, F, G, H	Y

If you have an odd number of inputs that are High, the output is High (1).

If you have an even number of inputs that are High, the output is Low (0).

For example:

Table 1-56. XOR8 Truth Table

A	B	C	D	E	F	G	H	Y
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1	0

2. Sequential Logic (Ask a Question)

Important: In SmartFusion 2 and IGLOO 2 devices, flip-flops do not power up in a known state. If no effective reset or set signal is applied to a flip-flop, the output state is considered indeterminate. This means the flip-flop could power up in either a '0' or '1' state, or even in a metastable state. It is recommended to initialize flip-flops to a known state using a reset signal at power-up.

2.1. DFN1 (Ask a Question)

D-Type Flip-Flop.

Figure 2-1. DFN1

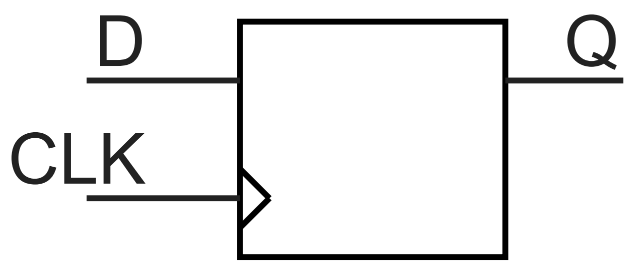


Table 2-1. DFN1 I/O

Input	Output
D, CLK	Q

Table 2-2. DFN1 Truth Table

CLK	D	Q_{n+1}
not Rising	X	Q_n
—	D	D

2.2. DFN1C0 (Ask a Question)

D-Type Flip-Flop with active-low Clear.

Figure 2-2. DFN1C0

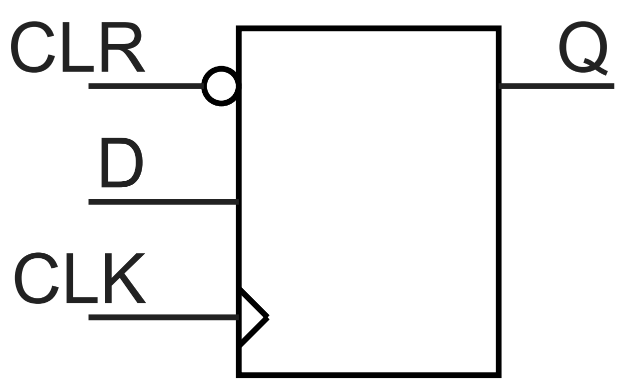


Table 2-3. DFN1C0 I/O

Input	Output
D, CLK, CLR	Q

Table 2-4. DFN1C0 Truth Table

CLR	CLK	D	Q_{n+1}
0	X	X	0
1	not Rising	X	Q_n
1	—	D	D

2.3. DFN1E1 (Ask a Question)

D-Type Flip-Flop with active high Enable.

Figure 2-3. DFN1E1

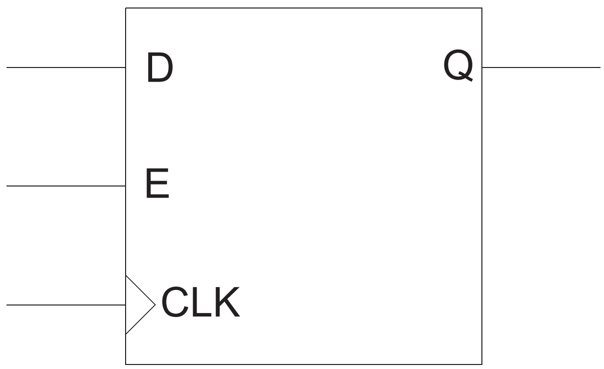


Table 2-5. DFN1E1 I/O

Input	Output
D, E, CLK	Q

Table 2-6. DFN1E1 Truth Table

E	CLK	D	Q_{n+1}
0	X	X	Q_n
1	not Rising	X	Q_n
1	—	D	D

2.4. DFN1E1C0 (Ask a Question)

D-Type Flip-Flop, with active-high Enable and active-low Clear.

Figure 2-4. DFN1E1C0

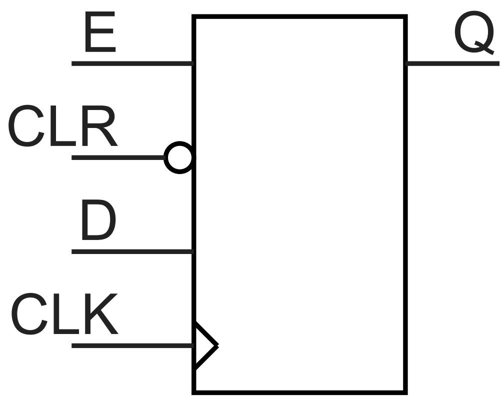


Table 2-7. DFN1E1C0 I/O

Input	Output
CLR, D, E, CLK	Q

Table 2-8. DFN1E1C0 Truth Table

CLR	E	CLK	D	Q_{n+1}
0	X	X	X	0
1	0	X	X	Q_n
1	1	not Rising	X	Q_n
1	1	—	D	D

2.5. DFN1E1P0 (Ask a Question)

D-Type Flip-Flop with active-high Enable and active-low Preset.

Figure 2-5. DFN1E1P0

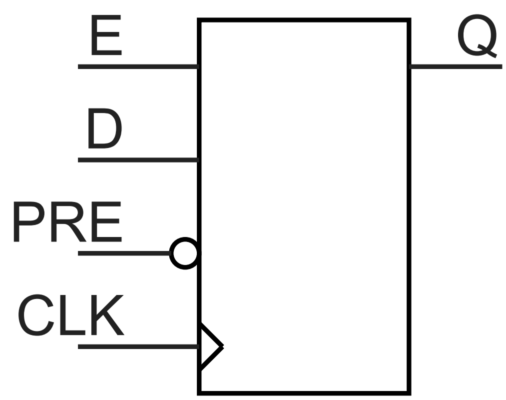


Table 2-9. DFN1E1P0 I/O

Input	Output
D, E, PRE, CLK	Q

Table 2-10. DFN1E1P0 Truth Table

PRE	E	CLK	D	Q_{n+1}
0	X	X	X	1
1	0	X	X	Q_n
1	1	not Rising	X	Q_n
1	1	—	D	D

2.6. DLN1 (Ask a Question)

Data Latch.

Figure 2-6. DLN1

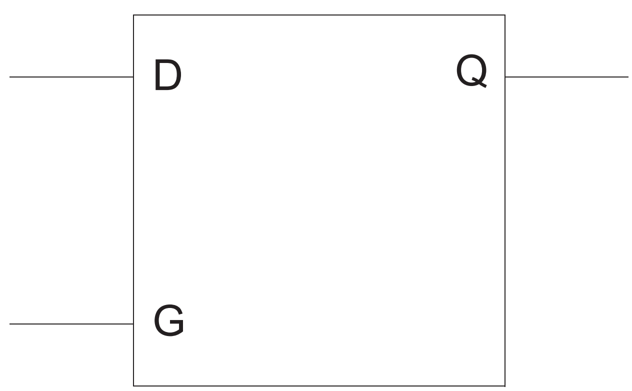


Table 2-11. DLN1 I/O

Input	Output
D, G	Q

Table 2-12. DLN1 Truth Table

G	D	Q
0	X	Q
1	D	D

2.7. DLN1C0 (Ask a Question)

Data Latch with active-low Clear.

Figure 2-7. DLN1C0

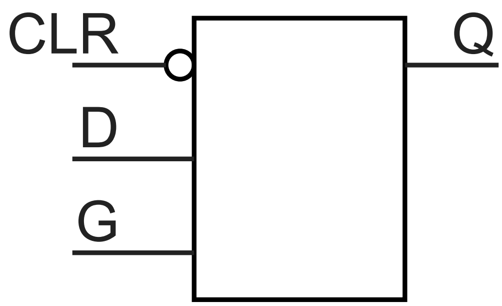


Table 2-13. I/O

Input	Output
CLR, D, G	Q

Table 2-14. Truth Table

CLR	G	D	Q
0	X	X	0
1	0	X	Q
1	1	D	D

2.8. DLN1P0 (Ask a Question)

Data Latch with active-low Preset.

Figure 2-8. DLN1P0

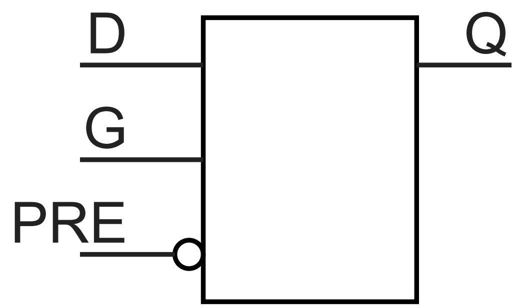


Table 2-15. DLN1C0 I/O

Input	Output
D, G, PRE	Q

Table 2-16. DLN1C0 Truth Table

PRE	G	D	Q
0	X	X	1
1	0	X	Q
1	1	D	D

2.9. SLE (Ask a Question)

Sequential Logic Element.

Figure 2-9. SLE

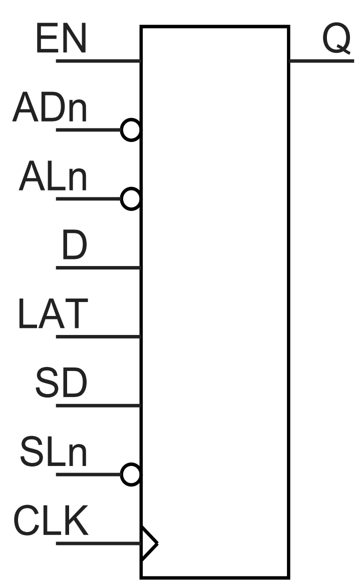


Table 2-17. SLE I/O

Input		Output
Name	Function	Q
D	Data input	
CLK	Clock input	
EN	Active-High CLK enable	
ALn	Asynchronous Load. This active-Low signal either sets the register or clears the register depending on the value of ADn.	
ADn[1]	Static asynchronous load data. When ALn is active, Q goes to the complement of ADn.	
SLn	Synchronous load. This active-Low signal either sets the register or clears the register depending on the value of SD, at the rising edge of clock.	
SD[1]	Static synchronous load data. When SLn is active (that is, low), Q goes to the value of SD at the rising edge of CLK.	
LAT[1]	Active-High Latch Enable. This signal enables latch mode when high and register mode when low.	

1. ADn, SD, and LAT are static signals defined at design time and need to be tied to 0 or 1.

Table 2-18. SLE Truth Table

ALn	ADn	LAT	CLK	EN	SLn	SD	D	Q_{n+1}
0	ADn	X	X	X	X	X	X	!ADn
1	X	0	Not rising	X	X	X	X	Qn
1	X	0	—	0	X	X	X	Qn
1	X	0	—	1	0	SD	X	SD
1	X	0	—	1	1	X	D	D
1	X	1	0	X	X	X	X	Qn
1	X	1	1	0	X	X	X	Qn
1	X	1	1	1	0	SD	X	SD
1	X	1	1	1	1	X	D	D

2.10. SLE_DEBUG (Ask a Question)

The SLE_DEBUG Macro is used to communicate with SmartDebug. The SLE_DEBUG mechanism gives ability to run synthesis while preserving a set of registers. It provides the ability to identify, rename, and classify registers for SmartDebug.

2.11. SLE_INIT (Ask a Question)

Refer to SLE macro for more details.

3. I/O (Ask a Question)

3.1. BIBUF (Ask a Question)

Bidirectional Buffer.

Figure 3-1. BIBUF

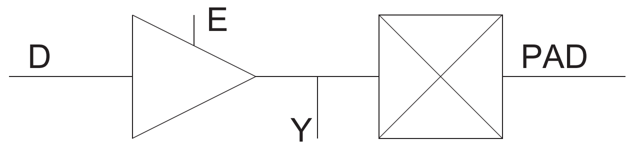


Table 3-1. BIBUF I/O

Input	Output
D, E, PAD	PAD, Y

Table 3-2. BIBUF Truth Table

MODE	E	D	PAD	Y
OUTPUT	1	D	D	D
INPUT	0	X	Z	X
INPUT	0	X	PAD	PAD

3.2. BIBUF_DIFF (Ask a Question)

Bidirectional Buffer, Differential I/O.

Figure 3-2. BIBUF_DIFF

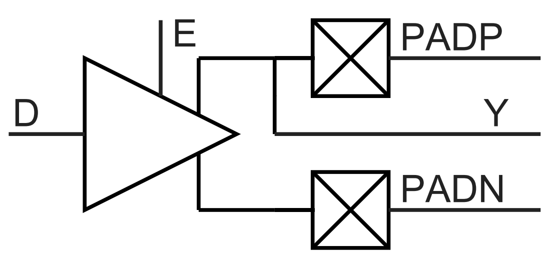


Table 3-3. BIBUF_DIFF I/O

Input	Output
D, E, PADP, PADN	PADP, PADN, Y

Table 3-4. BIBUF_DIFF Truth Table

MODE	E	D	PADP	PADN	Y
OUTPUT	1	0	0	1	0
OUTPUT	1	1	1	0	1
INPUT	0	X	Z	Z	X
INPUT	0	X	0	0	X

Table 3-4. BIBUF_DIFF Truth Table (continued)

MODE	E	D	PADP	PADN	Y
INPUT	0	X	1	1	X
INPUT	0	X	0	1	0
INPUT	0	X	1	0	1

3.3. CLKBIBUF (Ask a Question)

Bidirectional Buffer with Input to the global network.

Figure 3-3. CLKBIBUF

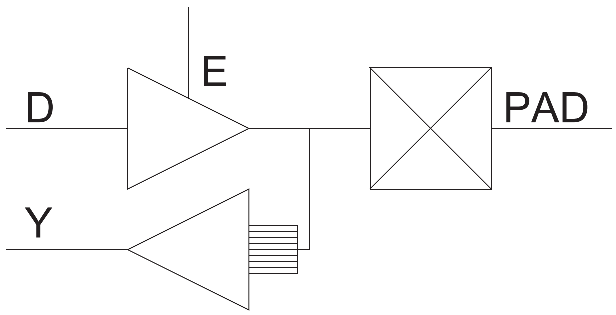


Table 3-5. CLKBIBUF I/O

Input	Output
D, E, PAD	PAD, Y

Table 3-6. CLKBIBUF Truth Table

D	E	PAD	Y
X	0	Z	X
X	0	0	0
X	0	1	1
0	1	0	0
1	1	1	1

3.4. CLKBUF (Ask a Question)

Input Buffer to the global network.

Figure 3-4. CLKBUF

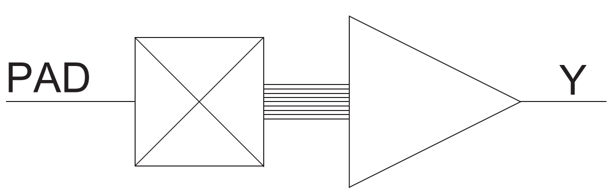


Table 3-7. CLKBUF I/O

Input	Output
PAD	Y

Table 3-8. CLKBUF Truth Table

PAD	Y
0	0
1	1

3.5. CLKBUF_DIFF (Ask a Question)

Differential I/O macro to the global network, Differential I/O.

Figure 3-5. CLKBUF_DIFF

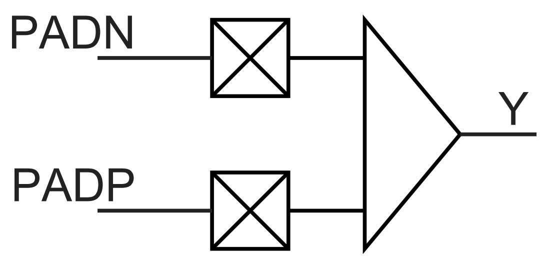


Table 3-9. INBUF_DIFF I/O

Input	Output
PADP, PADN	Y

Table 3-10. INBUF_DIFF Truth Table

PADP	PADN	Y
Z	Z	Y
0	0	X
1	1	X
0	1	0
1	0	1

3.6. GCLKBUF (Ask a Question)

Gated input I/O macro to the global network. The Enable signal can turn off the global network to save power.

Figure 3-6. GCLKBUF

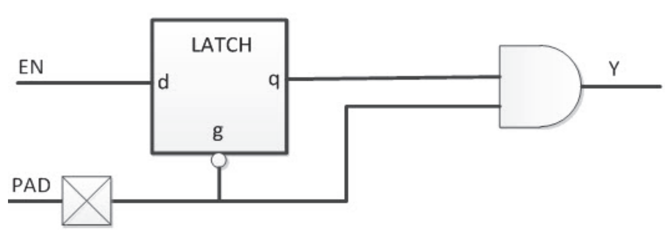


Table 3-11. GCLKBUF I/O

Input	Output
PAD, EN	Y

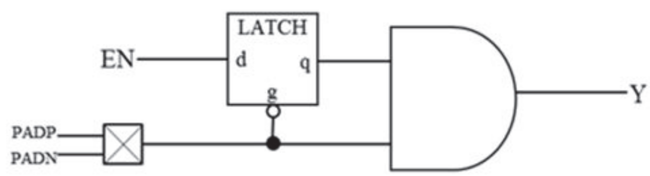
Table 3-12. GCLKBUF Truth Table

PAD	EN	q	Y
0	0	0	0
0	1	1	0
1	X	q	q
Z	X	X	X

3.7. GCLKBUF_DIFF (Ask a Question)

Gated differential I/O macro to global network; the Enable signal can be used to turn off the global network.

Figure 3-7. GCLKBUF_DIFF



Differential

Table 3-13. GCLKBUF_DIFF I/O

Input	Output
PADP, PADN, EN	Y

Table 3-14. GCLKBUF_DIFF Truth Table

PADP	PADN	EN	q	Y
0	1	0	0	0
0	1	1	1	0
1	0	X	q	q
0	0	X	X	X
1	1	X	X	X
Z	Z	X	X	X

3.8. INBUF (Ask a Question)

Input Buffer.

Figure 3-8. INBUF

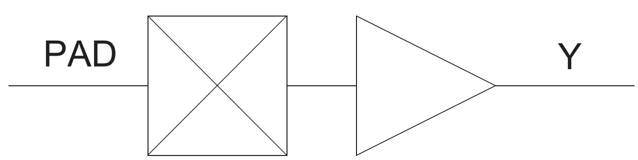


Table 3-15. INBUF I/O

Input	Output
PAD	Y

Table 3-16. INBUF Truth Table

PAD	Y
Z	X
0	0
1	1

3.9. INBUF_DIFF (Ask a Question)

Input Buffer, Differential I/O.

Figure 3-9. INBUF_DIFF

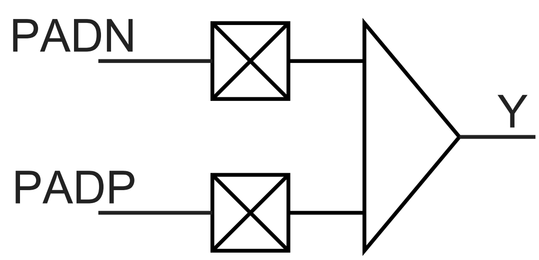


Table 3-17. INBUF_DIFF I/O

Input	Output
PADP, PADN	Y

Table 3-18. INBUF_DIFF Truth Table

PADP	PADN	Y
Z	Z	X
0	0	X
1	1	X
0	1	0
1	0	1

3.10. OUTBUF (Ask a Question)

Output buffer.

Figure 3-10. OUTBUF

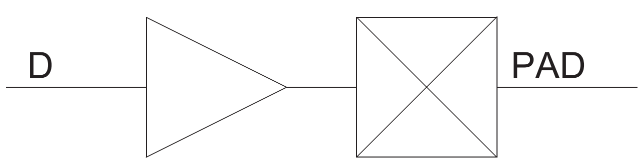


Table 3-19. OUTBUF I/O

Input	Output
D	PAD

Table 3-20. OUTBUF Truth Table

D	PAD
0	0
1	1

3.11. OUTBUF_DIFF (Ask a Question)

Output buffer, Differential I/O.

Figure 3-11. OUTBUF_DIFF

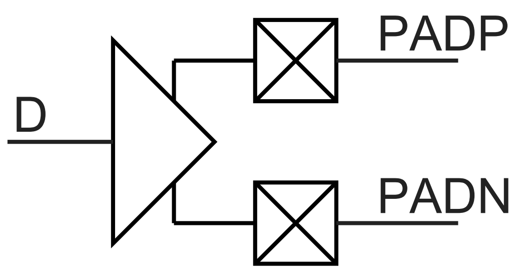


Table 3-21. OUTBUF_DIFF I/O

Input	Output
D	PADP, PADN

Table 3-22. OUTBUF_DIFF Truth Table

D	PADP	PADN
0	0	1
1	1	0

3.12. TRIBUFF (Ask a Question)

Tristate output buffer.

Figure 3-12. TRIBUFF

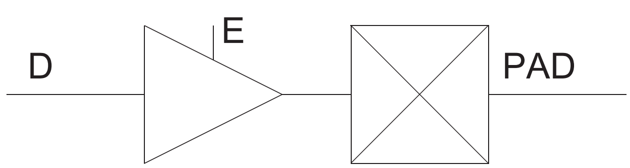


Table 3-23. TRIBUFF I/O

Input	Output
D, E	PAD

Table 3-24. TRIBUFF Truth Table

D	E	PAD
X	0	Z
D	1	D

3.13. TRIBUFF_DIFF (Ask a Question)

Tristate output buffer, Differential I/O.

Figure 3-13. TRIBUFF_DIFF

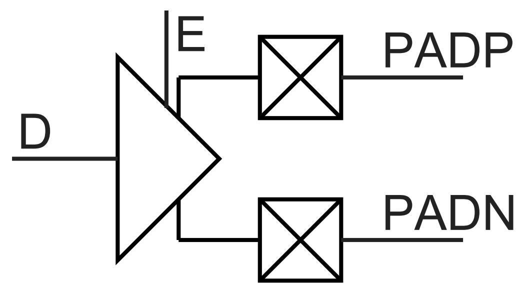


Table 3-25. TRIBUFF_DIFF I/O

Input	Output
D, E	PADP, PADN

Table 3-26. Truth Table

D	E	PADP	PADN
X	0	Z	Z
0	1	0	1
1	1	1	0

3.14. UJTAG (Ask a Question)

The UJTAG macro is a special purpose macro. It allows access to the user JTAG circuitry on board the chip.

You must instantiate a UJTAG macro in your design if you plan to make use of the user JTAG feature. The TMS, TDI, TCK, TRSTB, and TDO pins of the macro must be connected to top level ports of the design.

Figure 3-14. UJTAG

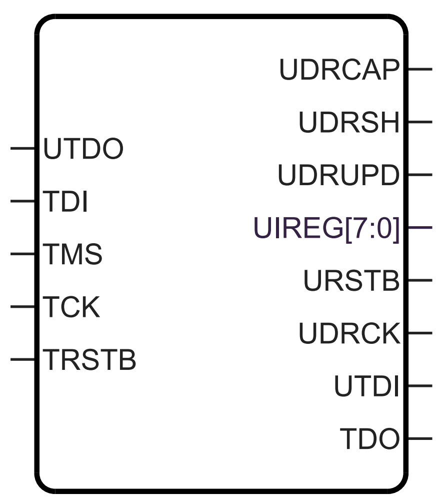


Table 3-27. Ports and Descriptions

Port	Direction	Polarity	Description
UIREG[7:0]	Output	—	This 8-bit bus carries the contents of the JTAG instruction register of each device. Instruction values 16 to 127 are not reserved and can be employed as user-defined instructions.
URSTB	Output	Low	URSTB is an Active-Low signal and is asserted when the TAP controller is in Test-Logic-Reset mode. URSTB is asserted at power-up, and a Power-on Reset signal resets the TAP controller state.
UTDI	Output	—	This port is directly connected to the TAP's TDI signal.
UTDO	Input	—	This port is the user TDO output. Inputs to the UTDO port are sent to the TAP TDO output MUX when the IR address is in user range.
UDRSH	Output	High	Active-High signal enabled in the Shift_DR TAP state.
UDRCAP	Output	High	Active-High signal enabled in the Capture_DR_TAP state.
UDRCK	Output	—	This port is directly connected to the TAP's TCK signal. **Note:** UDRCK must be connected to a global macro such as CLKINT. If this is not done, Synthesis/Compile will add it to the netlist to legalize it.
UDRUPD	Output	High	Active-High signal enabled in the Update_DR_TAP state.
TCK	Input	—	Test Clock. Serial input for JTAG boundary scan, ISP, and UJTAG. The TCK pin does not have an internal pull-up/pull-down resistor. Connect TCK to GND or +3.3V through a resistor (500-1 KΩ) placed closed to the FPGA pin to prevent totem-pole current on the input buffer and TMS from entering into an undesired state. If JTAG is not used, connect it to GND.
TDI	Input	—	Test Data In. Serial input for JTAG boundary scan. There is an internal weak pull-up resistor on the TDI pin.
TDO	Output	—	Test Data Out. Serial output for JTAG boundary scan. The TDO pin does not have an internal pull-up/pull-down resistor.
TMS	Input	—	Test mode select. The TMS pin controls the use of the IEEE®1532 boundary scan pins (TCK, TDI, TDO, and TRST). There is an internal weak pull-up resistor on the TMS pin.
TRSTB	Input	Low	Test reset. The TRSTB pin is an active-low input. It synchronously initializes (or resets) the boundary scan circuitry. There is an internal weak pull-up resistor on the TRSTB pin. To hold the JTAG in reset mode and prevent it from entering into undesired states in critical applications, connect TRSTB to GND through a 1 KΩ resistor (placed close to the FPGA pin).

3.15. UJTAG_SEC (Ask a Question)

The UJTAG_SEC macro is a special purpose macro. It allows access to the user JTAG circuitry on board the chip.

You must instantiate a UJTAG_SEC macro in your design if you plan to make use of the user JTAG feature. The TMS, TDI, TCK, TRSTB, and TDO pins of the macro must be connected to top level ports of the design.

Figure 3-15. UJTAG_SEC

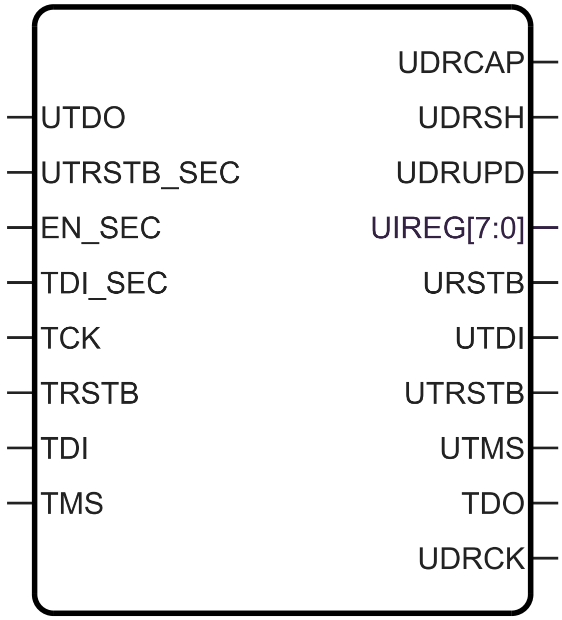


Table 3-28. Ports and Descriptions

Port	Direction	Polarity	Description
UIREG[7:0]	Output	—	This 8-bit bus carries the contents of the JTAG instruction register of each device. Instruction values 16 to 127 are not reserved and can be employed as user- defined instructions.
URSTB	Output	Low	URSTB is an Active Low signal and is asserted when the TAP controller is in Test-Logic-Reset mode. URSTB is asserted at power-up, and a Power-on Reset signal resets the TAP controller state.
UTDI	Output	—	This port is directly connected to the TAP's TDI signal.
UTDO	Input	—	This port is the user TDO output. Inputs to the UTDO port are sent to the TAP TDO output MUX when the IR addess is in user range.
UDRSH	Output	High	Active High signal enabled in the Shift_DR TAP state.
UDRCAP	Output	High	Active High signal enabled in the Capture_DR_TAP state.
UDRCK	Output	—	This port is directly connected to the TAP's TCK signal. **Note:** UDRCK must be connected to a global macro such as CLKINT. If this is not done, Synthesis/Compile will add it to the netlist to legalize it.
UDRUPD	Output	High	Active High signal enabled in the Update_DR_TAP state.

Table 3-28. Ports and Descriptions (continued)

Port	Direction	Polarity	Description
TCK	Input	—	Test Clock. Serial input for JTAG boundary scan, ISP, and UJTAG. The TCK pin does not have an internal pull-up/pull- down resistor. Connect TCK to GND or 3.3V through a resistor (500–1 KΩ) placed closed to the FPGA pin to prevent totem-pole current on the input buffer and TMS from entering into an undesired state. If JTAG is not used, connect it to GND.
TDI	Input	—	Test Data In. Serial input for JTAG boundary scan. There is an internal weak pull-up resistor on the TDI pin.
TDO	Output	—	Test Data Out. Serial output for JTAG boundary scan. The TDO pin does not have an internal pull-up/pull- down resistor.
TMS	Input	—	Test mode select. The TMS pin controls the use of the IEEE® 1532 boundary scan pins (TCK, TDI, TDO, and TRST). There is an internal weak pull- up resistor on the TMS pin.
TRSTB	Input	Low	Test reset. The TRSTB pin is an active-low input. It synchronously initializes (or resets) the boundary scan circuitry. There is an internal weak pull-up resistor on the TRSTB pin. To hold the JTAG in reset mode and prevent it from entering into undesired states in critical applications, connect TRSTB to GND through a 1 KΩ resistor (placed close to the FPGA pin).
EN_SEC	Input	High	Enable Security. Enables the user design to override the external TDI and TRSTB input to the TAP. Need to tie LOW in the design when not used.
TDI_SEC	Input	—	TDI Security override. Overrides the external TDI input to the TAP when SEC_EN is HIGH.
TRSTB_SEC	Input	Low	TRSTB Security override. Overrides the external TRSTB input to the TAP when SEC_EN is HIGH.

4. Clocking (Ask a Question)

4.1. CLKBIBUF (Ask a Question)

Bidirectional Buffer with Input to the global network.

Figure 4-1. CLKBIBUF

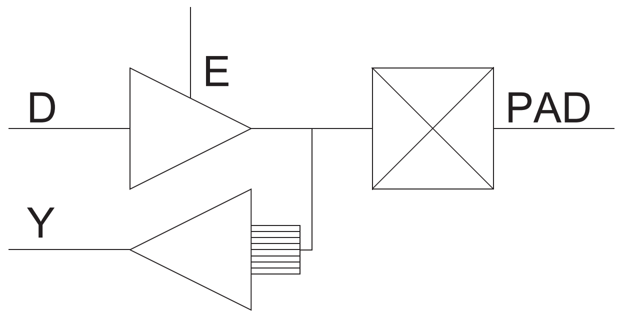


Table 4-1. CLKBIBUF I/O

Input	Output
D, E, PAD	PAD, Y

Table 4-2. CLKBIBUF Truth Table

D	E	PAD	Y
X	0	Z	X
X	0	0	0
X	0	1	1
0	1	0	0
1	1	1	1

4.2. CLKBUF (Ask a Question)

Input Buffer to the global network.

Figure 4-2. CLKBUF

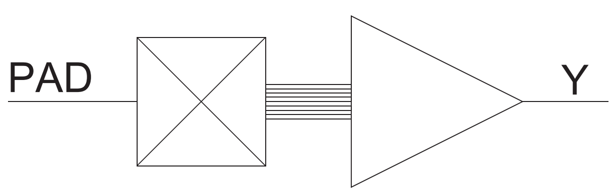


Table 4-3. CLKBUF I/O

Input	Output
PAD	Y

Table 4-4. CLKBUF Truth Table

PAD	Y
0	0
1	1

4.3. CLKBUF_DIFF (Ask a Question)

Differential I/O macro to the global network, Differential I/O.

Figure 4-3. CLKBUF_DIFF

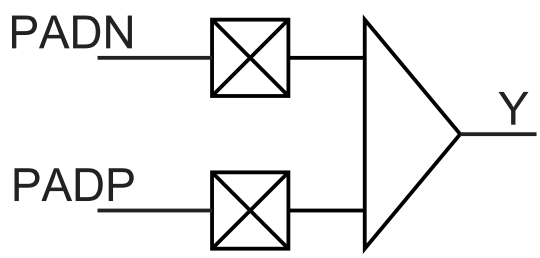


Table 4-5. INBUF_DIFF I/O

Input	Output
PADP, PADN	Y

Table 4-6. INBUF_DIFF Truth Table

PADP	PADN	Y
Z	Z	Y
0	0	X
1	1	X
0	1	0
1	0	1

4.4. CLKINT (Ask a Question)

This macro routes an internal fabric signal to the global network.

Figure 4-4. CLKINT

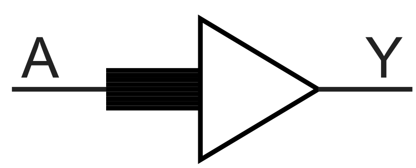


Table 4-7. CLKINT I/O

Input	Output
A	Y

Table 4-8. CLKINT Truth Table

A	Y
0	0
1	1

4.5. CLKINT_PRESERVE (Ask a Question)

This Macro routes an internal fabric signal to the global network. It has the same functionality as CLKINT except that this clock always stays on the global clock network and will not be demoted during design implementation.

Figure 4-5. CLKINT_PRESERVE

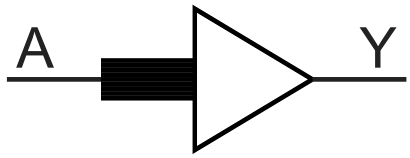


Table 4-9. CLKINT_PRESERVE I/O

Input	Output
A	Y

Table 4-10. CLKINT_PRESERVE Truth Table

A	Y
0	0
1	1

4.6. GCLKBUF (Ask a Question)

Gated input I/O macro to the global network. The Enable signal can turn off the global network to save power.

Figure 4-6. GCLKBUF

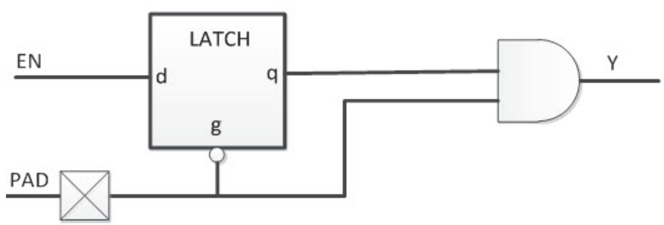


Table 4-11. GCLKBUF I/O

Input	Output
PAD, EN	Y

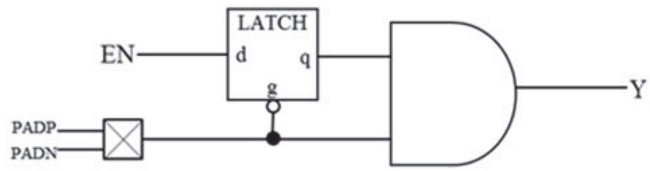
Table 4-12. GCLKBUF Truth Table

PAD	EN	q	Y
0	0	0	0
0	1	1	0
1	X	q	q
Z	X	X	X

4.7. GCLKBUF_DIFF (Ask a Question)

Gated differential I/O macro to global network; the Enable signal can be used to turn off the global network.

Figure 4-7. GCLKBUF_DIFF



Differential

Table 4-13. GCLKBUF_DIFF I/O

Input	Output
PADP, PADN, EN	Y

Table 4-14. GCLKBUF_DIFF Truth Table

PADP	PADN	EN	q	Y
0	1	0	0	0
0	1	1	1	0
1	0	X	q	q
0	0	X	X	X
1	1	X	X	X
Z	Z	X	X	X

4.8. GCLKINT (Ask a Question)

Gated macro used to route an internal fabric signal to the global network. The Enable signal can be used to turn off the global network to save power.

Figure 4-8. GCLKINT

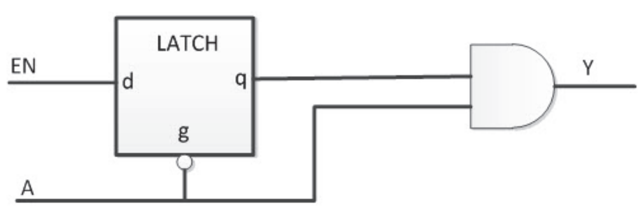


Table 4-15. GCLKINT I/O

Input	Output
A, EN	Y

Table 4-16. GCLKINT Truth Table

A	EN	q (Internal Signal)	Y
0	0	0	0
0	1	1	0
1	X	q	q

4.9. RCLKINT (Ask a Question)

Macro used to route an internal fabric signal to a row global buffer, thus creating a local clock.

Figure 4-9. RCLKINT

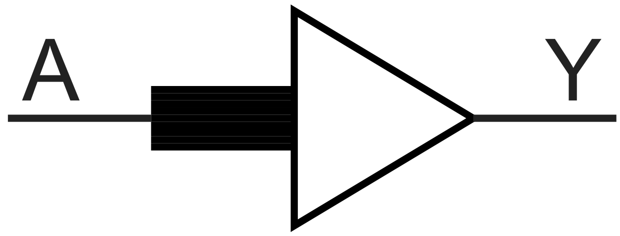


Table 4-17. RCLKINT I/O

Input	Output
A	Y

Table 4-18. RCLKINT Truth Table

A	Y
0	0
1	1

4.10. RGCLKINT (Ask a Question)

Gated macro used to route an internal fabric signal to a row global buffer, thus creating a local clock. The Enable signal can be used to turn off the local clock to save power.

Figure 4-10. RGCLKINT

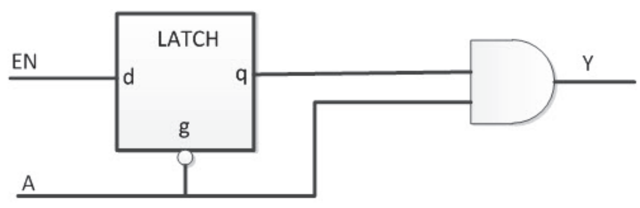


Table 4-19. RGCLKINT I/O

Input	Output
A, EN	Y

Table 4-20. RGCLKINT Truth Table

A	EN	q (Internal Signal)	Y
0	0	0	0
0	1	1	0
1	X	q	q

5. Special (Ask a Question)

5.1. FCEND_BUFF (Ask a Question)

Buffer, driven by the FCO pin of the last macro in the Carry-Chain.

Figure 5-1. FCEND_BUFF

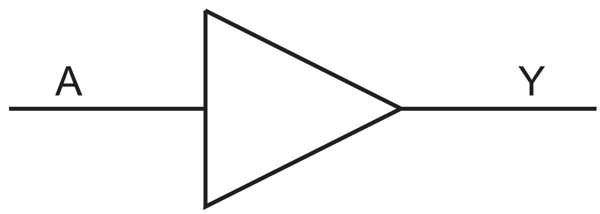


Table 5-1. FCEND_BUFF I/O

Input	Output
A	Y

Table 5-2. FCEND_BUFF Truth Table

A	Y
0	0
1	1

5.2. FCINIT_BUFF (Ask a Question)

Buffer, used to initialize the FCI pin of the first macro in the Carry-Chain.

Figure 5-2. FCINIT_BUFF

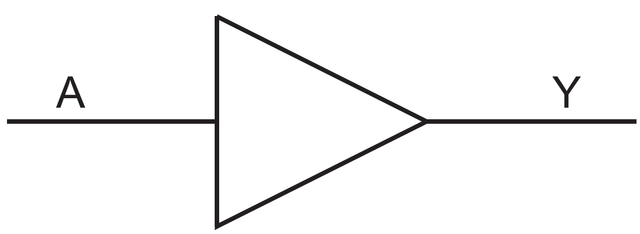


Table 5-3. FCINIT_BUFF I/O

Input	Output
A	Y

Table 5-4. FCINIT_BUFF Truth Table

A	Y
0	0
1	1

5.3. PF_SPI (Ask a Question)

The PF_SPI macro allows your design access to the dedicated System Controller SPI port, when SPI-Initiator mode is enabled, by tying both SC_SPI_EN and IO_CFG_INTF to high.

Figure 5-3. PF_SPI

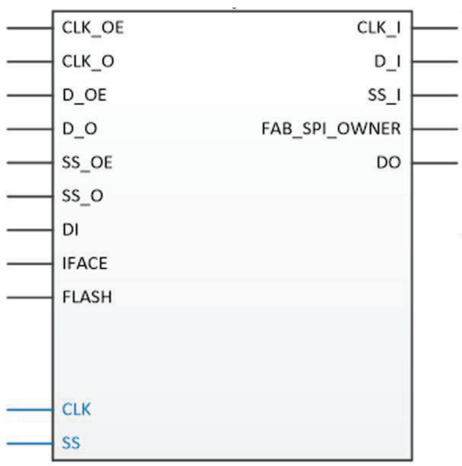


Table 5-5. Ports and Descriptions

Port	Direction	Polarity	Description
D_I	Output	—	This port is connected to the SPI DI pin.
FAB_SPI_OWNER	Output	High	Indicator to the Fabric SPI-Initiator if the SPI Port is available.
CLK_OE	Input	High	Enables the SPI CLK output.
CLK_O	Input	—	This port drives the SPI Clock pin. CLK_OE must be HIGH to drive.
D_OE	Input	High	Enables the Data output.
D_O	Input	—	This port drives the SPI DO pin. D_OE must be HIGH to drive.
SS_OE	Input	High	Enables the Target Select output.
SS_O	Input	High	This port drives the SPI Target Select (SS) pin. SS_OE must be HIGH to drive.
CLK	Output	—	SPI Clock output pin.
DI	Input	—	SPI Serial Data input pin.
DO	Output	—	SPI Serial Data output pin.
SS	Output	High	SPI Target Select output pin.
IFACE	Input	High	This port is mapped to the IO_CFG_INTF pin. This pin must be tied to high together with the SC_SPI_EN pin to enable SPI port for the fabric macro to work.
FLASH	Input	High	This port is mapped to the SC_SPI_EN pin. This pin must be tied to high together with the IO_CFG_INTF pin to enable the SPI port for fabric macro to work.

5.4. SC_STATUS (Ask a Question)

In the `SC_STATUS` macro, the `SUSPEND_EN` signal indicates that the device is in avionics mode, meaning device initialization is complete and all hardware defaults are set.

Important: This macro does not support simulation. To simulate the System Controller suspend mode, add the following pseudo-code to the simulation testbench:

- At simulation time t = 0, set `SUSPEND_EN` = 0, and `ACTIVE` = 1.
- 0.625 µs after the later of `DEVICE_INIT_DONE` = 1 or `AUTOCALIB_DONE` = 1 is asserted, set `SUSPEND_EN` = 1, and `ACTIVE` = 0 to indicate that the system controller has entered suspend mode.

Figure 5-4. SC_STATUS

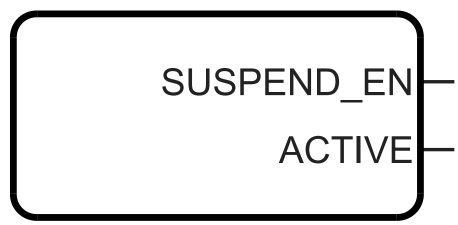


Table 5-6. SC_STATUS I/O

Value	Description
0	The device is not in avionics mode.
1	The device is in avionics mode.

For the ACTIVE signal, if SC is in suspend mode, the ACTIVE signal must not be asserted.

Table 5-7. Ports and Descriptions

Port	On Initiator			On Target			Description
	Presence	Width	Direction	Presence	Width	Direction	
SUSPEND_EN	Required	1	Output	Required	1	Input	Asserted when SC is in suspend mode.
ACTIVE	Required	1	Output	Required	1	Input	Asserted when SC is in active mode.

The following table shows the state after the system controller has finished its operations and entered suspend mode, not just when the suspend mode setting is enabled.

Table 5-8. SC_STATUS Expected Results

Port	Direction	Suspend Mode Enabled	Suspend Mode Disabled
SUSPEND_EN	OUTPUT	1	0
ACTIVE	OUTPUT	0	1

Before performing operations that require the system controller, like programming or debugging, ensure the device is not in suspend mode.

Important: Enabling suspend mode in Libero does not immediately restrict the system controller. The restriction takes effect only after the controller completes initialization and enters suspend mode.

5.5. OSC_RC160MHZ (Ask a Question)

The OSC_RC160MHZ oscillator is an RC oscillator that provides a free-running clock of 160 MHz at CLKOUT when OSC_160MHZ_ON is tied HIGH.

5.6. OSC_RC2MHZ (Ask a Question)

The OSC_RC2MHZ oscillator is an RC oscillator that provides a free-running clock of 2 MHz at CLKOUT when OSC_2MHZ_ON is tied HIGH.

5.7. LIVE_PROBE_A (Ask a Question)

This is one of the specialized probes. SmartDebug uses the dedicated and specialized probe points built in the FPGA fabric, which significantly accelerates and simplifies the debug process.

5.8. INIT (Ask a Question)

This macro does do not have any inputs. It has the following output ports, which are asserted after the specified time that is passed as parameters.

- **FABRIC_POR_N**: De-asserted when the fabric is operational.
- **PCIE_INIT_DONE**: Used by fabric logic to hold PCIe-related fabric logic in reset until the PCIe controller is initialized. PCIE_INIT_DONE is asserted after initializing the PCIe lane instances placed in the PCIe quad. If only XCVR lanes are placed in the PCIe quad, only XCVR_INIT_DONE is asserted.
- **RFU[0]**: Asserted when the XCVR block is initialized.
- **RFU[1]**: Asserted when μSRAM is initialized from sNVM.
- **RFU[2]**: Asserted when μSRAM is initialized from μPROM.
- **RFU[3]**: Asserted when μSRAM is initialized from SPI.
- **RFU[4]**: Asserted when SRAM is initialized from sNVM.
- **RFU[5]**: Asserted when SRAM is initialized from μPROM.
- **RFU[6]**: Asserted when SRAM is initialized from SPI.
- **RFU[7]**: Asserted when auto calibration is done.
- **SRAM_INIT_DONE**: Asserted when the LSRAM blocks are initialized.
- **USRAM_INIT_DONE**: Asserted when the μSRAM blocks are initialized.
- **GPIO_ACTIVE**: This signal can be used by user logic to determine if the calibration completes for each I/O banks. # denotes the bank number (0,1, 7, 8, and 9).
- **HSIO_ACTIVE**: This signal can be used to monitor if there is V_{DDI} power loss on specific I/O banks. This is an output signal from the INIT_MONITOR IP if any of the corresponding bank is selected. # denotes the bank number (0,1, 7, 8, and 9).

5.9. LANECTRL (Ask a Question)

The lane controller handles the complex operations necessary for the high-speed interfaces, such as DDR memory and CDR interfaces. To bridge the lane clock to the high-speed I/O clock, the lane controller is used to control an I/O FIFO in each IOD. The I/O FIFO interfaces with DDR memory by using the Data Q Strobe (DQS) on the lane clock. The lane controller can also delay the lane clock using a Process, Voltage, and Temperature (PVT)-calculated delay code from the DLL to provide a 90° shift. Certain I/O interfaces require a lane controller to handle the clock-domain that results with higher gear ratios. The lane controller also provides the functionality for the IOD CDR. Using the four phases from the CCC PLL, the lane controller creates eight phases and selects the proper phase for the current input condition with the input data.

5.10. LANECTRL_BYPASS (Ask a Question)

This macro puts the LANECTRL in bypass mode. It has three ports A, RESET and CLK_OUT_R.

5.11. PFSOC_SCSM (Ask a Question)

The `PFSOC_SCSM` macro allows a PolarFire SoC device, configured with the System Controller Suspend Mode (SCSM) enabled, to support MSS reboots during normal device operation. Without this macro, the System Controller:

- Will remain in suspend mode during normal operation after the initial power-up device initialization and MSS boot completion.
- Will not service subsequent MSS boot requests, even if you reset the MSS.

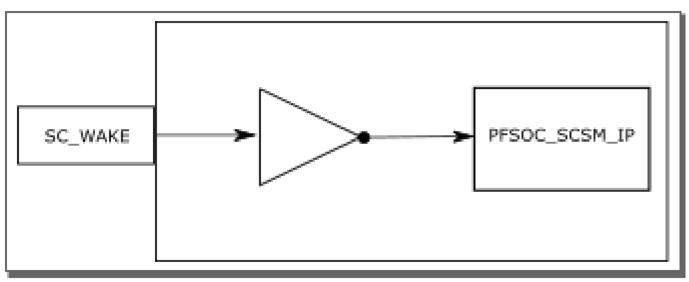
You must connect the `REBOOT_REQUESTED_M2F` output of the MSS component to the `SC_WAKE` input pin of the `PFSOC_SCSM` macro. Other connections are not permitted.

> **Important:** To enable the `REBOOT_REQUESTED_M2F` port in the MSS configurator, check the **Expose Feedback ports to Fabric** option under the **Misc** tab.

With this connection added to your FPGA fabric design, whenever the `MSS REBOOT_REQUESTED_M2F` output is asserted, the System Controller exits SCSM and processes the pending MSS reboot request. Once the MSS boots, the `REBOOT_REQUESTED_M2F` output will deassert and the System Controller returns to the suspend mode. The System Controller status can be monitored via the `SC_STATUS` macro.

> **Important:** The `PFSOC_SCSM` macro only supports the PolarFire SoC (production device) family. PolarFire SoC Engineering Silicon (ES) devices are not supported.

Figure 5-5. PFSOC_SCSM Macro



Design Rule Checks

Following are the Design Rule Checks (DRCs) for the `PFSOC_SCSM` macro.

- Check that the driver is only `MSS REBOOT_REQUESTED_M2F`, otherwise the `PFSOC_SCSM` macro will error out and stop the flow with the following message:

```
The PFSOC_SCSM primitive macro only supports being driven by the MSS component's
REBOOT_REQUESTED_M2F output. Other drivers are not supported. See the PolarFire SoC Macro
Library Guide for more information.
```

- If Suspend is not enabled, and the `PFSOC_SCSM` is instantiated, a warning is issued in the **Export Design Initialization Data and Memory Report** file.

```
The PFSOC_SCSM macro is instantiated in the design, but System Controller Suspend Mode is
not enabled. This macro is intended for use with System Controller Suspend Mode. See the
PolarFire SoC Macro Library Guide for more information.
```

- If Suspend is enabled, and the MSS is used, but the `PFSOC_SCSM` macro is not instantiated, a warning is issued in the **Export Design Initialization Data and Memory Report** file.

```
The PFSOC_SCSM macro is not instantiated even though System Controller Suspend Mode is
enabled and the MSS component is instantiated. The system controller will not be available
to process MSS reboot requests during operation. See the PolarFire SoC Macro Library Guide
for more information.
```

- The `PFSOC_SCSM` macro is visible only for the PolarFire SoC designs. This macro is required for the PolarFire SoC designs using the MSS component and for enabling System Controller Suspend Mode (SCSM). When not used in the PolarFire SoC designs, Libero ties the input to the System Controller to a static `1`.
- If an ES device or a PolarFire family is selected, the `PFSOC_SCSM` macro will error out and stop the flow in compile with the following error message, as only production PolarFire SoC devices are supported.

```
CMPPF_026: This design has one or more instances of the PFSOC_SCSM macro which is not
supported for the device in use. Before compiling, this macro(s) must be removed from the
design.
```

- For RT PolarFire SoC devices:

```
Error: The PFSOC_SCSM pin cannot be static. Refer to the PolarFire SoC Macro Library Guide
for more information."
```

6. TX_PLL (Ask a Question)

Stands for Transmit Phase-Locked Loop or Transmit PLL. Two variations of the TxPLLs embedded within the transceiver lanes are available based on protocol requirements. Both TxPLLs use ring VCO-based PLLs. Using a combination of TxPLL ranges and post-dividers produces frequencies across the entire supported range of the device. The TxPLLs includes the following types:

1. **TXPLL_SSC**: TxPLL with spread-spectrum generation (SSCG) modulation capabilities.
2. **TXPLL**: TxPLL without SSCG modulation capabilities.

Both varieties of TxPLLs employ the same half-rate fractional-N type (Frac-N) architectural design, which reduces the phase detector and frequency dividers' speed requirements. As a result, the VCO tuning range is expanded and phase noise performance is improved, all while having a substantial influence on the total power. The transmit PLL phase detector provides a valid output while driving a full-rate random data stream on both edges using the half-rate clock.

All transmit PLLs support a jitter-attenuator option. The jitter attenuator is used to track the data-rate of any noisy reference clock with a clean input reference clock to provide a 0 ppm offset from the noisy reference clock while providing a jitter-cleaned output. Each transceiver lane can select a transmit clock from the transmit PLLs that are close enough to drive their half rate clock. The PLL uses the input reference clock to generate a serial bit clock (at half the rate). The transmit PLL detects and signals a loss of lock in the event that the reference clock stops toggling or when the reference clock transitions to an incorrect frequency. There are also instances that include additional transmit PLLs, which can be used by the local transceiver quad and in a subset of lanes of adjacent quads. The output frequency of each transmit PLL is derived automatically from the reference clock frequency and the settings for the PLL multipliers. Each transmit lane can then divide this base transmit PLL rate per lane using the post-divider by 1, 2, 4, 8, or 11. The resulting frequency is half the bit rate based on the transmit half-rate architecture.

> **Example:** A 2.5 GHz clock is used for a 5 Gb/s transmit transceiver line rate. The programmable multipliers are defined and programmed by the Libero transceiver interface configurator as per the desired protocol. In addition, the transmit PLL can also provide the system clock for the FPGA logic.

7. RAM1K20 (Ask a Question)

The RAM1K20 block contains 20,480 (16,896 with ECC) memory bits and is a true dual-port memory. The RAM1K20 memory can also be configured in two-port mode. All read/write operations to the RAM1K20 memory are synchronous. To improve the read-data delay, an optional pipeline register at the output is available. In addition to the feed-through write mode option to enable immediate access to the write-data, RAM1K20 has a Read-before-write option in the dual-port mode. RAM1K20 also includes a Read-enable control for both dual-port and two-port modes. The RAM1K20 memory has two data ports, which can be independently configured in any of the following combination.

- Non-ECC Dual-Port RAM with the following configurations:
  - Any of 1Kx20, 2Kx10, 4Kx5, 8Kx2 or 16Kx1 on each port
- Non-ECC Two-Port RAM with the following configurations:
  - Any of 512x40, 1Kx20, 2Kx10, 4Kx5, 8Kx2 or 16Kx1 on each port
- ECC Two-Port RAM with the following configuration:
  - 512x33 on both ports

7.1. Functionality (Ask a Question)

The main features of the RAM1K20 memory block are as follows:

- A RAM1K20 block has 16,896 bits with ECC and 20,480 bits without ECC.
- A RAM1K20 block provides two independent data ports A and B.
- In non-ECC dual-port mode, each port can be independently configured to any of the following depth/width: 1Kx20, 2Kx10, 4Kx5, 8Kx2 or 16Kx1. There are 25 unique combinations of non-ECC dual-port aspect ratios:

1Kx20/1Kx20	1Kx20/2Kx10	1Kx20/4Kx5	1Kx16/8Kx2	1Kx16/16Kx1
2Kx10/1Kx20	2Kx10/2Kx10	2Kx10/4Kx5	2Kx8/8Kx2	2Kx8/16Kx1
4Kx5/1Kx20	4Kx5/2Kx10	4Kx5/4Kx5	4Kx4/8Kx2	4Kx4/16Kx1
8Kx2/1Kx16	8Kx2/2Kx8	8Kx2/4Kx4	8Kx2/8Kx2	8Kx2/16Kx1
16Kx1/1Kx16	16Kx1/2Kx8	16Kx1/4Kx4	16Kx1/8Kx2	16Kx1/16Kx1

- RAM1K20 also has a two-port mode. In this case, Port A will become the read port and Port B becomes the write port.
- In non-ECC two-port mode, each port can be independently configured to any of the following depth/width: 512x40, 1Kx20, 2Kx10, 4Kx5, 8Kx2 or 16Kx1. There are 36 unique combinations of non-ECC two-port aspect ratios:

512x40/512x40	512x40/1Kx20	512x40/2Kx10	512x40/4Kx5	512x32/8Kx2	512x32/16Kx1
1Kx20/512x40	1Kx20/1Kx20	1Kx20/2Kx10	1Kx20/4Kx5	1Kx16/8Kx2	1Kx16/16Kx1
2Kx10/512x40	2Kx10/1Kx20	2Kx10/2Kx10	2Kx10/4Kx5	2Kx8/8Kx2	2Kx8/16Kx1
4Kx5/512x40	4Kx5/1Kx20	4Kx5/2Kx10	4Kx5/4Kx5	4Kx4/8Kx2	4Kx4/16Kx1
8Kx2/512x32	8Kx2/1Kx16	8Kx2/2Kx8	8Kx2/4Kx4	8Kx2/8Kx2	8Kx2/16Kx1
16Kx1/512x32	16Kx1/1Kx16	16Kx1/2Kx8	16Kx1/4Kx4	16Kx1/8Kx2	16Kx1/16Kx1

- RAM1K20 has an ECC two-port mode, for which both ports have word widths equal to 33 bits. There is one unique combination of ECC two-port aspect ratio:
  512x33/512x33

- RAM1K20 performs synchronous operation for setting up the address as well as writing and reading the data.
- RAM1K20 has a Read-enable control for both dual-port and two-port modes.
- The address, data, block-port select, write-enable and read-enable inputs are registered.
- An optional pipeline register with a separate enable, synchronous-reset and asynchronous-reset is available at the read-data port to improve the clock-to-out delay.
- There is an independent clock for each port. The memory is triggered at the rising edge of the clock.
- The true dual-port mode supports an optional Read-before-write mode or a feed-through write mode, where the write-data also appears on the corresponding read-data port.
- Read from both ports at the same location is allowed.
- Read and write on the same location at the same time results in unknown data to be read. There is no collision prevention or detection. However, correct data is expected to be written into the memory.
- When ECC is enabled, each port of the RAM1K20 memory can raise flags to indicate single-bit-correct and double-bit-detect.

The following figure shows a simplified block diagram of the RAM1K20 memory block. The simplified block illustrates the two independent data ports, ECC, the read-data pipeline registers, read-before-write selection, and the feed-through multiplexors.

Figure 7-1. Simplified Block Diagram of RAM1K20

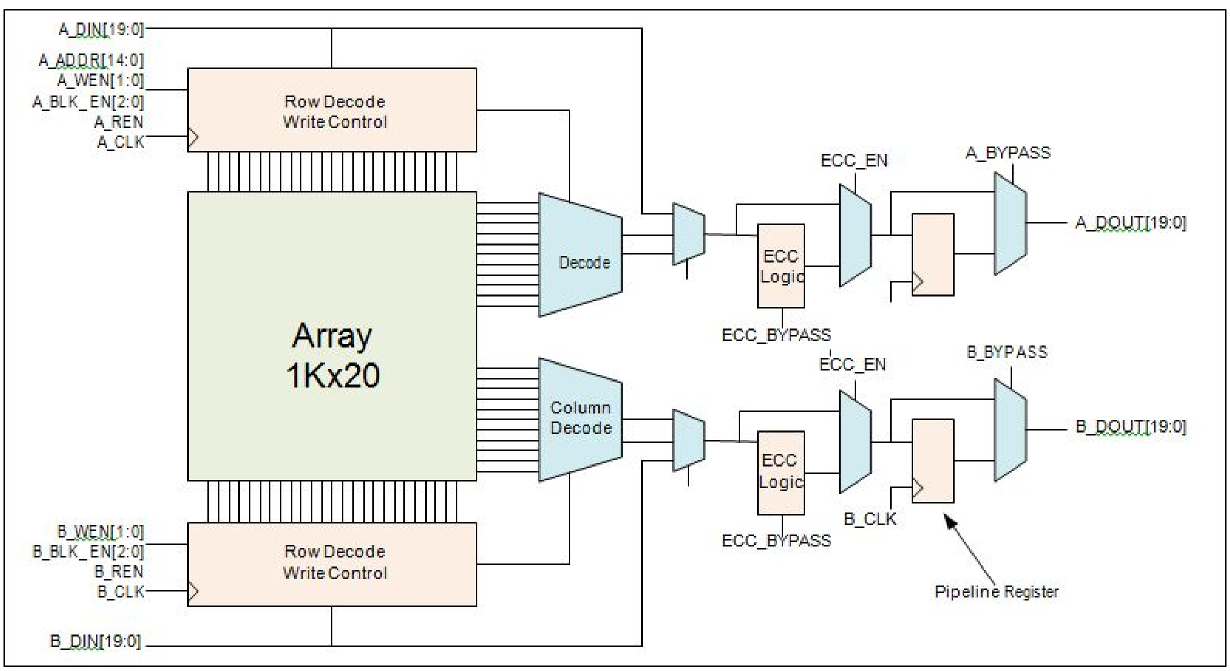


Table 7-1. Port List for RAM1K20

Pin Name	Pin Direction	Type	Description	Polarity
A_ADDR[13:0]	Input	Dynamic	Port A address	—
A_BLK_EN[2:0]	Input	Dynamic	Port A block selects	High
A_CLK	Input	Dynamic	Port A clock	Rising
A_DIN[19:0]	Input	Dynamic	Port A write-data	—
A_DOUT[19:0]	Output	Dynamic	Port A read-data	—

Table 7-1. Port List for RAM1K20 (continued)

Pin Name	Pin Direction	Type	Description	Polarity
A_WEN[1:0]	Input	Dynamic	Port A write-enables (per byte)	High
A_REN	Input	Dynamic	Port A read-enable	High
A_WIDTH[2:0]	Input	Static	Port A width/depth mode select	—
A_WMODE[1:0]	Input	Static	Port A Read-before-write and Feed-through write selects	High
A_BYPASS	Input	Static	Port A pipeline register select	Low
A_DOUT_EN	Input	Dynamic	Port A pipeline register enable	High
A_DOUT_SRST_N	Input	Dynamic	Port A pipeline register synchronous-reset	Low
A_DOUT_ARST_N	Input	Dynamic	Port A pipeline register asynchronous-reset	Low
B_ADDR[13:0]	Input	Dynamic	Port B address	—
B_BLK_EN[2:0]	Input	Dynamic	Port B block selects	High
B_CLK	Input	Dynamic	Port B clock	Rising
B_DIN[19:0]	Input	Dynamic	Port B write-data	—
B_DOUT[19:0]		Dynamic	Port B read-data	—
B_WEN[1:0]	Input	Dynamic	Port B write-enables (per byte)	High
B_REN	Input	Dynamic	Port B read-enable	High
B_WIDTH[2:0]	Input	Static	Port B width/depth mode select	—
B_WMODE[1:0]	Input	Static	Port B Read-before-write and Feed-through write selects	High
B_BYPASS	Input	Static	Port B pipeline register select	Low
B_DOUT_EN	Input	Dynamic	Port B pipeline register enable	High
B_DOUT_SRST_N	Input	Dynamic	Port B pipeline register synchronous-reset	Low
B_DOUT_ARST_N	Input	Dynamic	Port B pipeline register asynchronous-reset	Low
ECC_EN	Input	Static	Enable ECC	High
ECC_BYPASS	Input	Static	ECC pipeline register select	Low
SB_CORRECT	Output	Dynamic	Single-bit correct flag	High
DB_DETECT	Output	Dynamic	Double-bit detect flag	High
BUSY_FB	Input	Static	Lock access to FCB	High
ACCESS_BUSY	Output	Dynamic	Busy signal from FCB	High

Tip: Static inputs are defined at design time and need to be tied to 0 or 1.

A_WIDTH and B_WIDTH

The following table lists the width/depth mode selections for each port. Two-port mode is in effect when the width of at least one port is greater than 20, and A_WIDTH indicates the read width while B_WIDTH indicates the write width.

Table 7-2. Width/Depth Mode Selection

Depth x Width	A_WIDTH/B_WIDTH
16Kx1	000
8Kx2	001
4Kx4, 4Kx5	010
2Kx8, 2Kx10	011

Table 7-2. Width/Depth Mode Selection (continued)

Depth x Width	A_WIDTH/B_WIDTH
1Kx16, 1Kx20	100
512x32 (Two-port), 512x40 (Two-port), 512x33 (Two-port ECC)	101

A_WEN and B_WEN

The following table lists the write/read control signals for each port. Two-port mode is in effect when the width of at least one port is greater than 20, and read operation is always enabled.

Table 7-3. Write/Read Operation Select

Depth x Width	A_WEN/B_WEN	Result
16Kx1, 8Kx2, 4Kx5, 2Kx10	x0	Perform a read operation
	x1	Perform a write operation
1Kx16	00	Perform a read operation
	01	Write [8:5], [3:0]
	10	Write [18:15], [13:10]
	11	Write [18:15], [13:10], [8:5], [3:0]
1Kx20	00	Perform a read operation
	01	Write [9:0]
	10	Write [19:10]
	11	Write [19:0]
512x32 (Two-port write)	B_WEN[0] = 1	Write B_DIN[8:5], B_DIN[3:0]
	B_WEN[1] = 1	Write B_DIN[18:15], B_DIN[13:10]
	A_WEN[0] = 1	Write A_DIN[8:5], A_DIN[3:0]
	A_WEN[1] = 1	Write A_DIN[18:15], A_DIN[13:10]
512x40 (Two-port write)	B_WEN[0] = 1	Write B_DIN[9:0]
	B_WEN[1] = 1	Write B_DIN[19:10]
	A_WEN[0] = 1	Write A_DIN[9:0]
	A_WEN[1] = 1	Write A_DIN[19:10]
512x33 (Two-port ECC)	B_WEN[1:0] = 11	Write B_DIN[16:0]
	A_WEN[1:0] = 11	Write A_DIN[15:0]

A_ADDR and B_ADDR

The following table lists the address buses for the two ports. 14 bits are needed to address the 16K independent locations in x1 mode. In wider modes, fewer address bits are used. The required bits are MSB justified and unused LSB bits must be tied to 0. A_ADDR is synchronized by A_CLK while B_ADDR is synchronized to B_CLK. Two-port mode is in effect when the width of at least one port is greater than 20, and A_ADDR provides the read-address while B_ADDR provides the write-address.

Table 7-4. Address Bus Used and Unused Bits

Depth x Width	A_ADDR/B_ADDR	
	Used Bits	Unused Bits (must be tied to 0)
16Kx1	[13:0]	None
8Kx2	[13:1]	[0]
4Kx4, 4Kx5	[13:2]	[1:0]
2Kx8, 2Kx10	[13:3]	[2:0]

Table 7-4. Address Bus Used and Unused Bits (continued)

Depth x Width	A_ADDR/B_ADDR	
	Used Bits	Unused Bits (must be tied to 0)
1Kx16, 1Kx20	[13:4]	[3:0]
512x32 (Two-port), 512x40 (Two-port), 512x33 (Two-port ECC)	[13:5]	[4:0]

A_DIN and B_DIN

The following table lists the data input buses for the two ports. The required bits are LSB justified and unused MSB bits must be tied to 0. Two-port mode is in effect when the width of at least one port is greater than 20, and A_DIN provides the MSB of the write-data while B_DIN provides the LSB of the write-data.

Table 7-5. Data Input Buses Used and Unused Bits

Depth x Width	A_DIN/B_DIN	
	Used Bits	Unused Bits (must be tied to 0)
16Kx1	[0]	[19:1]
8Kx2	[1:0]	[19:2]
4Kx4	[3:0]	[19:4]
4Kx5	[4:0]	[19:5]
2Kx8	[8:5] is [7:4] [3:0] is [3:0]	[19:9] [4]
2Kx10	[9:0]	[19:10]
1Kx16	[18:15] is [15:12] [13:10] is [11:8] [8:5] is [7:4] [3:0] is [3:0]	[19] [14] [9] [4]
1Kx20	[19:0]	None
512x32 (Two-port write)	A_DIN[18:15] is [31:28] A_DIN[13:10] is [27:24] A_DIN[8:5] is [23:20] A_DIN[3:0] is [19:16] B_DIN[18:15] is [15:12 B_DIN[13:10] is [11:8] B_DIN[8:5] is [7:4] B_DIN[3:0] is [3:0]	A_DIN[19] A_DIN[14] A_DIN[9] A_DIN[4] B_DIN[19] B_DIN[14] B_DIN[9] B_DIN[4]
512x40 (Two-port write)	A_DIN[19:0] is [39:20] B_DIN[19:0] is [19:0]	None
512x33 (Two-port ECC)	A_DIN[15:0] is [32:17] B_DIN[16:0] is [16:0]	A_DIN[19:16] B_DIN[19:17]

A_DOUT and B_DOUT

The following table lists the data output buses for the two ports. The required bits are LSB justified. Two-port mode is in effect when the width of at least one port is greater than 20, and A_DOUT provides the MSB of the read-data while B_DOUT provides the LSB of the read-data.

Table 7-6. Data Output Buses Used and Unused Bits

Depth x Width	A_DOUT/B_DOUT	
	Used Bits	Unused Bits **(must be tied to 0)**
16Kx1	[0]	[19:1]
8Kx2	[1:0]	[19:2]
4Kx4	[3:0]	[19:4]
4Kx5	[4:0]	[19:5]
2Kx8	[8:5] is [7:4] [3:0] is [3:0]	[19:9] [4]
2Kx10	[9:0]	[19:10]
1Kx16	[18:15] is [15:12] [13:10] is [11:8] [8:5] is [7:4] [3:0] is [3:0]	[19] [14] [9] [4]
1Kx20	[19:0]	None
512x32 (Two-port write)	A_DIN[18:15] is [31:28] A_DIN[13:10] is [27:24] A_DIN[8:5] is [23:20] A_DIN[3:0] is [19:16] B_DIN[18:15] is [15:12 B_DIN[13:10] is [11:8] B_DIN[8:5] is [7:4] B_DIN[3:0] is [3:0]	A_DIN[19] A_DIN[14] A_DIN[9] A_DIN[4] B_DIN[19] B_DIN[14] B_DIN[9] B_DIN[4]
512x40 (Two-port write)	A_DOUT[19:0] is [39:20] B_DOUT[19:0] is [19:0]	None
512x33 (Two-port ECC)	A_DOUT15:0] is [32:17] B_DOUT[16:0] is [16:0]	A_DOUT[19:16] B_DOUT[19:17]

A_BLK_EN and B_BLK_EN

The following table lists the block-port select control signals for the two ports. A_BLK is synchronized by A_CLK while B_BLK is synchronized to B_CLK. Two-port mode is in effect when the width of at least one port is greater than 20, and A_BLK_EN controls the read operation while B_BLK_EN controls the write operation.

Table 7-7. Block-Port Select

Block-port Select Signal	Value	Result
A_BLK_EN[2:0]	111	Perform read or write operation on Port A, unless the width is greater than 20 and a read is performed from both ports A and B.
A_BLK_EN[2:0]	Any one bit is 0	No operation in memory from Port A. Port A read-data will be forced to 0. If the width is greater than 20, the read-data from both ports A and B will be forced to 0.
B_BLK_EN[2:0]	111	Perform read or write operation on Port B, unless the width is greater than 20 and a write is performed to both ports A and B.
B_BLK_EN[2:0]	Any one bit is 0	No operation in memory from Port B. Port B read-data will be forced to 0, unless the width is greater than 20 and write operation to both ports A and B is gated.

A_WMODE and B_WMODE

In true dual-port write mode, each port has a feed-through write or read-before-write option.

- Logic 00 = Read-data port holds the previous value.
- Logic 01 = Feed-through, that is write-data appears on the corresponding read-data port. This setting is invalid when the width of at least one port is greater than 20 and the two-port mode is in effect.
- Logic 10 = Read-before-write, that is previous content of the memory appears on the corresponding read-data port before it is overwritten. This setting is invalid when the width of at least one port is greater than 20 and the two-port mode is in effect.
  A_CLK and B_CLK

All signals in ports A and B are synchronous to the corresponding port clock. All address, data, block-port select, write-enable and read-enable inputs must be set up before the rising edge of the clock. The read or write operation begins with the rising edge. Two-port mode is in effect when the width of at least one port is greater than 20, and A_CLK provides the read clock while B_CLK provides the write clock.

A_REN and B_REN

Enables read operation from the memory on the corresponding port. Two-port read mode is in effect when the width of port A is greater than 20, and A_REN controls the read operation.

Read-data Pipeline Register Control Signals

A_BYPASS and B_BYPASS A_DOUT_EN and B_DOUT_EN A_DOUT_SRST_N and B_DOUT_SRST_N A_DOUT_ARST_N and B_DOUT_ARST_N

Two-port mode is in effect when the width of at least one port is greater than 20, and the A_DOUT register signals control both the MSB and LSB of the read-data, and the B_DOUT register signals are “don’t-cares”.

The following table describes the functionality of the control signals on the A_DOUT and B_DOUT pipeline registers.

Table 7-8. Truth Table for A_DOUT and B_DOUT Registers

ARST_N	_BYPASS	_CLK	_EN	_SRST_N	D	Q_{n+1}
0	X	X	X	X	X	0
1	0	Not rising	X	X	X	Q_n
1	0	?	0	X	X	Q_n
1	0	?	1	0	X	0
1	0	?	1	1	D	D
1	1	X	X	X	D	D

ECC_EN and ECC_BYPASS

ECC operation is only allowed in Two-port mode and the width of both ports is greater than 20.

- ECC_EN = 0: Disable ECC.
- ECC_EN = 1, ECC_BYPASS= 0: Enable ECC Pipelined.
- ECC Pipelined mode inserts an additional clock cycle to Read-data.
- In addition, Write-feed-thru and Read-before-write modes add another clock cycle to Read- data.
- ECC_EN = 1, ECC_BYPASS= 1: Enable ECC Non-pipelined.
  SB_CORRECT and DB_DETECT

Error detection and correction flags become available when ECC operation is enabled in Two-port mode and the width of both ports is greater than 20. The following table describes the functionality of the error detection and correction flags.

Table 7-9. Error Detection and Correction Flags

DB_DETECT	SB_CORRECT	Flag
0	0	No errors have been detected.
0	1	A single bit error has been detected and corrected in the data output.
1	1	Multiple bit errors have been detected, but have not been corrected.

BUSY_FB

Control signal, when 1 locks the entire RAM1K20 memory from being accessed by the FCB.

ACCESS_BUSY

This output indicates that the RAM1K20 memory is being accessed by the FCB.

8. RAM64x12 (Ask a Question)

The RAM64x12 block contains 768 memory bits and is a two-port memory providing one write port and one read port. Write operations to the RAM64x12 memory are synchronous. Read operations can be asynchronous or synchronous for setting up the address and reading out the data. Enabling synchronous operation at the read-address port improves setup timing for the read-address and its enable signals. Enabling synchronous operation at the read-data port improves clock-to-out delay. Each data port on the RAM64x12 memory is configured to a fixed configuration of 64x12.

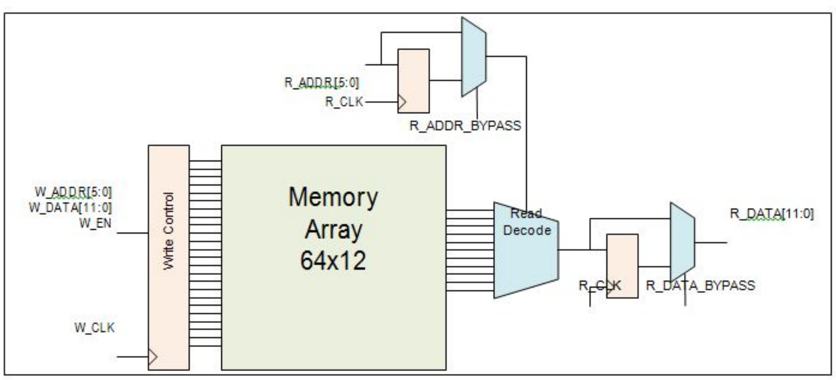
Functionality

The main features of the RAM64x12 memory block are as follows:

- There is one read-data port and one write-data port.
- Both read-data and write-data ports are configured to 64x12.
- The write operation is always synchronous. The write-address, write-data and write-enable inputs are registered.
- Setting up the read-address can be synchronous or asynchronous. The read-address registers have an independent enable, synchronous-load and asynchronous-load for synchronous mode operation, which can be bypassed for asynchronous mode operation.
- The read-data pipeline registers have an independent enable, synchronous-load and asynchronous-load for pipeline mode operation, which can be bypassed for asynchronous mode operation.
- Therefore, there are four read operation modes:
  - Synchronous read-address without read-data pipeline registers (sync-async)
  - Synchronous read-address with read-data pipeline registers (sync-sync)
  - Asynchronous read-address with read-data pipeline registers (async-sync)
  - Asynchronous read-address without read-data pipeline registers (async-async)
- There is an independent clock for each port. The memory will be triggered at the rising edge of the clock.
- Read and write on the same location at the same time results in unknown data to be read.

There is no collision prevention or detection. However, correct data is expected to be written into the memory.

Figure 8-1. Simplified Block Diagram of RAM64x12



Port List

The following table gives the port descriptions.

Table 8-1. Port List for RAM1K20

Pin Name	Pin Direction	Type	Description	Polarity
W_EN	Input	Dynamic	Write port enable	High
W_CLK	Input	Dynamic	Write clock. All write-address, write-data and write-enable inputs must be set up before the rising edge of the clock. The write operation begins with the rising edge.	Rising
W_ADDR[5:0]	Input	Dynamic	Write address	—
W_DATA[11:0]	Input	Dynamic	Write-data	—
BLK_EN	Input	Dynamic	Read port block select. When High, read operation is performed. When Low, read-data will be forced to zero. BLK_EN signal is registered through R_CLK when R_ADDR_BYPASS is Low.	High
R_CLK	Input	Dynamic	Read registers clock. All read-address, block- port select and read-enable inputs must be set up before the rising edge of the clock. The read operation begins with the rising edge.	Rising
R_ADDR[5:0]	Input	Dynamic	Read-address	—
R_ADDR_BYPASS	Input	Static	Read-address and BLK_EN register select.	Low
R_ADDR_EN	Input	Dynamic	Read-address register enable.	High
R_ADDR_SL_N	Input	Dynamic	Read-address register synchronous load.	Low
R_ADDR_SD	Input	Static	Read-address register synchronous load data.	High
R_ADDR_AL_N	Input	Dynamic	Read-address register asynchronous load.	Low
R_ADDR_AD_N	Input	Static	Read-address register asynchronous load data.	Low
R_DATA[11:0]	Output	Dynamic	Read-data	—
R_DATA_BYPASS	Input	Static	Read-data pipeline register select.	Low
R_DATA_EN	Input	Dynamic	Read-data pipeline register enable.	High
R_DATA_SL_N	Input	Dynamic	Read-data pipeline register synchronous load.	Low
R_DATA_SD	Input	Static	Read-data pipeline register synchronous load data.	High
R_DATA_AL_N	Input	Dynamic	Read-data pipeline register asynchronous load.	Low
R_DATA_AD_N	Input	Dynamic	Read-data pipeline register asynchronous load data.	Low
BUSY_FB	Input	Static	Lock access to FCB.	High
ACCESS_BUSY	Output	Dynamic	Busy signal from FCB.	High

Tip: Static inputs are defined at design time and need to be tied to 0 or 1.

Read-address and Read-data Pipeline Register Control Signals

The following table describes the functionality of the control signals on the R_ADDR and R_DATA registers.

Table 8-2. Truth Table for R_ADDR and R_DATA Registers

_AL_N	_AD_N	_BYPASS	_CLK	_EN	_SL_N	_SD	D	Q_{n+1}
0	ADn	X	X	X	X	X	X	!ADn
1	X	0	Not rising	X	X	X	X	Q_n
1	X	0	?	0	X	X	X	Q_n
1	X	0	?	1	0	SD	X	SD
1	X	0	?	1	1	X	D	D
1	X	1	X	X	X	X	D	D

9. MACC_PA (Ask a Question)

The MACC_PA macro implements multiplication, multiply-add, and multiply-accumulate functions. The MACC_PA block can accumulate the current multiplication product with a previous result, a constant, a dynamic value, or a result from another MACC_PA block. Each MACC_PA block can also be configured to perform a Dot-product operation. All the signals of the MACC_PA block have optional registers.

9.1. Features (Ask a Question)

The main features of the MACC_PA block are as follows:

- Native 18 x 18 signed multiplication and supports 17 x 17 unsigned multiplication.
- Independent third input C of data width 48 bits along with a CARRYIN, optionally registered.
- Pre-adder of B with an independent fourth input D of data width 18 bits, optionally registered.
- Internal cascade signals (48-bit CDIN and CDOUT) enable cascading of the Math blocks to support larger accumulator, adder, and subtracter without extra logic.
- Normal addition/subtraction: CARRYIN + C[47:0] + E[47:0] ± { (B[17:0] ± D[17:0]) x A[17:0] }.
- Dot product mode: (B[8:0] ± D[8:0]) x A[17:9] ± (B[17:9] ± D[17:9]) x A[8:0].
- SIMD mode for dual independent multiplication of two pairs of 9-bit operands.
- Supports both registered and unregistered inputs and outputs.
- Arithmetic right-shift by 17 bits of the loopback of CDIN.

The following figure shows a simplified block diagram of the MACC_PA block.

Figure 9-1. Simplified Block Diagram of MACC_PA

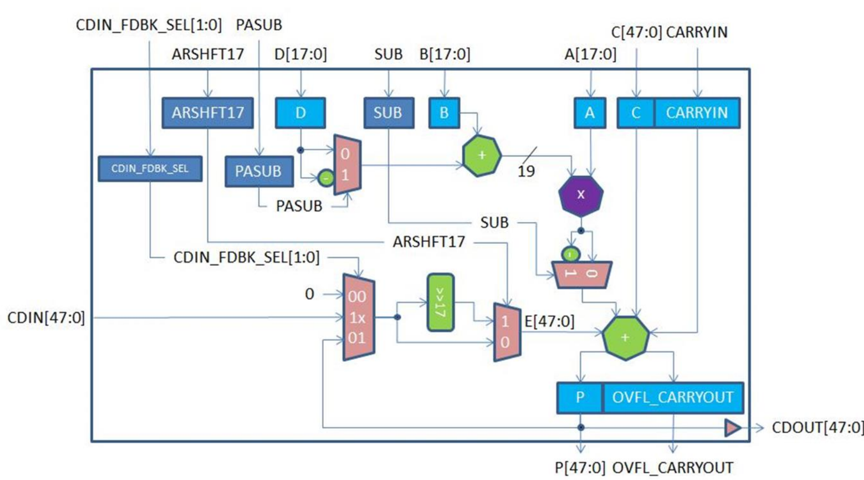


Table 9-1. MACC_PA Pin Descriptions

Port Name	Direction	Type	Polarity	Description
DOTP	Input	Static	High	Dot-product mode. When DOTP = 1, MACC_PA block performs Dot- product of two pairs of 9-bit operands. • SIMD must not be 1. • C[8:0] must be connected to CARRYIN.

Table 9-1. MACC_PA Pin Descriptions (continued)

Port Name	Direction	Type	Polarity	Description
SIMD	Input	Static	High	SIMD mode. When SIMD = 1, MACC_PA block performs dual independent multiplication of two pairs of 9-bit operands. • DOTP must not be 1. • ARSHFT17 must be 0. • D[8:0] must be 0. • C[17:0] must be 0. • E[17:0] must be 0. Refer to Table 9-2 to see how operand E is obtained from P, CDIN or 0.
OVFL_CARRYOUT_SEL	Input	Static	High	Generate OVERFLOW or CARRYOUT with result P. • OVERFLOW when OVFL_CARRYOUT_SEL = 0 • CARRYOUT when OVFL_CARRYOUT_SEL = 1
CLK	Input	Dynamic	Rising edge	Clock for A, B, C, CARRYIN, D, P, OVFL_CARRYOUT, ARSHFT17, CDIN_FDBK_SEL, PASUB and SUB registers.
AL_N	Input	Dynamic	Low	Asynchronous load for A, B, P, OVFL_CARRYOUT, ARSHFT17, CDIN_FDBK_SEL, PASUB and SUB registers. Connect to 1, if none are registered. When asserted, A, B, P and OVFL_CARRYOUT registers are loaded with zero, while the ARSHFT17, CDIN_FDBK_SEL, PASUB and SUB registers are loaded with the complementary. value of the respective _AD_N.
A[17:0]	Input	Dynamic	High	Input data A.
A_BYPASS	Input	Static	High	Bypass data A registers. Connect to 1, if not registered. See Table 9-6.
A_SRST_N	Input	Dynamic	Low	Synchronous reset for data A registers. Connect to 1, if not registered. See Table 9-6.
A_EN	Input	Dynamic	High	Enable for data A registers. Connect to 1, if not registered. See Table 9-6.
B[17:0]	Input	Dynamic	High	Input data B to Pre-adder with data D.
B_BYPASS	Input	Static	High	Bypass data B registers. Connect to 1, if not registered. See Table 9-6.
B_SRST_N	Input	Dynamic	Low	Synchronous reset for data B registers. Connect to 1, if not registered. See Table 9-6.
B_EN	Input	Dynamic	High	Enable for data B registers. Connect to 1, if not registered. See Table 9-6.
D[17:0]	Input	Dynamic	High	Input data D to Pre-adder with data B. When SIMD = 1, connect D[8:0] to 0.
D_BYPASS	Input	Static	High	Bypass data D registers. Connect to 1, if not registered. See Table 9-7.
D_ARST_N	Input	Dynamic	Low	Asynchronous reset for data D registers. Connect to 1, if not registered. See Table 9-7.
D_SRST_N	Input	Dynamic	Low	Synchronous reset for data D registers. Connect to 1, if not registered. See Table 9-7.
D_EN	Input	Dynamic	High	Enable for data D registers. Connect to 1, if not registered. See Table 9-7.
CARRYIN	Input	Dynamic	High	CARRYIN for input data C.
C[47:0]	Input	Dynamic	High	Input data C. When DOTP = 1, connect C[8:0] to CARRYIN. When SIMD = 1, connect C[8:0] to 0.

Table 9-1. MACC_PA Pin Descriptions (continued)

Port Name	Direction	Type	Polarity	Description
C_BYPASS	Input	Static	High	Bypass CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
C_ARST_N	Input	Dynamic	Low	Asynchronous reset for CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
C_SRST_N	Input	Dynamic	Low	Synchronous reset for CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
C_EN	Input	Dynamic	High	Enable for CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
CDIN[47:0]	Input	Cascade	High	Cascaded input for operand E. The entire bus must be driven by an entire CDOUT of another MACC_PA or MACC_PA_BC_ROM block. In Dot-product mode, the driving CDOUT must also be generated by a MACC_PA or MACC_PA_BC_ROM block in Dot-product mode. Refer to Table 9-2 to see how CDIN is propagated to operand E.
P[47:0]	Output	—	High	Result data. See Table 9-3.
OVFL_CARRYOUT	Output	—	High	OVERFLOW or CARRYOUT. See Table 9-4.
P_BYPASS	Input	Static	High	Bypass P and OVFL_CARRYOUT registers. Connect to 1, if not registered. See Table 9-6. P_BYPASS must be 0 when CDIN_FDBK_SEL[0] = 1. See Table 9-2.
P_SRST_N	Input	Dynamic	Low	Synchronous reset for P and OVFL_CARRYOUT registers. Connect to 1, if not registered. See Table 9-6.
P_EN	Input	Dynamic	High	Enable for P and OVFL_CARRYOUT registers. Connect to 1, if not registered. See Table 9-6.
CDOUT[47:0]	Output	Cascade	High	Cascade output of result P. See Table 9-3. Value of CDOUT is the same as P. The entire bus must either be dangling or drive an entire CDIN of another MACC_PA or MACC_PA_BC_ROM block in cascaded mode.
PASUB	Input	Dynamic	High	Subtract operation for Pre-adder of B and D.
PASUB_BYPASS	Input	Static	High	Bypass PASUB register. Connect to 1, if not registered. See Table 9-5.
PASUB_AD_N	Input	Static	Low	Asynchronous load data for PASUB register. See Table 9-5.
PASUB_SL_N	Input	Dynamic	Low	Synchronous load for PASUB register. Connect to 1, if not registered. See Table 9-5.
PASUB_SD_N	Input	Static	Low	Synchronous load data for PASUB register. See Table 9-5.
PASUB_EN	Input	Dynamic	High	Enable for PASUB register. Connect to 1, if not registered. See Table 9-5.
CDIN_FDBK_SEL[1:0]	Input	Dynamic	High	Select CDIN, P or 0 for operand E. See Table 9-2.
CDIN_FDBK_SEL_BYPASS	Input	Static	High	Bypass CDIN_FDBK_SEL register. Connect to 1, if not registered. See Table 9-5.
CDIN_FDBK_SEL_AD_N[1:0] [1:0]	Input	Static	Low	Asynchronous load data for CDIN_FDBK_SEL register. See Table 9-5.
CDIN_FDBK_SEL_SL_N	Input	Dynamic	Low	Synchronous load for CDIN_FDBK_SEL register. Connect to 1, if not registered. See Table 9-5.
CDIN_FDBK_SEL_SD_N[1:0] [1:0] CDIN_FDBK_SEL_SD_N [1:0]	Input	Static	Low	Synchronous load data for CDIN_FDBK_SEL register. See Table 9-5.
CDIN_FDBK_SEL_EN	Input	Dynamic	High	Enable for CDIN_FDBK_SEL register. Connect to 1, if not registered. See Table 9-5.

Table 9-1. MACC_PA Pin Descriptions (continued)

Port Name	Direction	Type	Polarity	Description
ARSHFT17	Input	Dynamic	High	Arithmetic right-shift for operand E. When asserted, a 17-bit arithmetic right-shift is performed on operand E. Refer to Table 9-2 to see how operand E is obtained from P, CDIN or 0. When SIMD = 1, ARSHFT17 must be 0.
ARSHFT17_BYPASS	Input	Static	High	Bypass ARSHFT17 register. Connect to 1, if not registered. See Table 9-5.
ARSHFT17_AD_N	Input	Static	Low	Asynchronous load data for ARSHFT17 register. See Table 9-5.
ARSHFT17_SL_N	Input	Dynamic	Low	Synchronous load for ARSHFT17 register. Connect to 1, if not registered. See Table 9-5.
ARSHFT17_SD_N	Input	Static	Low	Synchronous load data for ARSHFT17 register. See Table 9-5.
ARSHFT17_EN	Input	Dynamic	High	Enable for ARSHFT17 register. Connect to 1, if not registered. See Table 9-5.
SUB	Input	Dynamic	High	Subtract operation.
SUB_BYPASS	Input	Static	High	Bypass SUB register. Connect to 1, if not registered. See Table 9-5.
SUB_AD_N	Input	Static	Low	Asynchronous load data for SUB register. See Table 9-5
SUB_SL_N	Input	Dynamic	Low	Synchronous load for SUB register. Connect to 1, if not registered. See Table 9-5.
SUB_SD_N	Input	Static	Low	Synchronous load data for SUB register. See Table 9-5.
SUB_EN	Input	Dynamic	High	Enable for SUB register. Connect to 1, if not registered. See Table 9-5.

Tip: Static inputs are defined at design time and need to be tied to 0 or 1.

Table 9-2. Truth Table—Propagating Data to Operand E

CDIN_FDBK_SEL[1]	CDIN_FDBK_SEL[0]	ARSHFT17	Operand E
0	0	X	48'b0
0	1	0	P[47:0]
0	1	1	{{17{P[47]}},P[47:17]}
1	X	0	CDIN[47:0]
1	X	1	{{17{CDIN[47]}},CDIN[47:17]}

Table 9-3. Truth Table—Computation of Result P and CDOUT

SIMD	DOTP	SUB	PASUB	Result P and CDOUT
0	0	0	0	CARRYIN + C[47:0] + E[47:0] + { (B[17:0] + D[17:0]) x A[17:0] }
0	0	0	1	CARRYIN + C[47:0] + E[47:0] + { (B[17:0] - D[17:0]) x A[17:0] }
0	0	1	0	CARRYIN + C[47:0] + E[47:0] - { (B[17:0] + D[17:0]) x A[17:0] }
0	0	1	1	CARRYIN + C[47:0] + E[47:0] - { (B[17:0] - D[17:0]) x A[17:0] }
0	1	0	0	CARRYIN + C[47:0] + E[47:0] + { (B[8:0] + D[8:0]) x A[17:9] + (B[17:9] + D[17:9]) x A[8:0] } x 29
0	1	0	1	CARRYIN + C[47:0] + E[47:0] + { (B[8:0] - D[8:0]) x A[17:9] + (B[17:9] - D[17:9]) x A[8:0] } x 29
0	1	1	0	CARRYIN + C[47:0] + E[47:0] + { (B[8:0] + D[8:0]) x A[17:9] - (B[17:9] + D[17:9]) x A[8:0] } x 29

Table 9-3. Truth Table—Computation of Result P and CDOUT (continued)

SIMD	DOTP	SUB	PASUB	Result P and CDOUT
0	1	1	1	CARRYIN + C[47:0] + E[47:0] + { (B[8:0] - D[8:0]) x A[17:9] - (B[17:9] - D[17:9]) x A[8:0] } x 29
1	0	0	0	P[17:0] = CARRYIN + { B[8:0] x A[8:0] } P[47:18] = C[47:18] + E[47:18] + { (B[17:9] + D[17:9]) x A[17:9] }
1	0	0	1	P[17:0] = CARRYIN + { B[8:0] x A[8:0] } P[47:18] = C[47:18] + E[47:18] + { (B[17:9] - D[17:9]) x A[17:9] }
1	0	1	0	P[17:0] = CARRYIN + { B[8:0] x A[8:0] } P[47:18] = C[47:18] + E[47:18] - { (B[17:9] + D[17:9]) x A[17:9] }
1	0	1	1	P[17:0] = CARRYIN + { B[8:0] x A[8:0] } P[47:18] = C[47:18] + E[47:18] - { (B[17:9] - D[17:9]) x A[17:9] }

Table 9-4. Truth Table—Computation of OVFL_CARRYOUT

OVFL_CARRYOUT_SEL	OVFL_CARRYOUT	Description
0	(SUM[49] ^ SUM[48]) | (SUM[48] ^ SUM[47])	True if overflow or underflow occurred.
1	C[47] ^ E[47] ^ SUM[48]	A signal that can be used to extend the final adder in the fabric.

SUM[49:0] is defined similarly to P[47:0] as shown in Table 9-3, except that SUM is a 50-bit quantity so that no overflow can occur. SUM[48] is the carry out bit of a 48-bit final adder producing P[47:0].

Table 9-5. Truth Table for Control Registers ARSHFT17, CDIN_FDBK_SEL, PASUB, and SUB

AL_N	_AD_N	_BYPASS	CLK	_EN	_SL_N	_SD_N	D	Qn+1
0	AD_N	0	X	X	X	X	X	!AD_N
1	X	0	Not rising	X	X	X	X	Qn
1	X	0	?	0	X	X	X	Qn
1	X	0	?	1	0	SD_N	X	!SD_N
1	X	0	?	1	1	X	D	D
X	X	1	X	0	X	X	X	Qn
X	X	1	X	1	0	SD_N	X	!SD_N
X	X	1	X	1	1	X	D	D

Table 9-6. Truth Table—Data Registers A, B, P, and OVFL_CARRYOUT

AL_N	_BYPASS	CLK	_EN	_SRST_N	D	Qn+1
0	0	X	X	X	X	0
1	0	Not rising	X	X	X	Qn
1	0	?	0	X	X	Qn
1	0	?	1	0	X	0
1	0	?	1	1	D	D
X	1	X	0	X	X	Qn
X	1	X	1	0	X	0
X	1	X	1	1	D	D

Table 9-7. Truth Table—Data Registers C, CARRYIN, and D

_ARST_N	_BYPASS	CLK	_EN	_SRST_N	D	Qn+1
0	0	X	X	X	X	0
1	0	Not rising	X	X	X	Qn
1	0	?	0	X	X	Qn
1	0	?	1	0	X	0

Table 9-7. Truth Table—Data Registers C, CARRYIN, and D (continued)

_ARST_N	_BYPASS	CLK	_EN	_SRST_N	D	Qn+1
1	0	?	1	1	D	D
X	1	X	0	X	X	Qn
X	1	X	1	0	X	0
X	1	X	1	1	D	D

10. MACC_PA_BC_ROM (Ask a Question)

The MACC_PA_BC_ROM macro extends the functionality of the MACC_PA macro to provide a 16x18 ROM at the A input along with a pipelined output of B for cascading.

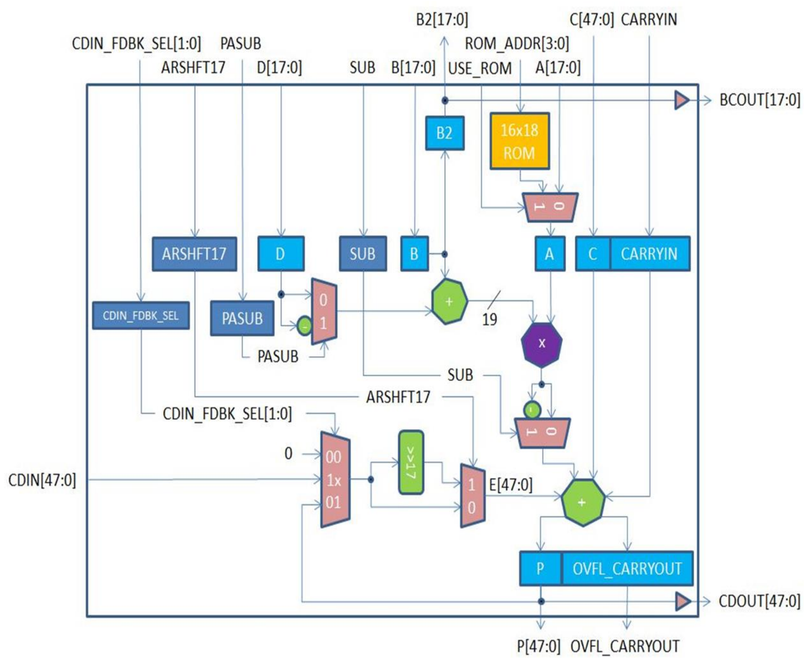
10.1. Features (Ask a Question)

The additional features of the MACC_PA_BC_ROM block are as follows:

- Selection of the A input from a 16 x 18 ROM.
- Additional pipelining of the B input for cascading to the next Math block or output to the fabric.
- Due to routing bandwidth limitations, either result P or B2 output can be used in the same MACC_PA_BC_ROM block.

The following table shows a simplified block diagram of the MACC_PA_BC_ROM block.

Figure 10-1. Slimplified Block Diagram of MACC_PA



10.2. Parameters (Ask a Question)

There is one parameter, INIT, to hold the 16x18 ROM content as a linear array. The first 18 bits is word 0, the next 18 bits is word 1, and so on.

Table 10-1. MACC_PA_BC_ROM Parameter Descriptions

Parameter	Dimensions	Description

INIT	parameter [287:0] INIT = {
	18'h0, 18'h0, 18'h0, 18'h0, 18'h0, 18'h0, 18'h0, 18'h0,
	18'h0, 18'h0, 18'h0, 18'h0, 18'h0, 18'h0, 18'h0, 18'h0
	};
		16 x 18 ROM content specified in Verilog

Table 10-1. MACC_PA_BC_ROM Parameter Descriptions (continued)

Parameter	Dimensions	Description
INIT	generic map(INIT => (B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"& B"00_0000_0000_0000_0000"))	16 x 18 ROM content specified in VHDL

PORT LIST

Table 10-2. MACC_PA_BC_ROM Pin Descriptions

Port Name	Direction	Type	Polarity	Description
DOTP	Input	Static	High	Dot-product mode. When DOTP = 1, MACC_PA_BC_ROM block performs Dot-product of two pairs of 9-bit operands. • SIMD must not be 1. • C[8:0] must be connected to CARRYIN.
SIMD	Input	Static	High	SIMD mode. When SIMD = 1, MACC_PA_BC_ROM block performs dual independent multiplication of two pairs of 9-bit operands. • DOTP must not be 1. • ARSHFT17 must be 0. • D[8:0] must be 0. • C[17:0] must be 0. E[17:0] must be 0. Refer to Table 9-2 to see how operand E is obtained from P, CDIN or 0.
OVFL_CARRYOUT_SEL	Input	Static	High	Generate OVERFLOW or CARRYOUT with result P. • OVERFLOW when OVFL_CARRYOUT_SEL = 0 • CARRYOUT when OVFL_CARRYOUT_SEL = 1
CLK	Input	Dynamic	Rising edge	Clock for A, B, C, CARRYIN, D, P, OVFL_CARRYOUT, ARSHFT17, CDIN_FDBK_SEL, PASUB and SUB registers.
AL_N	Input	Dynamic	Low	Asynchronous load for A, B, B2, P, OVFL_CARRYOUT, ARSHFT17, CDIN_FDBK_SEL, PASUB and SUB registers. Connect to 1, if none are registered. When asserted, A, B, P and OVFL_CARRYOUT registers are loaded with zero, while the ARSHFT17, CDIN_FDBK_SEL, PASUB and SUB registers are loaded with the complementary value of the respective _AD_N.
USE_ROM	Input	Static (virtual)	High	Selection for operand A. • When USE_ROM = 0, select input data A. • When USE_ROM = 1, select ROM data at ROM_ADDR.
ROM_ADDR[3:0]	Input	Dynamic	High	Address of ROM data for operand A when USE_ROM = 1.
A[17:0]	Input	Static	High	Input data for operand A when USE_ROM = 0.
A_BYPASS	Input	Dynamic	High	Bypass data A registers. Connect to 1, if not registered. See Table 9-6.

Table 10-2. MACC_PA_BC_ROM Pin Descriptions (continued)

Port Name	Direction	Type	Polarity	Description
A_SRST_N	Input	Dynamic	Low	Synchronous reset for data A registers. Connect to 1, if not registered. See Table 9-6.
A_EN	Input	Dynamic	High	Enable for data A registers. Connect to 1, if not registered. See Table 9-6.
B[17:0]	Input	Dynamic	High	Input data B to Pre-adder with data D.
B_BYPASS	Input	Static	High	Bypass data B registers. Connect to 1, if not registered. See Table 9-6.
B_SRST_N	Input	Dynamic	Low	Synchronous reset for data B registers. Connect to 1, if not registered. See Table 9-6.
B_EN	Input	Dynamic	High	Enable for data B registers. Connect to 1, if not registered. See Table 9-6.
B2[17:0]	Output	Dynamic	High	Pipelined output of input data B. Result P must be floating when B2 is used.
B2_BYPASS	Input	Static	High	Bypass data B2 registers. Connect to 1, if not registered. See Table 9-6
B2_SRST_N	Input	Dynamic	Low	Synchronous reset for data B2 registers. Connect to 1, if not registered. See Table 9-6.
B2_EN	Input	Dynamic	High	Enable for data B2 registers. Connect to 1, if not registered. See Table 9-6.
BCOUT[17:0]	Output	Cascade	High	Cascade output of B2. Value of BCOUT is the same as B2. The entire bus must either be dangling or drive an entire B input of another MACC_PA or MACC_PA_BC_ROM block.
D[17:0]	Input	Dynamic	High	Input data D to Pre-adder with data B. When SIMD = 1, connect D[8:0] to 0.
D_BYPASS	Input	Static	High	Bypass data D registers. Connect to 1, if not registered. See Table 9-7.
D_ARST_N	Input	Dynamic	Low	Asynchronous reset for data D registers. Connect to 1, if not registered. See Table 9-7.
D_SRST_N	Input	Dynamic	Low	Synchronous reset for data D registers. Connect to 1, if not registered. See Table 9-7.
D_EN	Input	Dynamic	High	Enable for data D registers. Connect to 1, if not registered. See Table 9-7.
CARRYIN	Input	Dynamic	High	CARRYIN for input data C.
C[47:0]	Input	Dynamic	High	Input data C. When DOTP = 1, connect C[8:0] to CARRYIN. When SIMD = 1, connect C[8:0] to 0.
C_BYPASS	Input	Static	High	Bypass CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
C_ARST_N	Input	Dynamic	Low	Asynchronous reset for CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
C_SRST_N	Input	Dynamic	Low	Synchronous reset for CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
C_EN	Input	Dynamic	High	Enable for CARRYIN and C registers. Connect to 1, if not registered. See Table 9-7.
CDIN[47:0]	Input	Cascade	High	Cascaded input for operand E. The entire bus must be driven by an entire CDOUT of another MACC_PA or MAC_PA_BC_ROM block. In Dot-product mode, the driving CDOUT must also be generated by a MACC_PA or MAC_PA_BC_ROM block in Dot-product mode. Refer to Table 9-2 to see how CDIN is propagated to operand E.

Table 10-2. MACC_PA_BC_ROM Pin Descriptions (continued)

Port Name	Direction	Type	Polarity	Description
P[47:0]	Output		High	Result data. See Table 9-3. B2 output must be floating when P is used.
OVFL_CARRYOUT	Output		High	OVERFLOW or CARRYOUT. See Table 9-4.
P_BYPASS	Input	Static	High	Bypass P and OVFL_CARRYOUT registers. Connect to 1, if not registered. See Table 9-6. P_BYPASS must be 0 when CDIN_FDBK_SEL[0] = 1. See Table 9-2.
P_SRST_N	Input	Dynamic	Low	Synchronous reset for P and OVFL_CARRYOUT registers. Connect to 1, if not registered. See Table 9-6.
P_EN	Input	Dynamic	High	Enable for P and OVFL_CARRYOUT registers. Connect to 1, if not registered. See Table 9-6.
CDOUT[47:0]	Output	Cascade	High	Cascade output of result P. See Table 9-3. Value of CDOUT is the same as P. The entire bus must either be dangling or drive an entire CDIN of another MACC_PA or MAC_PA_BC_ROM block incascaded mode.
PASUB	Input	Dynamic	High	Subtract operation for Pre-adder of B and D.
PASUB_BYPASS	Input	Static	High	Bypass PASUB register. Connect to 1, if not registered. See Table 9-5.
PASUB_AD_N	Input	Static	Low	Asynchronous load data for PASUB register. See Table 9-5.
PASUB_SL_N	Input	Dynamic	Low	Synchronous load for PASUB register. Connect to 1, if not registered. See Table 9-5.
PASUB_SD_N	Input	Static	Low	Synchronous load data for PASUB register. See Table 9-5.
PASUB_EN	Input	Dynamic	High	Enable for PASUB register. Connect to 1, if not registered. See Table 9-5.
CDIN_FDBK_SEL[1:0]	Input	Dynamic	High	Select CDIN, P or 0 for operand E. See Table 9-2.
CDIN_FDBK_SEL_BYPASS	Input	Static	High	Select CDIN, P or 0 for operand E. See Table 9-2.
CDIN_FDBK_SEL_AD_N [1:0]	Input	Static	Low	Asynchronous load data for CDIN_FDBK_SEL register. See Table 9-5.
CDIN_FDBK_SEL_SL_N	Input	Dynamic	Low	Synchronous load for CDIN_FDBK_SEL register. Connect to 1, if not registered. See Table 9-5.
CDIN_FDBK_SEL_SD_N [1:0]	Input	Static	Low	Synchronous load data for CDIN_FDBK_SEL register. See Table 9-5.
CDIN_FDBK_SEL_EN	Input	Dynamic	High	Enable for CDIN_FDBK_SEL register. Connect to 1, if not registered. See Table 9-5.
ARSHFT17	Input	Dynamic	High	Arithmetic right-shift for operand E. When asserted, a 17-bit arithmetic right-shift is performed on operand E. Refer to Table 9-2 to see how operand E is obtained from P, CDIN or 0. When SIMD = 1, ARSHFT17 must be 0.
ARSHFT17_BYPASS	Input	Static	High	Bypass ARSHFT17 register. Connect to 1, if not registered. See Table 9-5.
ARSHFT17_AD_N	Input	Static	Low	Asynchronous load data for ARSHFT17 register. See Table 9-5.
ARSHFT17_SL_N	Input	Dynamic	Low	Synchronous load for ARSHFT17 register. Connect to 1, if not registered. See Table 9-5.
ARSHFT17_SD_N	Input	Static	Low	Synchronous load data for ARSHFT17 register. See Table 9-5.
ARSHFT17_EN	Input	Dynamic	High	Enable for ARSHFT17 register. Connect to 1, if not registered. See Table 9-5.

Table 10-2. MACC_PA_BC_ROM Pin Descriptions (continued)

Port Name	Direction	Type	Polarity	Description
SUB	Input	Dynamic	High	Subtract operation.
SUB_BYPASS	Input	Static	High	Bypass SUB register. Connect to 1, if not registered. See Table 9-5.
SUB_AD_N	Input	Static	Low	Asynchronous load data for SUB register. See Table 9-5.
SUB_SL_N	Input	Dynamic	Low	Synchronous load for SUB register. Connect to 1, if not registered. Table 9-5.
SUB_SD_N	Input	Static	Low	Synchronous load data for SUB register. See Table 9-5.
SUB_EN	Input	Dynamic	High	Enable for SUB register. Connect to 1, if not registered. See Table 9-5.

Tip: Static inputs are defined at design time and need to be tied to 0 or 1.

11. Post-Layout Macros (Ask a Question)

11.1. GB (Ask a Question)

It is a Post-layout macro that stands for Global Buffer. The global clock network is composed of global buffers (GBs) for clock distribution. GBs distribute clocks to the left/right half of the fabric through vertical clock stripes. Each GB drives an independent half-chip global clock (GCLK). Two GBs—one from each half, are instantiated by the Libero SoC PolarFire to distribute a clock to the entire FPGA fabric. There are a total of 24 full-chip global signals and 48 half-chip global signals that may be used in a design. The half-chip globals can be driven by up to 24 fabric routed signals. Clocks driven from regular I/Os, internally generated clocks, and high fan-out signals such as resets can be routed to GBs.

Table 11-1. Truth Table

An	ENn	q (internal signal)	YNn	YSn
1	0	1	1	1
1	1	0	1	1
0	X	q	!q	!q

11.2. RGB (Ask a Question)

It is a Post-layout macro. Each GB drives row global buffers (RGBs) present on the vertical clock stripes to reach the logic sectors. Each RGB selects a global clock, a regional clock, or a fabric routed clock to drive the logic sectors located on the left or right side of the vertical clock stripe. RGBs are situated on the vertical stripes of the global network architecture inside the FPGA fabric. The global signals from the GBs are routed to the RGBs. Each RGB is independent and can be driven by fabric routing in addition to being driven by GBs. This facilitates the use of RGBs to drive regional clocks spanning a small fabric area, such as the the clock network for SERDES.

11.3. CC_CONFIG (Ask a Question)

The CC_CONFIG macro is a Post-layout macro that is responsible for generating the Carry bit for each ARI1_CC cell in the cluster. CI and CO are the carry-in and carry-out, respectively, to the cell. The intermediate carry-bits are given by CC[11:0]. The functionality of the CC_CONFIG is evaluating CC using CC[n] = !Px!Y3+PxCC[n-1] where, CC[-1] is CI and CC[12] is CO.

11.4. CFG0 (Ask a Question)

Post-layout macro that is a zero input LUT with output tied to either 0 or 1 as per INIT.

11.5. CRN_INT (Ask a Question)

The CRN_INT cell is a Post-layout macro that routes clocks from the fabric to corner cells (PLL, DLL).

Table 11-2. Truth Table

A	Y
0	0
1	1

11.6. ICB_INT (Ask a Question)

The ICB_INT cell is a Post-layout macro that routes clocks from fabric to the Interface Clock Block (ICB).

11.7. ICB_CLKINT (Ask a Question)

The ICB_CLKINT cell is a Post-layout macro that routes clocks from the Interface Clock Block (ICB) to global buffers.

11.8. HS_IO_CLK (Ask a Question)

High-speed I/O bank clock networks are Post-layout macros that are integrated into I/O banks and distribute clocks along the entire I/O bank with low-skew. These are used to clock data in and out of the I/O logic while implementing the high-speed interfaces.

11.9. CFG1A_TEST (Ask a Question)

A Post-layout macro that the user can see in NLV. It is remapped to BUFF macros.

11.10. CFG1B_TEST (Ask a Question)

A Post-layout macro that the user can see in NLV. It is remapped to BUFF macros.

11.11. CFG1C_TEST (Ask a Question)

A Post-layout macro that the user can see in NLV. It is remapped to BUFF macros.

11.12. CFG1D_TEST (Ask a Question)

A Post-layout macro that the user can see in NLV. It is remapped to BUFF macros.

11.13. CFG1A (Ask a Question)

A Post-layout macro used to implement a buffer. Output Y= f(A).

11.14. CFG1B (Ask a Question)

A Post-layout macro used to implement a buffer. Output Y= f(B).

11.15. CFG1C (Ask a Question)

A Post-layout macro used to implement a buffer. Output Y= f(C).

11.16. CFG1D (Ask a Question)

A Post-layout macro used to implement a buffer. Output Y= f(D).

11.17. CFG4A (Ask a Question)

Post-layout macro used to implement any 4-input combinational logic function. Output Y is dependent on the INIT string parameter and the value of A. The INIT string parameter is 16 bits wide. Refer to CFG4 macro for more details.

11.18. CFG4_ROM (Ask a Question)

Post-layout macro used to implement any 4-input combinational logic function. Output Y is dependent on the INIT string parameter and the value of A. The INIT string parameter is 16 bits wide. Refer to CFG4A macro for more details.

11.19. CFG4_IP_ABCD (Ask a Question)

Post-layout macro used to implement any 4-input combinational logic function. Output Y is dependent on the INIT string parameter and the value of A, B, C, and D. The INIT string parameter is 16 bits wide. It has 3 additional outputs IPB, IPC, and IPD and those are inverter outputs for B, C, and D inputs, respectively. Refer to CFG4 macro for more details.

11.20. RAM64x12_IP (Ask a Question)

Post-layout macro called within RAM64x12. Refer to RAM64x12 macro for more details. The RAM64x12 macro is designed by using pipeline register and RAM64x12_IP macro.

11.21. RAM1K20_IP (Ask a Question)

Post-layout macro for RAM1K20. Refer to RAM1K20 macro for more details. The RAM1K20 macro is designed by using pipeline register and RAM1K20_IP macro.

11.22. MACC _IP (Ask a Question)

It is the Post-layout macro for Multiply and Accumulate (MACC).

11.23. IOIN_IB (Ask a Question)

Buffer macro available in the post-layout netlist only.

Table 11-3. Truth Table

YIN	Y
Z	X
0	0
1	1

11.24. IOIN_IB_E (Ask a Question)

Buffer macro available in the post-layout netlist only. See IOIN_IB for more information.

11.25. IOIN_IB_E_ODT (Ask a Question)

Buffer macro available in the post-layout netlist with an additional ODT input.

11.26. IOTRI_OB_EB (Ask a Question)

The I/O feed through macro is available in the post-layout netlist only.

Table 11-4. Truth Table

D/E	DOUT/EOUT
0	0
1	1

11.27. IOBI_IB_OB_EB (Ask a Question)

The I/O feed through macro is available in the post-layout netlist only.

Table 11-5. Truth Table

D/E/YIN	DOUT/EOUT/Y
0	0
1	1

11.28. IO_DIFF (Ask a Question)

The I/O differential macro is available in the post-layout netlist (place holder to reserve the N location).

11.29. IOPAD_IN (Ask a Question)

Input I/O macro is available in the post-layout netlist only.

Table 11-6. Truth Table

PAD	Y, Y_HW
Z	X
0	0
1	1

11.30. IOPAD_TRI (Ask a Question)

Tri-state output buffer is available in the post-layout netlist only.

Table 11-7. Truth Table

D	E	PAD
X	0	Z
0	1	0
1	1	1

11.31. IOPAD_BI (Ask a Question)

The I/O output bypass macro is available in the post-layout netlist only.

Table 11-8. Truth Table

MODE	E	D	PAD	Y	Y_HW
OUTPUT	1	D	D	D	D
INPUT	0	X	Z	X	X
INPUT	0	X	PAD	PAD	PAD

11.32. IOPADP_IN (Ask a Question)

The I/O PAD input macro is available in the post-layout netlist only.

Table 11-9. Truth Table

PADP	N2PIN_P	IOUT_P	IOUT_HW_P
Z	X	X	X
0	X	0	0
1	X	1	1

11.33. IOPADN_IN (Ask a Question)

The I/O PAD input macro is available in the post-layout netlist only.

Table 11-10. Truth Table

PAD_P	N2POUT_P
0	1
1	0

11.34. IOPADP_TRI (Ask a Question)

The I/O PAD tri-state output macro is available in the post-layout netlist only.

Table 11-11. Truth Table

OIN_P	EIN_P	PAD_P
X	0	Z
OIN_P	1	OIN_P

11.35. IOPADN_TRI (Ask a Question)

The I/O PAD tri-state output macro is available in the post-layout netlist only.

Table 11-12. Truth Table

OIN_P	EIN_P	PAD_P
X	0	Z
0	1	1
1	1	0

11.36. IOPADP_BI (Ask a Question)

The I/O PAD bidirectional macro is available in the post-layout netlist only.

Table 11-13. Truth Table

MODE	EIN_P	OIN_P	PAD_P	N2PIN_P	IOUT_P	OUT_HW_P
OUTPUT	1	0	0	1	0	0
OUTPUT	1	1	1	0	1	1
INPUT	0	X	Z	Z	X	X
INPUT	0	X	0	0	X	X
INPUT	0	X	1	1	X	X
INPUT	0	X	0	1	0	0
INPUT	0	X	1	0	1	1

11.37. IOPADN_BI (Ask a Question)

The I/O PAD bidirectional macro is available in the post-layout netlist only.

Table 11-14. Truth Table

MODE	EIN_P	OIN_P	PAD_P	N2OUT_P
OUTPUT	1	1	0	0
OUTPUT	1	0	1	1
INPUT	0	X	Z	X
INPUT	0	X	0	X
INPUT	0	X	1	X
INPUT	0	X	0	0
INPUT	0	X	1	1

11.38. IOPADP_IN_MIPI (Ask a Question)

The differential I/O PAD input macro with MIPI low-power escape support available in the post-layout netlist only.

Table 11-15. Truth Table

PAD	Y
0	0
1	1

11.39. IOPAD_FEEDBACK (Ask a Question)

It is a Post-layout macro that is a bidirectional buffer. Refer to BIBUF for more information.

11.40. IOPADP_FEEDBACK (Ask a Question)

The I/O PAD output macro with the bidirectional input feedback enabled. It is available in the post-layout netlist only.

11.41. IOPADN_FEEDBACK (Ask a Question)

It is a Post-layout macro similar to IOPAD_FEEDBACK except that input D is inverted inside.

11.42. IOREG (Ask a Question)

A single post-layout macro for IOINFF, IOUTFF, and IOEFF. The IOD block includes registers for data-in, data-out, and output enable signals. The input registers (IOINFF) provide the registered version of the input signals from the IOA to the FPGA fabric. The output registers (IOUTFF) provide the registered version of the output signals from the FPGA fabric to the IOA. The output enable register (IOENFF) acts as a control signal for the output if the I/O is configured as tri-stated or bidirectional. These registers in IOD blocks are similar to the D-type flip-flops available in fabric logic elements.

11.43. PLL_IP (Ask a Question)

Post-layout macro used within PLL. It does frequency synthesis of the given configuration and it has an interconnection with PLL_DELAY.

11.44. PLL_DELAY (Ask a Question)

It is a Post-layout macro. Each PLL has a programmable delay line that can be configured in the reference clock path or feedback clock path. For PLLs, adding delay in the reference clock path enables clock delay, and adding delay in the feedback clock path enables clock advancement with respect to the reference clock. The PLL must be configured in external feedback mode to add the delay line in the feedback path.

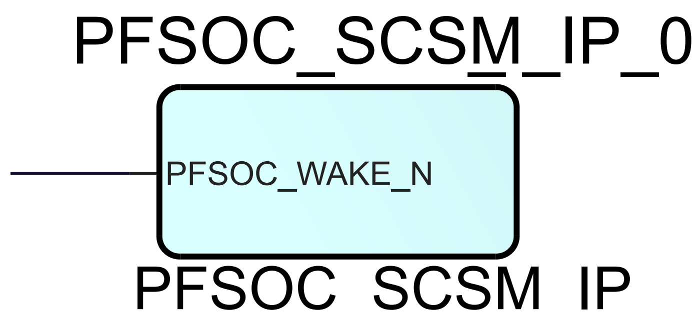
11.45. XCVR_APB_LINK (Ask a Question)

Post-layout macro used to access a transceiver quad memory map using the APB Link protocol.

11.46. PFSOC_SCSM_IP (Ask a Question)

It is a Post-layout macro that has similar functionality as of `PFSOC_SCSM`. The only difference is that the input is inverted. See the PFSOC_SCSM macro for more information.

Figure 11-1. PFSOC_SCSM_IP



12. Revision History (Ask a Question)

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	Date	Description
R	12/2025	The following change is made in this revision: • Updated the SC_STATUS section.
Q	05/2025	The following change is made in this revision: • Updated the PFSOC_SCSM section.
P	09/2024	The following change is made in this revision: • Updated the SC_STATUS section.
N	08/2024	This document is released with Libero SoC Design Suite v2024.2 without changes from v2024.1.
M	02/2024	This document is released with Libero SoC Design Suite v2024.1 without changes from v2023.2.
L	08/2023	This document is released with Libero SoC Design Suite v2023.2 without changes from v2023.1.
K	05/2023	The following change is made in this revision: • Updated the OSC_RC2MHZ section.
J	04/2023	The following change is made in this revision: • Added OSC_RC2MHZ macro to Special macros.
H	12/2022	This document is released with Libero SoC Design Suite v2022.3 without changes from v2022.2.
G	08/2022	This document is released with Libero SoC Design Suite v2022.2 without changes from v2022.1.
F	04/2022	The following changes are made in this revision: • Changed the title of the document from PolarFire FPGA Macro Library Guide to PolarFire FPGA and PolarFire SoC FPGA Macro Library User Guide. • Updated the PFSOC_SCSM macro section.
E	04/2022	The following changes are made in this revision: • Added PFSOC_SCSM macro to Special macros. • Added PFSOC_SCSM_IP macro to Post Layout macros.
D	12/2021	This document is released with Libero SoC Design Suite v2021.3 without changes from v2021.2.
C	08/2021	• Added SLE_DEBUG and SLE_INIT • Added OSC_RC160MHZ, LIVE_PROBE_A, INIT, LANECTRL , and LANECTRL_BYPASS • Added TX_PLL • Added Post-Layout Macros
B	04/2021	Editorial updates only. No technical content updates.
A	11/2020	• Migrated the document from Microsemi to Microchip format. • Formatted this document per Microchip's standards.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

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**

Microchip Information

Trademarks

The "Microchip" name and logo, the "M" logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries ("Microchip Trademarks"). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-2431-5

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.