iCE40 Technology Library

Technical Note

FPGA-TN-02026-3.3

September 2024

Disclaimers

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
I2C	Inter-Integrated Circuit
I/O	Input/Output
LED	Light-emitting Diode
LPSG	Low Power Strobe Generator
PLL	Phase Locked Loop
RAM	Random Access Memory
SPI	Serial Peripherall Interface

1. Register Primitives

1.1. SB_DFF – D Flip-Flop

Data: D is loaded into the flip-flop during a rising clock edge transition.

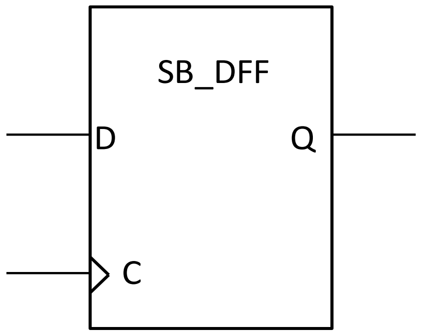

Figure 1.1. SB_DFF – D Flip-Flop

Table 1.1. SB_DFF – D Flip-Flop

Inputs			Output
	D	**C**	**Q**
	0	↑	0
	1	↑	1
Power On State	X	X	0

Key
↑ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL use

This register is inferred during synthesis and can also be explicitly instantiated.

Verilog Instantiation

// SB_DFF - D Flip-Flop.

```
SB_DFF SB_DFF_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       );
// End of SB_DFF instantiation
```

VHDL Instantiation

-- SB_DFF - D Flip-Flop.

```
SB_DFF_inst: SB_DFF
       port map (
       Q => Q, -- Registered Output
       C => C, -- Clock
       D => D, -- Data
       );
-- End of SB_DFF instantiation
```

1.2. SB_DFFE –D Flip-Flop with Clock Enable

Data D is loaded into the flip-flop when Clock Enable E is high, during a rising clock edge transition.

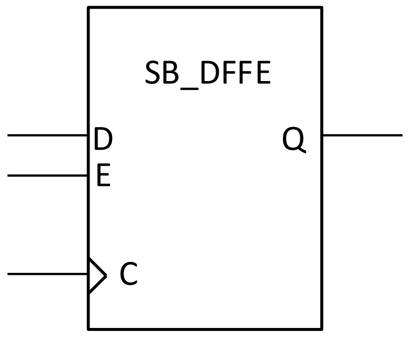

Figure 1.2. SB_DFFE –D Flip-Flop with Clock Enable

Table 1.2. SB_DFFE –D Flip-Flop with Clock Enable

Inputs			Output
E	**D**	**C**	**Q**
0	X	X	Previous Q
1	0	↗	0
1	1	↗	1
Power On State	X	X	0

Key
↗ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don’t care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Input E: Logic ‘1’

Note that explicitly connecting a logic ‘1’ value to port E results in a non-optimal implementation, since an extra LUT is used to generate the logic ‘1’. It is recommended that the user leave the port E unconnected, or use the corresponding flip-flop without Enable functionality i.e. the DFF primitive.

Verilog Instantiation

// SB_DFFE - D Flip-Flop with Clock Enable.

```
SB_DFFE      SB_DFFE_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .E(E), // Clock Enable
       );
// End of SB_DFFE instantiation
```

VHDL Instantiation

-- SB_DFFE - D Flip-Flop with Clock Enable.

```
SB_DFFE_inst: SB_DFFE
      port map (
      Q => Q,       -- Registered Output
      C => C,       -- Clock
      D => D,       -- Data
      E => E,       -- Clock Enable
      );

-- End of SB_DFFE instantiation
```

1.3. SB_DFFSR – D Flip-Flop with Synchronous Reset

Data: D is loaded into the flip-flop when Reset R is low during a rising clock edge transition.

Reset: R input is active high, overrides all other inputs and resets the Q output during a rising clock edge.

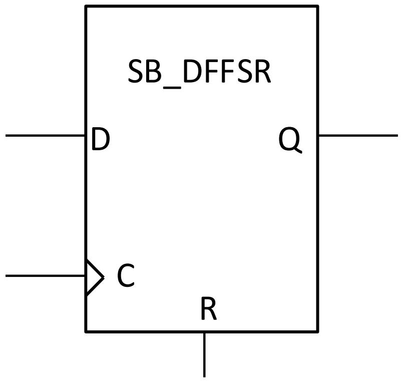

Figure 1.3. SB_DFFSR – D Flip-Flop with Synchronous Reset

Table 1.3. SB_DFFSR – D Flip-Flop with Synchronous Reset

Inputs			Output
R	**D**	**C**	**Q**
1	X	X	Previous Q
X	0	↑	0
0	1	↑	1
0	X	X	0
Power On State			

Key
↑ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input R: Logic '0'

Verilog Instantiation

// SB_DFFSR - D Flip-Flop, Reset is synchronous with the rising clock edge

```
SB_DFFSR SB_DFFSR_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .R(R)  // Synchronous Reset
       );

// End of SB_DFFSR instantiation
```

VHDL Instantiation

-- SB_DFFSR - D Flip-Flop, Reset is synchronous with the rising clock edge

```
SB_DFFSR_inst : SB_DFFSR
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       D => D,       -- Data
       R => R -- Synchronous Reset
       );

-- End of SB_DFFSR instantiation
```

1.4. SB_DFFR – D Flip-Flop with Asynchronous Reset

Data: D is loaded into the flip-flop when R is low during a rising clock edge transition.

Reset: R input is active high, overrides all other inputs and asynchronously resets the Q output.

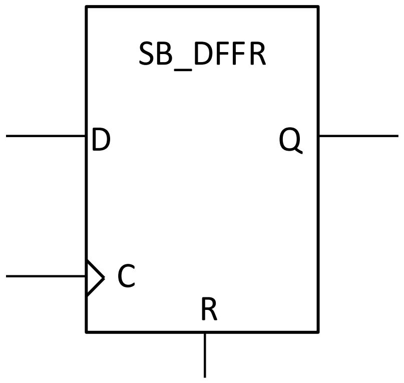

Figure 1.4. SB_DFFR – D Flip-Flop with Asynchronous Reset

Table 1.4. SB_DFFR – D Flip-Flop with Asynchronous Reset

Inputs			Output
R	**D**	**C**	**Q**
1	X	X	0
0	0	↑	0
0	1	↑	1
Power On State	X	X	0

Key

↑	– Rising Edge
1	– High logic level
0	– Low logic level
X	– Don’t care
?	– Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Input R: Logic ‘0’

Verilog Instantiation

// SB_DFFR - D Flip-Flop, Reset is asynchronous to the clock.

```
SB_DFFR  SB_DFFR_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .R(R)  // Asynchronous Reset
       );
```

VHDL Instantiation

-- SB_DFFR - D Flip-Flop, Reset is asynchronous to the clock.

```
SB_DFFR_inst: SB_DFFR
       port map (
       Q => Q,      -- Registered Output
       C => C,      -- Clock
       D => D,      -- Data
       R => R -- Asynchronous Reset
       );

-- End of SB_DFFR instantiation
```

1.5. SB_DFFSS – D Flip-Flop with Synchronous Set

Data: D is loaded into the flip-flop when the Synchronous Set S is low during a rising clock edge transition.

Set: S input is active high, overrides all other inputs and synchronously sets the Q output.

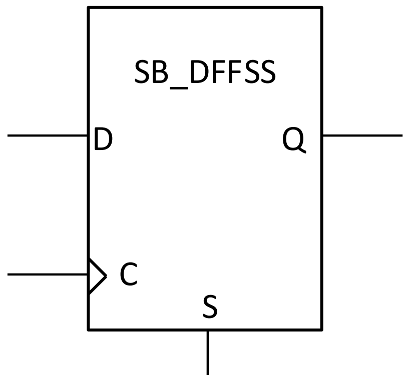

Figure 1.5. SB_DFFSS – D Flip-Flop with Synchronous Set

Table 1.5. SB_DFFSS – D Flip-Flop with Synchronous Set

Inputs			Output
S	**D**	**C**	**Q**
1	X	↗	1
0	0	↗	0
0	1	↗	1
Power On State	X	X	0

Key
↗ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input S: Logic '0'

Verilog Instantiation

// SB_DFFSS - D Flip-Flop, Set is synchronous with the rising clock edge,

```
SB_DFFSS SB_DFFSS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .S(S)  // Synchronous Set
       );

// End of SB_DFFSS instantiation
```

VHDL Instantiation

-- SB_DFFSS - D Flip-Flop, Set is synchronous with the rising clock edge

```
SB_DFFSS_instSB_DFFSS
      port map (
      Q => Q,      -- Registered Output
      C => C,      -- Clock
      D => D,      -- Data
      S => S -- Synchronous Set
      );

-- End of SB_DFFSS instantiation
```

1.6. SB_DFFS – D Flip-Flop with Asynchronous Set

Data: D is loaded into the flip-flop when S is low during a rising clock edge transition.

Set: S input is active high, and it overrides all other inputs and asynchronously sets the Q output.

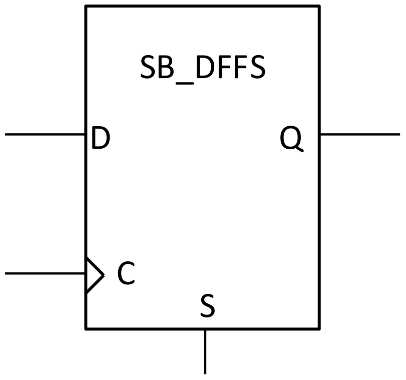

Figure 1.6. SB_DFFS – D Flip-Flop with Asynchronous Set

Table 1.6. SB_DFFS – D Flip-Flop with Asynchronous Set

Inputs			Output	
S	**D**	**C**	**Q**	Key
				↗ – Rising Edge
1	X	X	1	1 – High logic level
0	0	↗	0	0 – Low logic level
0	1	↗	1	X – Don't care
Power On State	X	X	0	? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input S: Logic '0'

Verilog Instantiation

// SB_DFFS - D Flip-Flop, Set is synchronous with the rising clock edge,

```
SB_DFFS  SB_DFFS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .S(S)  // Asynchronous Set
       );

// End of SB_DFFS instantiation
```

VHDL Instantiation

-- SB_DFFS - D Flip-Flop, Set is synchronous with the rising clock edge

```
SB_DFFS_inst: SB_DFFS
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       D => D,       -- Data
       S => S -- Asynchronous Set
       );

-- End of SB_DFFS instantiation
```

1.7. SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset

Data: D is loaded into the flip-flop when S is low during a rising clock edge transition.

Set: S input is active high, and it overrides all other inputs and asynchronously sets the Q output.

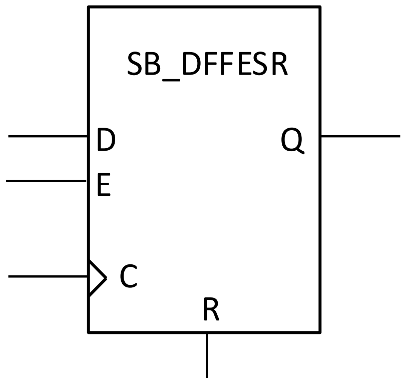

Figure 1.7. SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset

Table 1.7. SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset

Inputs				Output
R	**E**	**D**	**C**	**Q**
1	1	1	↗	0
0	X	0	X	0
0	0	1	↗	
	0	1	↗	1
Power On State	Power On Stage	X	X	0

Key
↗ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input S: Logic '0'

Verilog Instantiation

// SB_DFFS - D Flip-Flop, Set is asynchronous to the rising clock edge

```
SB_DFFS  SB_DFFS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .S(S)  // Asynchronous Set
       );

// End of SB_DFFS instantiation
```

VHDL Instantiation

-- SB_DFFS - D Flip-Flop, Set is asynchronous to the rising clock edge

```
SB_DFFS_inst: SB_DFFS
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       D => D,       -- Data
       S => S -- Asynchronous Set
       );

-- End of SB_DFFS instantiation
```

1.8. SB_DFFER – D Flip-Flop with Clock Enable and Asynchronous Reset

Data: D is loaded into the flip-flop when Reset R is low and Clock Enable E is high during a rising clock edge transition.

Reset: R, when asserted with Clock Enable E high, synchronously resets the Q output during a rising clock edge.

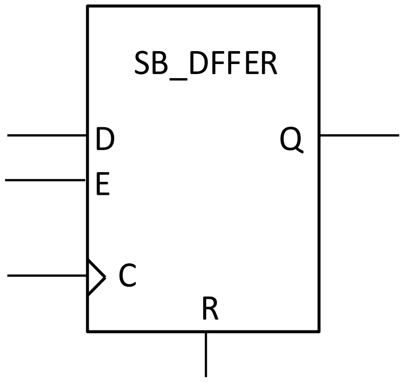

Figure 1.8. SB_DFFER – D Flip-Flop with Clock Enable and Asynchronous Reset

Table 1.8. SB_DFFER – D Flip-Flop with Clock Enable and Asynchronous Reset

Inputs				**Output**
R	**E**	**D**	**C**	**Q**
1	X	X	X	0
0	0	X	X	Previous **Q**
0	1	0	↗	0
0	1	1	↗	1
Power On State	X	X	X	0

Key
↗ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input R: Logic '0'

Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E results in a non-optimal implementation, since an extra LUT is used to generate the Logic '1'. If your intention is to keep the FF always enabled, it is recommended that either port E be left unconnected, or the corresponding FF primitive without a Clock Enable port be used.

Verilog Instantiation

// SB_DFFER - D Flip-Flop, Reset is asynchronously on rising clock edge with Clock Enable.

```
SB_DFFER SB_DFFER_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .R(R)  // Asynchronously Reset
       );

       // End of SB_DFFER instantiation
```

VHDL Instantiation

-- SB_DFFER - D Flip-Flop, Reset is asynchronously -- on rising clock edge with Clock Enable.

```
SB_DFFER_inst : SB_DFFER
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       E => E,       -- Clock Enable
       D => D,       -- Data
       R => R -- Asynchronously Reset
       );

-- End of SB_DFFER instantiation
```

1.9. SB_DFFESS – D Flip-Flop with Clock Enable and Synchronous Set

Data: D is loaded into the flip-flop when S is low and E is high during a rising clock edge transition.

Set: Asserting S when Clock Enable E is high, synchronously sets the Q output.

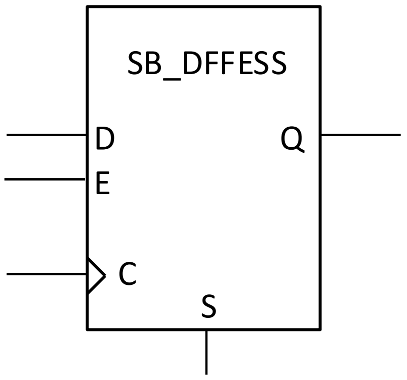

Figure 1.9. SB_DFFESS – D Flip-Flop with Clock Enable and Synchronous Set

Table 1.9. SB_DFFESS – D Flip-Flop with Clock Enable and Synchronous Set

Inputs				Output
S	**E**	**D**	**C**	**Q**
1	1	X	↑	1
0	0	X	X	Previous **Q**
0	1	0	↑	0
0	1	1	↑	1
Power On State	X	X	X	0

Key
↑ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input R: Logic '0'

Input S: Logic '0'

Verilog Instantiation

// SB_DFFESS - D Flip-Flop, Set is synchronous with rising clock edge and Clock Enable.

```
SB_DFFESS SB_DFFESS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .S(S)  // Synchronously Set
       );

// End of SB_DFFESS instantiation
```

VHDL Instantiation

-- SB_DFFESS - D Flip-Flop, Set is synchronous with rising clock edge and Clock Enable.

```
SB_DFFESS_inst : SB_DFFESS
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       E => E,       -- Clock Enable
       D => D,       -- Data
       S => S -- Synchronously Set
       );

-- End of SB_DFFESS instantiation
```

1.10. SB_DFFES – D Flip-Flop with Clock Enable and Asynchronous Set

Data: D is loaded into the flip-flop when S is low and E is high during a rising clock edge transition.

Set: S input is active high, overrides all other inputs and asynchronously sets the Q output.

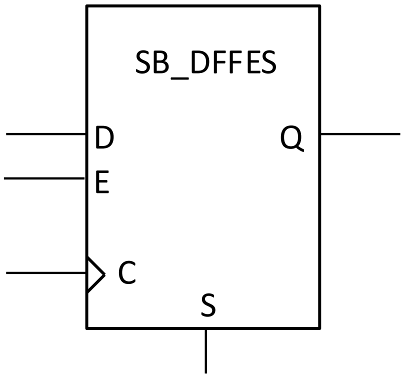

Figure 1.10. SB_DFFES – D Flip-Flop with Clock Enable and Asynchronous Set

Table 1.10. SB_DFFES – D Flip-Flop with Clock Enable and Asynchronous Set

Inputs				Output
S	**E**	**D**	**CLK**	**Q**
1	X	X	X	1
0	0	X	X	Previous **Q**
0	1	0	↑	0
0	1	1	↑	1
Power On State	X	X	X	0

Key
↑ – Rising Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input S: Logic '0'

Input E: Logic '1'

Verilog Instantiation

// SB_DFFES - D Flip-Flop, Set is synchronous with rising clock edge and Clock Enable.

```
SB_DFFESS SB_DFFESS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .S(S)        // Asynchronously Set
       );

// End of SB_DFFES instantiation
```

VHDL Instantiation

-- SB_DFFES - D Flip-Flop, Set is synchronous with rising clock edge and Clock Enable.

```
SB_DFFES_inst : SB_DFFESS
        port map (
        Q => Q,       -- Registered Output
        C => C,       -- Clock
        E => E,       -- Clock Enable
        D => D,       -- Data
        S => S -- Asynchronously Set
        );

-- End of SB_DFFES instantiation
```

1.11. SB_DFFN – D Flip-Flop – Negative Edge Clock

Data: D is loaded into the flip-flop during the falling clock edge transition.

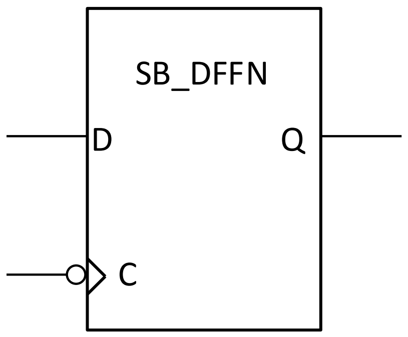

Figure 1.11. SB_DFFN – D Flip-Flop – Negative Edge Clock

Table 1.11. SB_DFFN – D Flip-Flop – Negative Edge Clock

Inputs			Output
E	**D**	**C**	**Q**
1	0	↘	0
1	1	↘	1
Power On State	X	X	0

Key	
↘	– Falling Edge
1	– High logic level
0	– Low logic level
X	– Don’t care
?	– Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Verilog Instantiation

// SB_DFFN - D Flip-Flop – Negative Edge Clock.

```
SB_DFFN SB_DFFN_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       );

// End of SB_DFFN instantiation
```

VHDL Instantiation

-- SB_DFFN - D Flip-Flop – Negative Edge Clock.

```
SB_DFFN_inst : SB_DFFN
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       D => D,       -- Data
       );

-- End of SB_DFFN instantiation
```

1.12. SB_DFFNE – D Flip-Flop – Negative Edge Clock and Clock Enable

Data: D is loaded into the flip-flop when E is high, during the falling clock edge transition.

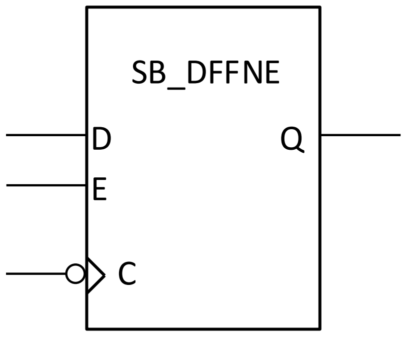

Figure 1.12. SB_DFFNE – D Flip-Flop – Negative Edge Clock and Clock Enable

Table 1.12. SB_DFFNE – D Flip-Flop – Negative Edge Clock and Clock Enable

Inputs			Output
E	**D**	**C**	**Q**
0	X	X	0
1	0	↘	0
1	1	↘	1
Power On State	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Input E: Logic ‘1’

Note that explicitly connecting a Logic ‘1’ value to port E results in a non-optimal implementation, since an extra LUT is used to generate the Logic ‘1’. If your intention is to keep the FF always enabled, it is recommended that either port E be left unconnected, or the corresponding FF without a Clock Enable port be used.

Verilog Instantiation

// SB_DFFNE - D Flip-Flop – Negative Edge Clock and Clock Enable.

```
SB_DFFNE     SB_DFFNE_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .E(E), // Clock Enable
       );

// End of SB_DFFNE instantiation
```

VHDL Instantiation

-- SB_DFFNE - D Flip-Flop – Negative Edge Clock and Clock Enable.

```
SB_DFFNE_inst : SB_DFFNE
       port map (
       Q => Q,      -- Registered Output
       C => C,      -- Clock
       D => D,      -- Data
       E => E,      -- Clock Enable
       );

-- End of SB_DFFNE instantiation
```

1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset

Data: D is loaded into the flip-flop when R is low during the falling clock edge transition.

Reset: R input is active high, overrides all other inputs and resets the Q output during the falling clock edge transition.

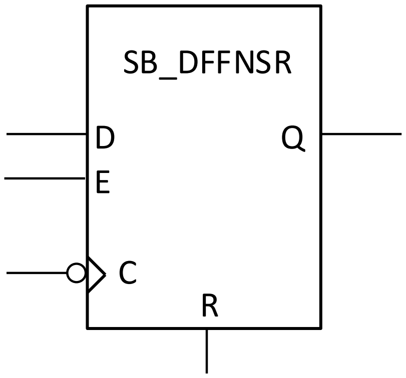

Figure 1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset

Table 1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset

Inputs			Output
R	**D**	**C**	**Q**
1	X	↘	1
X	X	↗	No Change
0	0	↘	0
0	1	↘	1
Power On State	X	X	0

Key
↗ – Rising Edge
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input R: Logic '0'

Verilog Instantiation

// SB_DFFNSR - D Flip-Flop – Negative Edge Clock, Reset is synchronous with the falling.

```
SB_DFFNSR  SB_DFFNSR_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .R(R)  // Synchronous Reset
       );

// End of SB_DFFNSR instantiation
```

VHDL Instantiation

-- SB_DFFNSR - D Flip-Flop – Negative Edge Clock, Reset is synchronous with the falling clock edge.

```
SB_DFFNSR_inst: SB_DFFNSR
      port map (
      Q => Q,      -- Registered Output
      C => C,      -- Clock
      D => D,      -- Data
      R => R -- Synchronous Reset
      );

End of SB_DFFNSR instantiation
```

1.14. SB_DFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset

Data: D is loaded into the flip-flop when R is low during the falling clock edge transition.

Reset: R input is active high, overrides all other inputs and asynchronously resets the Q output

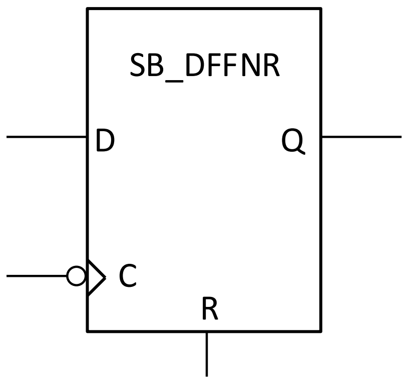

Figure 1.14. SB_DFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset

Table 1.14. SB_DFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset

Inputs			Output
E	**D**	**CLK**	**Q**
1	X	X	0
0	0	↘	0
0	1	↘	1
Power On State	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don’t care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Input R: Logic ‘0’

Verilog Instantiation

// SB_DFFNR - D Flip-Flop – Negative Edge Clock, Reset is asynchronous to the clock.

```
SB_DFFNR  SB_DFFNR_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .R(R)  // Asynchronously Reset
       );

// End of SB_DFFNR instantiation
```

VHDL Instantiation

-- SB_DFFNR - D Flip-Flop – Negative Edge Clock, Reset is asynchronous to the clock.

```
SB_DFFNR_inst : SB_DFFNR
       port map (
       Q => Q,       -- Registered Output
       C => C,       -- Clock
       D => D,       -- Data
       R => R -- Asynchronously Reset
       );

-- End of SB_DFFNR instantiation
```

1.15. SB_DFFNSS – D Flip-Flop – Negative Edge Clock with Synchronous Set

Data: D is loaded into the flip-flop when S is low during the falling clock edge transition.

Set: S input is active high, overrides all other inputs and synchronously sets the Q output.

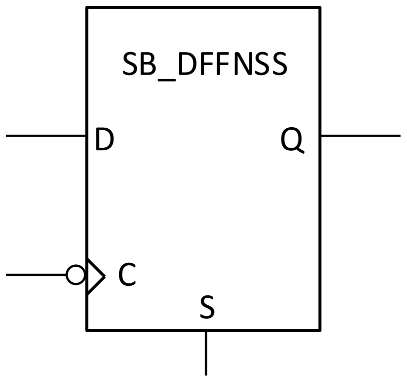

Figure 1.15. SB_DFFNSS – D Flip-Flop – Negative Edge Clock with Synchronous Set

Table 1.15. SB_DFFNSS – D Flip-Flop – Negative Edge Clock with Synchronous Set

Inputs			Output
S	**D**	**C**	**Q**
1	X	↘	1
0	0	↘	0
0	1	↘	1
Power On State	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input S: Logic '0'

Verilog Instantiation

// SB_DFFNSS - D Flip-Flop – Negative Edge Clock, Set is synchronous with the falling clock edge,

```
SB_DFFNSS SB_DFFNSS_inst (
      .Q(Q), // Registered Output
      .C(C), // Clock
      .D(D), // Data
      .S(S) // Synchronous Set
      );

// End of SB_DFFNSS instantiation
```

VHDL Instantiation

-- SB_DFFNSS - D Flip-Flop – Negative Edge Clock, Set is synchronous with the falling clock edge.

```
SB_DFFNSS_inst : SB_DFFNSS
      port map (
      Q => Q, -- Registered Output
      C => C, -- Clock
      D => D, -- Data
      S => S -- Synchronous Set
      );

-- End of SB_DFFNSS instantiation
```

1.16. SB_DFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set

Data: D is loaded into the flip-flop when S is low during the falling clock edge transition.

Set: S input is active high, overrides all other inputs and asynchronously sets the Q output.

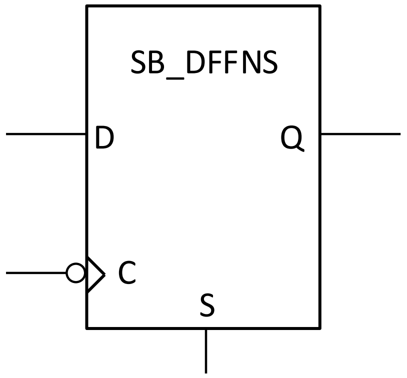

Figure 1.16. SB_DFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set

Table 1.16. SB_DFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set

Inputs			Output
S	**D**	**C**	**Q**
1	X	X	1
0	0	↘	0
0	1	↘	1
Power On State	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don’t care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Input S: Logic ‘0’

Verilog Instantiation

// SB_DFFNS - D Flip-Flop – Negative Edge Clock, Set is asynchronous to the falling clock edge.

```
SB_DFFNS  SB_DFFNS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .D(D), // Data
       .S(S)  // Asynchronous Set
       );

// End of SB_DFFNS instantiation
```

VHDL Instantiation

-- SB_DFFNS - D Flip-Flop – Negative Edge Clock, Set is asynchronous to the falling clock edge.

```
SB_DFFNS_inst : SB_DFFNS
        port map (
        Q => Q,       -- Registered Output
        C => C,       -- Clock
        D => D,       -- Data
        S => S -- Asynchronous Set
        );

-- End of SB_DFFNS instantiation
```

1.17. SB_DFFNESR – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Reset

Data: D is loaded into the flip-flop when R is low and E is high during the falling clock edge transition.

Reset: Asserting R when the Clock Enable E is high, synchronously resets the Q output during the falling clock edge.

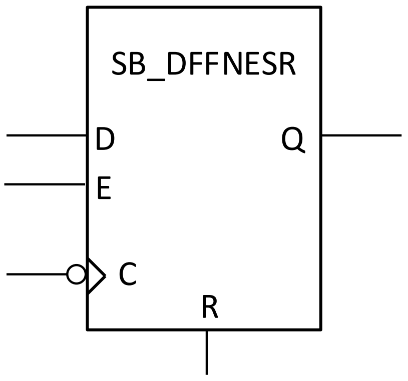

Figure 1.17. SB_DFFNESR – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Reset

Table 1.17. SB_DFFNESR – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Reset

Inputs				**Output**
R	**E**	**D**	**C**	**Q**
1	1	X	↘	1
X	0	X	X	Previous **Q**
0	1	0	↘	0
0	1	1	↘	1
Power On State	X	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input R: Logic '0'

Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E results in a non-optimal implementation, since an extra LUT is used to generate the Logic '1'. If your intention is to keep the FF always enabled, it is recommended that either port E be left unconnected, or the corresponding FF without a Clock Enable port be used.

Verilog Instantiation

// SB_DFFNESR - D Flip-Flop – Negative Edge Clock, Reset is synchronous with falling clock edge Clock Enable.

```
SB_DFFNESR   SB_DFFNESR_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .R(R)  // Synchronous Reset
       );

// End of SB_DFFNESR instantiation
```

VHDL Instantiation

-- SB_DFFNESR - D Flip-Flop – Negative Edge Clock, Reset is synchronous with falling clock edge Clock Enable.

```
SB_DFFNESR_inst : SB_DFFNESR
       port map (
       Q => Q,      -- Registered Output
       C => C,      -- Clock
       E => E,      -- Clock Enable
       D => D,      -- Data
       R => R -- Synchronous Reset
       );

-- End of SB_DFFNESR instantiation
```

1.18. SB_DFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset

Data: D is loaded into the flip-flop when R is low and E is high during the falling clock edge transition.

Reset: R input is active high, and it overrides all other inputs and asynchronously resets the Q output.

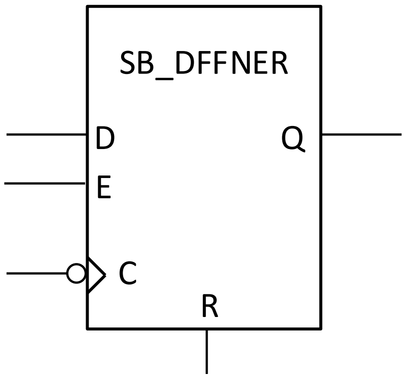

Figure 1.18. SB_DFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset

Table 1.18. SB_DFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset

Inputs				**Output**
R	**E**	**D**	**C**	**Q**
1	1	X	X	1
0	0	X	X	Previous **Q**
0	1	0	↘	0
0	1	1	↘	1
Power On State	X	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input R: Logic '0'

Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E results in a non-optimal implementation, since an extra LUT is used to generate the Logic '1'. If your intention is to keep the FF always enabled, it is recommended that either port E be left unconnected, or the corresponding FF without a Clock Enable port be used.

Verilog Instantiation

// SB_DFFNER - D Flip-Flop – Negative Edge Clock, Reset is asynchronously
// on falling clock edge and Clock Enable.

```
SB_DFFNER SB_DFFNER_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .R(R) // Asynchronously Reset
);

// End of SB_DFFNER instantiation
```

VHDL Instantiation

-- SB_DFFNER - D Flip-Flop – Negative Edge Clock, Reset is asynchronously

-- on falling clock edge and Clock Enable.

```
SB_DFFNER_inst: SB_DFFNER
       port map (
       Q => Q, -- Registered Output
       C => C, -- Clock
       E => E, -- Clock Enable
       D => D, -- Data
       R => R -- Asynchronously Reset
       );

-- End of SB_DFFNER instantiation
```

1.19. SB_DFFNESS – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Set

Data: D is loaded into the flip-flop when S is low and E is high during the falling clock edge transition.

Set: S and E inputs high, synchronously sets the Q output on the falling clock edge transition.

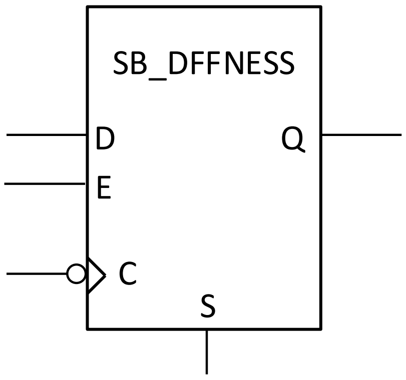

Figure 1.19. SB_DFFNESS – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Set

Table 1.19. SB_DFFNESS – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Set

Inputs				Output
S	**E**	**D**	**C**	**Q**
1	1	X	↘	1
X	0	X	X	Previous **Q**
0	1	0	↘	0
0	1	1	↘	1
Power On State	X	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic '0'

Input C: Logic '0'

Input S: Logic '0'

Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E results in a non-optimal implementation, since an extra LUT is used to generate the Logic '1'. If your intention is to keep the FF always enabled, it is recommended that either port E be left unconnected, or the corresponding FF without a Clock Enable port be used.

Verilog Instantiation

// SB_DFFNESS - D Flip-Flop – Negative Edge Clock, Set is synchronous with falling clock edge,

// and Clock Enable.

```
SB_DFFNESS SB_DFFNESS_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .S(S) // Synchronously Set
       );

// End of SB_DFFNESSinstantiation
```

VHDL Instantiation

-- SB_DFFNESS - D Flip-Flop – Negative Edge Clock, Set is synchronous with falling clock edge,

-- and Clock Enable.

```
SB_DFFNESS_inst : SB_DFFNESS
       port map (
       Q => Q, -- Registered Output
       C => C, -- Clock
       E => E, -- Clock Enable
       D => D, -- Data
       S => S -- Synchronously Set
       );

-- End of SB_DFFNESS instantiation
```

1.20. SB_DFFNES – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Set

Data: D is loaded into the flip-flop when S is low and E is high during the falling clock edge transition.

Set: S input is active high, and it overrides all other inputs and asynchronously sets the Q output.

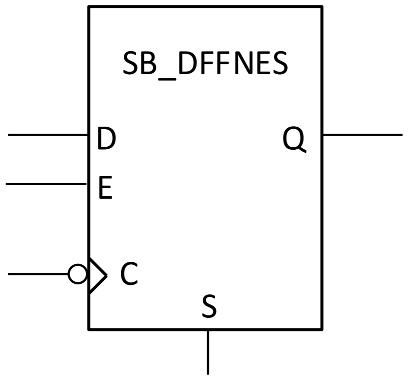

Figure 1.20. SB_DFFNES – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Set

Table 1.20. SB_DFFNES – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Set

Inputs				**Output**
S	**E**	**D**	**CLK**	**Q**
1	X	X	X	1
0	0	X	X	Previous **Q**
0	1	0	↘	0
0	1	1	↘	1
Power On State	X	X	X	0

Key
↘ – Falling Edge
1 – High logic level
0 – Low logic level
X – Don't care
? – Unknown

HDL Usage

This register is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:

Input D: Logic ‘0’

Input C: Logic ‘0’

Input S: Logic ‘0’

Input E: Logic ‘1’

Note that explicitly connecting a Logic ‘1’ value to port E results in a non-optimal implementation, since an extra LUT is used to generate the Logic ‘1’. If your intention is to keep the FF always enabled, it is recommended that either port E be left unconnected, or the corresponding FF without a Clock Enable port be used.

Verilog Instantiation

// SB_DFFNES - D Flip-Flop – Negative Edge Clock, Set is asynchronous on falling clock edge with clock // Enable.

```
SB_DFFNES SB_DFFNES_inst (
       .Q(Q), // Registered Output
       .C(C), // Clock
       .E(E), // Clock Enable
       .D(D), // Data
       .S(S) // Asynchronously Set
       );

// End of SB_DFFNESinstantiation
```

VHDL Instantiation

-- SB_DFFNES - D Flip-Flop – Negative Edge Clock, Set is asynchronous

-- on falling clock edge and Clock Enable.

```
SB_DFFNES_inst: SB_DFFNES
       port map (
       Q => Q, -- Registered Output
       C => C, -- Clock
       E => E, -- Clock Enable
       D => D, -- Data
       S => S -- Asynchronously Set
       );

-- End of SB_DFFNES instantiation
```

2. Combinational Logic Primitives

2.1. SB_LUT4

The LUT unit is a simple ROM 4 input look-up function table.

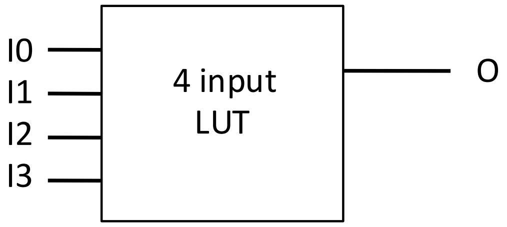

Figure 2.1. SB_LUT4

Initialization Values

LUT state initialization parameter LUT_INIT = 16'hxxxx;

Table 2.1. SB_LUT4

Inputs				Output
I3	**I2**	**I1**	**I0**	**O**
0	0	0	0	LUT_INIT[0]
0	0	0	1	LUT_INIT[1]
0	0	1	0	LUT_INIT[2]
0	0	1	1	LUT_INIT[3]
0	1	0	0	LUT_INIT[4]
0	1	0	1	LUT_INIT[5]
0	1	1	0	LUT_INIT[6]
0	1	1	1	LUT_INIT[7]
1	0	0	0	LUT_INIT[8]
1	0	0	1	LUT_INIT[9]
1	0	1	0	LUT_INIT[10]
1	0	1	1	LUT_INIT[11]
1	1	0	0	LUT_INIT[12]
1	1	0	1	LUT_INIT[13]
1	1	1	0	LUT_INIT[14]
1	1	1	1	LUT_INIT[15]

HDL Usage

This primitive is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns logic value ‘0’ to unconnected input ports.

Verilog Instantiation

// SB_LUT4 : 4-input Look-Up Table

```
SB_LUT4      SB_LUT4_inst (
       .O (O),       // output
       .I0 (I0),     // data input 0
       .I1 (I1),     // data input 1
       .I2 (I2),     // data input 2
       .I3 (I3)      // data input 3
       );
defparam SB_LUT4_inst.LUT_INIT=16'hxxxx;
//LUT state initialization parameter, 16 bits.

//End of SB_LUT4 instantiation
```

VHDL Instantiation

-- SB_LUT4 : 4-input Look-Up Table.

```
SB_LUT4_inst: SB_LUT4
      generic map(
         LUT_INIT => X"0001"  -- LUT state initialization parameter, 16 bits
                   )
      port map (
         I0 => I0,
         I1 => I1,
         I2 => I2,
         I3 => I3,
         O => O
      );
```

2.2. SB_CARRY

Carry Logic

The dedicated Carry Logic within each Logic Cell primarily accelerates and improves the efficiency of arithmetic logic such as adders, accumulators, subtracters, incrementers, decrementers, counters, ALUs, and comparators. The Carry Logic also supports a limited number of wide combinational logic functions.

The figure below illustrates the Carry Logic structure within a Logic Cell. The Carry Logic shares inputs with the associated Look-Up Table (LUT). The I1 and I2 inputs of the LUT directly feed the Carry Logic. The carry input from the previous adjacent Logic Cell optionally provides an alternate input to the LUT4 function, supplanting the I3 input.

Carry Logic Structure within a Logic Cell

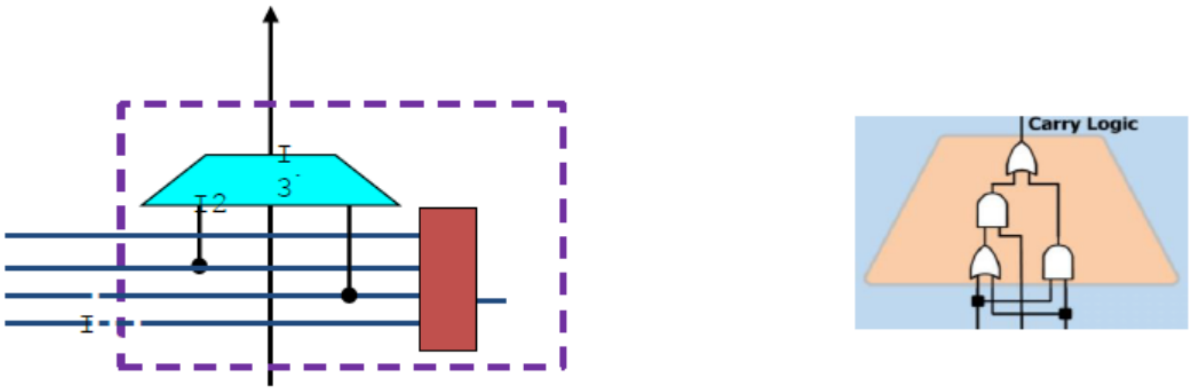

Figure 2.2. Carry Logic Structure within a Logic Cell

Table 2.2. SB_CARRY

Inputs			Output
I0	I1	CI	CO
0	0	X	0
0	X	0	0
X	1	1	1
X	0	0	0
1	X	1	1
1	1	X	1

HDL Usage

This primitive is inferred during synthesis and can also be explicitly instantiated.

Default Signal Values

The iCEcube2 software assigns logic value '0' to unconnected input ports.

Verilog Instantiation

```
SB_CARRY my_carry_inst (
       .CO(CO),
       .I0(I0),
       .I1(I1),
       .CI(CI));
```

VHDL Instantiation

```
my_carry_inst : SB_CARRY
       port map (
       CO => CO,
       CI => CI,
       I0 => I0,
       I1 => I1
       ););
```

3. iCE40 Block RAM Primitives

The iCE40 architecture supports dual ported synchronous RAM, with 4096 bits, and a fixed 16-bit data width. The block is arranged as 256 × 16 bit words. The RAM block may be configured to be used as a RAM with data between 1-16 bits.

Each iCE40 device includes multiple high-speed synchronous RAM blocks, each 4 kbit in size. The RAM block has separate write and read ports, each with independent control signals. Each RAM block can be configured into a RAM block of size 256 × 16, 512 × 8, 1024 × 4 or 2048 × 2. The data contents of the RAM block are optionally pre-loaded during ICE device configuration.

The following table lists the supported dual port synchronous RAM configurations, each of 4 kbits in size. The RAM blocks can be directly instantiated in the top module and taken through iCube2 flow.

Table 3.1. iCE40 Block RAM

Block RAM Configuration	Block RAM Size	WADDR Port Size (Bits)	WDATA Port Size (Bits)	RADDR Port Size (Bits)	RDATA Port Size (Bits)	MASK Port Size (Bits)
SB_RAM256x16 SB_RAM256x16NR SB_RAM256x16NW SB_RAM256x16NRNW	256x16 (4K)	8 [7:0]	16 [15:0]	8 [7:0]	16 [15:0]	16 [15:0]
SB_RAM512x8 SB_RAM512x8NR SB_RAM512x8NW SB_RAM512x8NRNW	512x8 (4K)	9 [8:0]	8 [7:0]	8 [8:0]	8 [7:0]	No Mask Port
SB_RAM1024x4 SB_RAM1024x4NR SB_RAM1024x4NW SB_RAM1024x4NRNW	1024x4 (4K)	10 [9:0]	4 [3:0]	10 [9:0]	4 [3:0]	No Mask Port
SB_RAM2048x2 SB_RAM2048x2NR SB_RAM2048x2NW SB_RAM2048x2NRNW	2048x2 (4K)	11 [10:0]	2 [1:0]	10 [9:0]	2 [1:0]	No Mask Port

The Lattice Technologies convention for the iCE40 RAM primitives with negedge Read or Write clock is that the base primitive name is post fixed with N and R or W according to the clock that is affected, as displayed in the table below for 256 × 16 RAM block configuration.

Table 3.2. 256 × 16 RAM Block Configuration

RAM Primitive Name	Description
SB_RAM256x16	Posedge Read clock, Posedge Write clock
SB_RAM4256x16NR	Negedge Read clock, Posedge Write clock
SB_RAM256x16NW	Posedge Read clock, Negedge Write clock
SB_RAM256x16NRNW	Negedge Read clock, Negedge Write clock

3.1. SB_RAM256×16

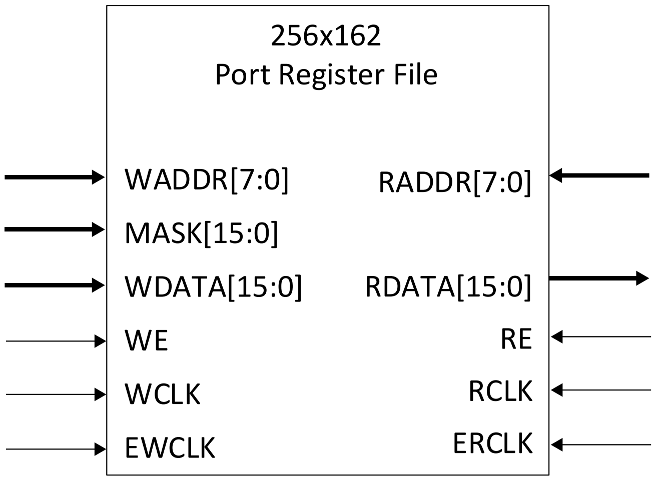

Figure 3.1. SB_RAM256×16

3.1.1. SB_RAM256x16

The following modules are the complete list of SB_RAM256x16 based primitives

SB_RAM256x16 //Posedge clock RCLK WCLK

(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM256x16 ram256x16_inst (
       .RDATA(RDATA_c[15:0]),
       .RADDR(RADDR_c[7:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[7:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[15:0]),
       .WE(WE_c),
       .MASK(MASK_c[15:0])
       );
defparam ram256x16_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram256x16_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram256x16_inst :    SB_RAM256x16 generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLK => RCLK_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLK=> WCLK_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       MASK   => MASK_c,
       WE => WE_c
      );
```

3.1.2. SB_RAM256×16NR

SB_RAM256x16NR   // Negative edged Read Clock – i.e. RCLKN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM256x16NR ram256x16NR_inst (
       .RDATA(RDATA_c[15:0]),
       .RADDR(RADDR_c[7:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[7:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[15:0]),
       .WE(WE_c),
       .MASK(MASK_c[15:0])
);
defparam ram256x16nr_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram256x16nr_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram256x16nr_inst : SB_RAM256x16NR
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLKN => RCLKN_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLK=> WCLK_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       MASK => MASK_c,
      WE =>
```

3.1.3. SB_RAM256x16NW

SB_RAM256x16NR // Negative edged Read Clock – i.e. RCLKN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM256x16NR ram256x16NR_inst (
       .RDATA(RDATA_c[15:0]),
       .RADDR(RADDR_c[7:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[7:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[15:0]),
       .WE(WE_c),
       .MASK(MASK_c[15:0])
);
defparam ram256x16nr_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram256x16nr_inst : SB_RAM256x16NR
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLKN => RCLKN_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLK=> WCLK_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       MASK => MASK_c,
       WE => WE_c
      );
```

3.1.4. SB_RAM256x16NRNW

SB_RAM256x16NRNW **// Negative edged Read and Write – i.e. RCLKN WRCKLN**
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM256x16NRNW ram256x16nrnw_inst (
       .RDATA(RDATA_c[15:0]),
       .RADDR(RADDR_c[7:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[7:0]),
       .WCLKN(WCLKN_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[15:0]),
       .WE(WE_c),
       .MASK(MASK_c[15:0])
);
defparam ram256x16nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram256x16nrnw_inst : SB_RAM256x16NRNW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (
      RDATA => RDATA_c,
      RADDR => RADDR_c,
      RCLKN => RCLKN_c,
      RCLKE => RCLKE_c,
      RE => RE_c,
      WADDR => WADDR_c,
      WCLKN=> WCLKN_c,
      WCLKE => WCLKE_c,
      WDATA => WDATA_c,
      MASK => MASK_c,
      WE => WE_c
     );
```

3.2. SB_RAM512x8

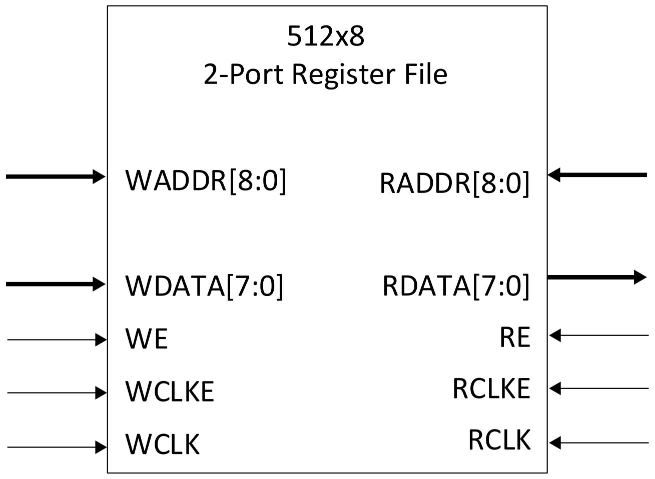

Figure 3.2. SB_RAM512×8

The following modules are the complete list of SB_RAM512x8 based primitives

3.2.1. SB_RAM512x8

SB_RAM512x8 **//Posedge clock RCLK WCLK**
(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM512x8 ram512x8_inst (
       .RDATA(RDATA_c[7:0]),
       .RADDR(RADDR_c[8:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[8:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[7:0]),
       .WE(WE_c)
       );
defparam ram512x8_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram512x8_inst : SB_RAM512x8
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
```

```
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (
RDATA => RDATA_c,
RADDR => RADDR_c,
RCLK => RCLK_c,
RCLKE => RCLKE_c,
RE => RE_c,
```

3.2.2. SB_RAM512x8NR

SB_RAM512x8NR **// Negative edged Read Clock – i.e. RCLKN**

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM512x8NRram512x8nr_inst (
       .RDATA(RDATA_c[7:0]),
       .RADDR(RADDR_c[8:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[8:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[7:0]),
       .WE(WE_c)
       );

defparam ram512x8nr_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nr_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram512x8nr_inst:    SB_RAM512x8NR
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (
      RDATA => RDATA_c, RADDR => RADDR_c, RCLKN => RCLKN_c,
      RCLKE => RCLKE_c, RE => RE_c,
      WADDR => WADDR_c, WCLK=> WCLK_c, WCLKE => WCLKE_c,
      WDATA => WDATA_c,
      WE => WE_c
     );
```

3.2.3. SB_RAM512x8NW

SB_RAM512x8NW // Negative edged Write Clock – i.e. WCLKN

(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM512x8NWram512x8nw_inst (
       .RDATA(RDATA_c[7:0]),
       .RADDR(RADDR_c[8:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[8:0]),
       .WCLKN(WCLKN_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[7:0]),
       .WE(WE_c)
       );

defparam ram512x8nw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram512x8nw_inst:    SB_RAM512x8NW
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLK => RCLK_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

3.2.4. SB_RAM512x8NRNW

SB_RAM512x8NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM512x8NRNW     ram512x8nrnw_inst (
       .RDATA(RDATA_c[7:0]),
       .RADDR(RADDR_c[8:0]),
       .RCLKN(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[8:0]),
       .WCLKN(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[7:0]),
       .WE(WE_c)
       );

defparam ram512x8nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram512x8nrnw_inst:  SB_RAM512x8NRNW
generic map ( INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLKN => RCLKN_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLKN=> WCLKN_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       WE => WE_c
     );
```

3.3. SB_RAM1024x4

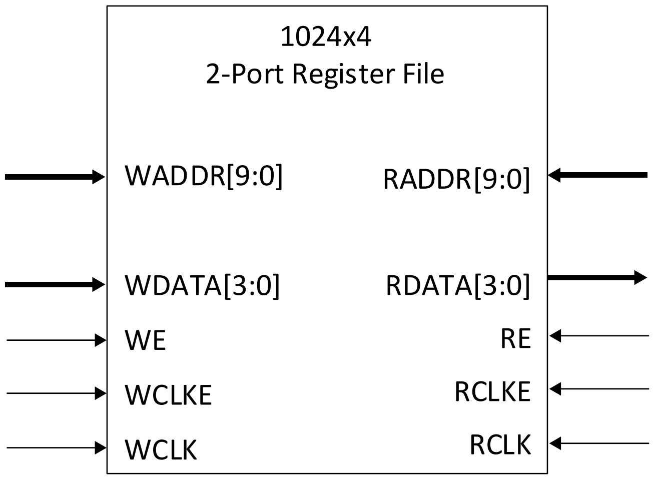

Figure 3.3. SB_RAM1024×4

The following modules are the complete list of SB_RAM1024x4 based primitives.

3.3.1. SB_RAM1024x4

SB_RAM1024x4 //Posedge clock RCLK WCLK

(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM1024x4 ram1024x4_inst (
       .RDATA(RDATA_c[3:0]),
       .RADDR(RADDR_c[9:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[3:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[9:0]),
       .WE(WE_c)
       );
defparam ram1024x4_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
Ram1024x4_inst:     SB_RAM1024x4 generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLK => RCLK_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLK=> WCLK_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

3.3.2. SB_RAM1024x4NR

SB_RAM1024x4NR // Negative edged Read Clock – i.e. RCLKN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM1024x4NR      ram1024x4nr_inst (
       .RDATA(RDATA_c[3:0]),
       .RADDR(RADDR_c[9:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[3:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[9:0]),
       .WE(WE_c)
       );
defparam ram1024x4nr_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram1024x4nr_inst:   SB_RAM1024x4NR
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLKN => RCLKN_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLK=> WCLK_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

3.3.3. SB_RAM1024x4NW

SB_RAM1024x4NW   // Negative edged Write Clock – i.e. WCLKN

(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM1024x4NW      ram1024x4nw_inst (
       .RDATA(RDATA_c[3:0]),
       .RADDR(RADDR_c[9:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[3:0]),
       .WCLKN(WCLKN_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[9:0]),
       .WE(WE_c)
       );
defparam ram1024x4_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram1024x4nw_inst :  SB_RAM1024x4NW
generic map ( INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLK => RCLK_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

3.3.4. SB_RAM1024x4NRNW

SB_RAM1024x4NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM1024x4NRNW    ram1024x4nrnw_inst (
       .RDATA(RDATA_c[3:0]),
       .RADDR(RADDR_c[9:0]),
       .RCLKN(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[3:0]),
       .WCLKN(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[9:0]),
       .WE(WE_c)
       );
defparam ram1024x4nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram1024x4nrnw_inst :      SB_RAM1024x4NRNW
generic map ( INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000", INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000", INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLKN => RCLKN_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
       );
```

3.4. SB_RAM2048x2

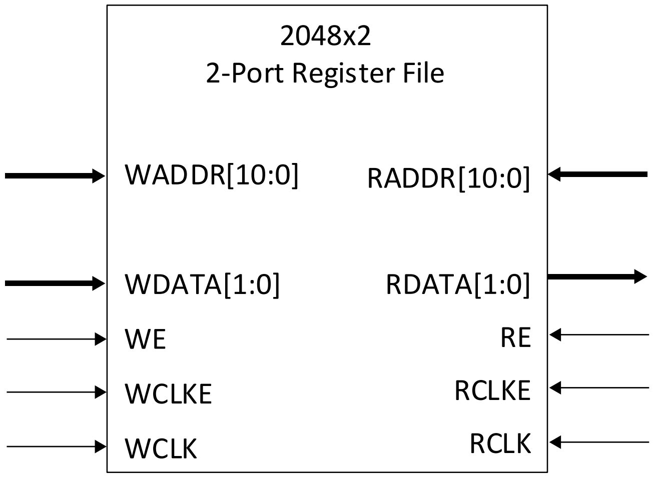

Figure 3.4. SB_RAM2048×2

The following modules are the complete list of SB_RAM2048x2 based primitives.

3.4.1. SB_RAM2048x2

SB_RAM2048x2 //Posedge clock RCLK WCLK

(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM2048x2 ram2048x2_inst (
       .RDATA(RDATA_c[1:0]),
       .RADDR(RADDR_c[10:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[10:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[1:0]),
       .WE(WE_c)
       );
defparam ram2048x2_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
Ram2048x2_inst : SB_RAM2048x2
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLK => RCLK_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLK=> WCLK_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       WE => WE_c
      );
```

3.4.2. SB_RAM2048x2NR

SB_RAM2048x2NR // Negative edged Read Clock – i.e. RCLKN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation

```
SB_RAM2048x2NR ram2048x2nr_inst (
       .RDATA(RDATA_c[1:0]),
       .RADDR(RADDR_c[10:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[10:0]),
       .WCLK(WCLK_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[1:0]),
       .WE(WE_c)
       );
defparam ram2048x2nr_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram2048x2nr_inst : SB_RAM2048x2NR
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLKN => RCLKN_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLK=> WCLK_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       WE => WE_c
      );
```

3.4.3. SB_RAM2048x2NW

SB_RAM2048x2NW // Negative edged Write Clock – i.e. WCLKN

(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation

```
SB_RAM2048x2NW ram2048x2nw_inst (
       .RDATA(RDATA_c[1:0]),
       .RADDR(RADDR_c[10:0]),
       .RCLK(RCLK_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[10:0]),
       .WCLKN(WCLKN_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[1:0]),
       .WE(WE_c)
       );
defparam ram2048x2nw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram2048x2nw_inst: SB_RAM2048x2NW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
       RDATA => RDATA_c,
       RADDR => RADDR_c,
       RCLK => RCLK_c,
       RCLKE => RCLKE_c,
       RE => RE_c,
       WADDR => WADDR_c,
       WCLKN=> WCLKN_c,
       WCLKE => WCLKE_c,
       WDATA => WDATA_c,
       WE => WE_c
      );
```

3.4.4. SB_RAM2048x2NRNW

SB_RAM2048x2NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN

(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

```
SB_RAM2048x2NRNW ram2048x2nrnw_inst (
       .RDATA(RDATA_c[1:0]),
       .RADDR(RADDR_c[10:0]),
       .RCLKN(RCLKN_c),
       .RCLKE(RCLKE_c),
       .RE(RE_c),
       .WADDR(WADDR_c[10:0]),
       .WCLKN(WCLKN_c),
       .WCLKE(WCLKE_c),
       .WDATA(WDATA_c[1:0]),
       .WE(WE_c)
       );
defparam ram2048x2nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
ICE Technology Library 82
Lattice Semiconductor Corporation Confidential
defparam ram2048x2nrnw_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram2048x2nrnw_inst : SB_RAM2048x2NRNW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
       )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLKN => RCLKN_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
       );
```

3.5. SB_RAM40_4K

SB_RAM40_4K is the basic physical RAM primitive which can be instantiated and configured to different depth and dataports. The SB_RAM40_4K block has a size of 4 Kbits with separate write and read ports, each with independent control signals. By default, input and output data is 16 bits wide, although the data width is configurable using the READ_MODE and WRITE_MODE parameters. The data contents of the SB_RAM40_4K block are optionally pre-loaded during iCE device configuration.

Table 3.3. SB_RAM40_4K Naming Convention Rules

RAM Primitive Name	Description
SB_RAM40_4K	Posedge Read clock, Posedge Write clock
SB_RAM40_4KNR	Negedge Read clock, Posedge Write clock
SB_RAM40_4KNW	Posedge Read clock, Negedge Write clock
SB_RAM40_4KNRNW	Negedge Read clock, Negedge Write clock

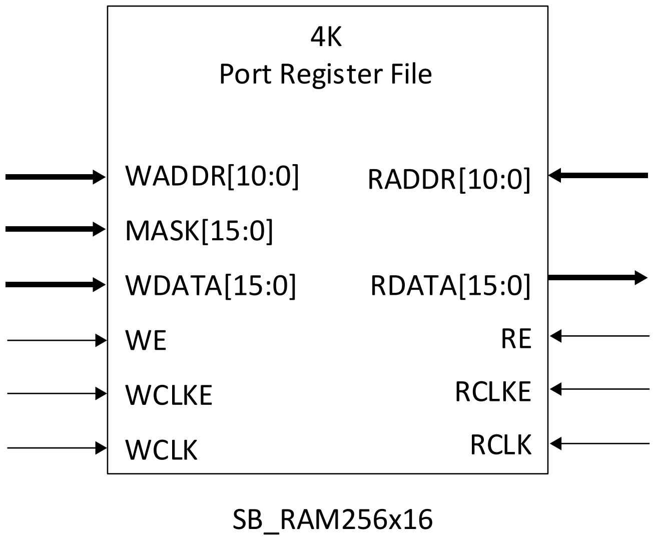

Figure 3.5. SB_RAM40_4K

The following table lists the signals for both ports.

Table 3.4. SB_RAM40_4K RAM Port Signals

Signal Name	Direction	Description
WDATA[15:0]	Input	Write Data input
MASK[15:0]*	Input	Bit-line Write Enable input, active low. Applicable only when WRITE_MODE parameter is set to 0.
WADDR[7:0]	Input	Write Address input. Selects up to 256 possible locations
WE	Input	Write Enable input, active high
WCLK	Input	Write Clock input, rising-edge active
WCLKE	Input	Write Clock Enable input
RDATA[15:0]	Output	Read Data output
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible locations
RE	Input	Read Enable input, active high
RCLK	Input	Read Clock input, rising-edge active
RCLKE	Input	Read Clock Enable input

Table 3.5. SB_RAM40_4K RAM Parameters

Parameter Name	Description	Parameter Value	Configuration
INIT_0, … …,INIT_F	RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16 × 256 = 4096 total bits)	INIT_0 to INIT_F	Initialize the RAM with predefined value
WRITE_MODE	Sets the RAM block write port configuration	0	256x16
		1	512x8
		2	1024x4
		3	2048x2
READ_MODE	Sets the RAM block read port configuration	0	256x16
		1	512x8
		2	1024x4
		3	2048x2

Verilog Instantiation:

```
// Physical RAM Instance without Pre Initialization
SB_RAM40_4K ram40_4kinst_physical (
       .RDATA(RDATA),
       .RADDR(RADDR),
       .WADDR(WADDR),
       .MASK(MASK),
       .WDATA(WDATA),
       .RCLKE(RCLKE),
       .RCLK(RCLK),
       .RE(RE),
       .WCLKE(WCLKE),
       .WCLK(WCLK),
       .WE(WE)
    );
defparam ram40_4kinst_physical.READ_MODE=0;
defparam ram40_4kinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

```
-- Physical RAM Instance without Pre Initialization
ram40_4kinst_physical : SB_RAM40_4K
generic map (
READ_MODE => 0,
WRITE_MODE= >0 )
port map (
        RDATA=>RDATA,
        RADDR=>RADDR,
        WADDR=>WADDR,
        MASK=>MASK,
        WDATA=>WDATA,
        RCLKE=>RCLKE,
        RCLK=>RCLK,
        RE=>RE,
        WCLKE=>WCLKE,
        WCLK=>WCLK,
        WE=>WE
       );
```

3.5.1. SB_RAM40_4KNR

Verilog Instantiation:

```
// Physical RAM Instance without Pre Initialization
SB_RAM40_4KNR ram40_4knrinst_physical (
       .RDATA(RDATA),
       .RADDR(RADDR),
       .WADDR(WADDR),
       .MASK(MASK),
       .WDATA(WDATA),
       .RCLKE(RCLKE),
       .RCLKN(RCLKN),
       .RE(RE),
       .WCLKE(WCLKE),
       .WCLK(WCLK),
       .WE(WE)
   );
defparam ram40_4knrinst_physical.READ_MODE=0;
defparam ram40_4knrinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

```
-- Physical RAM Instance without Pre Initialization
ram40_4knrinst_physical : SB_RAM40_4KNR
generic map (
READ_MODE => 0,
WRITE_MODE= >0
                     )
port map (
       RDATA=>RDATA,
       RADDR=>RADDR,
       WADDR=>WADDR,
       MASK=>MASK,
       WDATA=>WDATA,
       RCLKE=>RCLKE,
       RCLKN=>RCLKN,
       RE=>RE,
       WCLKE=>WCLKE,
       WCLK=>WCLK,
       WE=>WE
      );
```

3.5.2. SB_RAM40_4KNW

Verilog Instantiation:

```
// Physical RAM Instance without Pre Initialization
SB_RAM40_4KNW ram40_4knwinst_physical (
       .RDATA(RDATA),
       .RADDR(RADDR),
       .WADDR(WADDR),
       .MASK(MASK),
       .WDATA(WDATA),
       .RCLKE(RCLKE),
       .RCLK(RCLK),
       .RE(RE),
       .WCLKE(WCLKE),
       .WCLKN(WCLKN),
       .WE(WE)
       );
defparam ram40_4knwinst_physical.READ_MODE=0;
defparam ram40_4knwinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

```
-- Physical RAM Instance without Pre Initialization
ram40_4knwinst_physical : SB_RAM40_4KNW
generic map (
       READ_MODE => 0,
       WRITE_MODE= >0
                          )
port map (
        RDATA=>RDATA,
        RADDR=>RADDR,
        WADDR=>WADDR,
        MASK=>MASK,
        WDATA=>WDATA,
        RCLKE=>RCLKE,
        RCLK=>RCLK,
        RE=>RE,
        WCLKE=>WCLKE,
        WCLKN=>WCLKN,
        WE=>WE
      );
```

3.5.3. SB_RAM40_4KNRNW

Verilog Instantiation:

```
// Physical RAM Instance without Pre Initialization
SB_RAM40_4KNRNW ram40_4knrnwinst_physical (
       .RDATA(RDATA),
       .RADDR(RADDR),
       .WADDR(WADDR),
       .MASK(MASK),
       .WDATA(WDATA),
       .RCLKE(RCLKE),
       .RCLKN(RCLKN),
       .RE(RE),
       .WCLKE(WCLKE),
       .WCLKN(WCLKN),
       .WE(WE)
       );
defparam ram40_4knrnwinst_physical.READ_MODE=0;
defparam ram40_4knrnwinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

```
-- Physical RAM Instance without Pre Initialization
ram40_4knrnwinst_physical : SB_RAM40_4KNRNW
generic map (
READ_MODE => 0,
WRITE_MODE= >0
                   )
port map (
        RDATA=>RDATA,
        RADDR=>RADDR,
        WADDR=>WADDR,
        MASK=>MASK,
        WDATA=>WDATA,
        RCLKE=>RCLKE,
        RCLKN=>RCLKN,
        RE=>RE,
        WCLKE=>WCLKE,
        WCLKN=>WCLKN,
        WE=>WE
       );
```

4. I/O Primitives

4.1. SB_IO

The SB_IO block contains five registers. The following figure and Verilog template illustrate the complete user accessible logic diagram, and its Verilog instantiation.

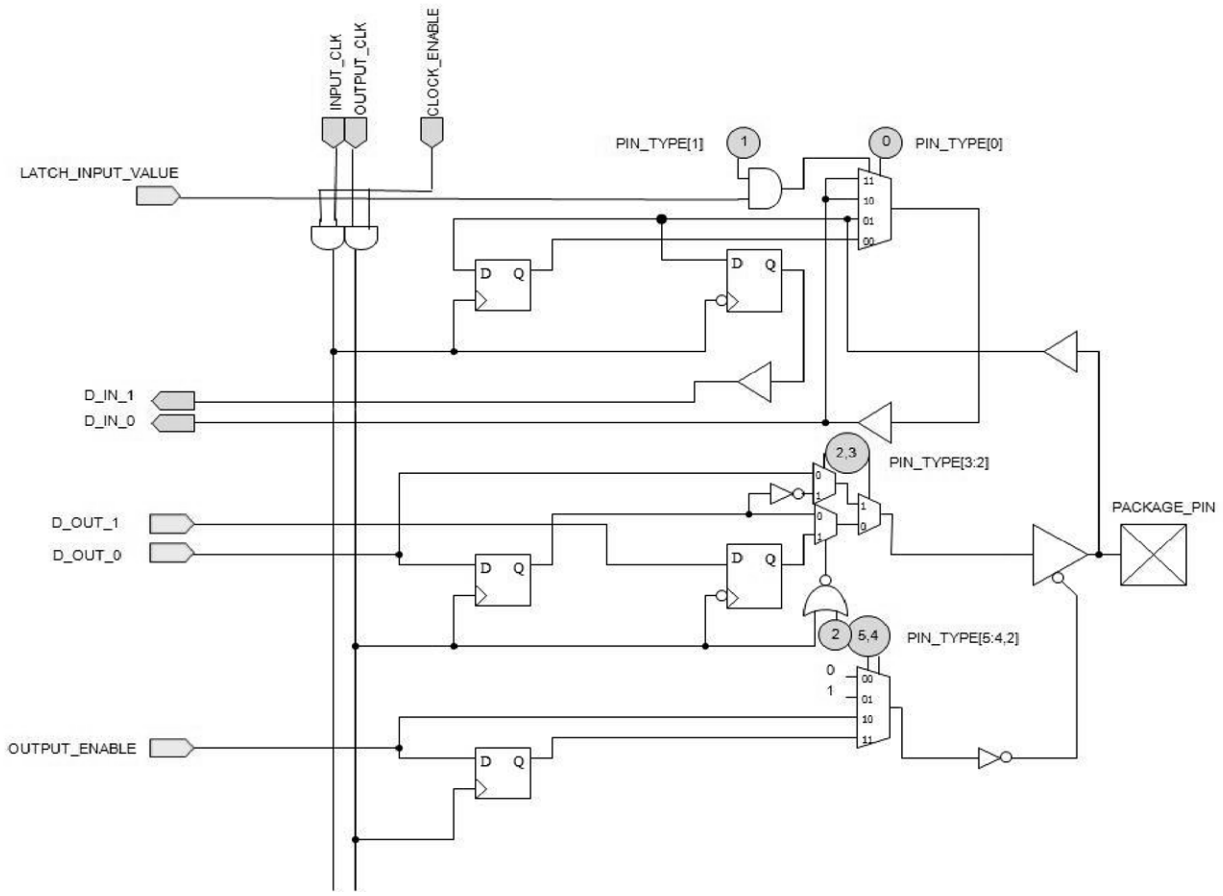

Figure 4.1. SB_IO

VHDL Code

```
COMPONENT SB_IO IS

       GENERIC(
       PIN_TYPE : std_logic_vector(5 downto 0) := "000000";
       IO_STANDARD: string := "SB_LVCMOS";
       NEG_TRIGGER: std_logic_vector := '0' ;
       PULLUP: std_logic_vector := '0'
       );
       PORT(
       PACKAGE_PIN : in std_logic;
       LATCH_INPUT_VALUE : in std_logic;
       CLOCK_ENABLE : in std_logic;
       INPUT_CLK : in std_logic;
       OUTPUT_CLK : in std_logic;
       OUTPUT_ENABLE : in std_logic;
       D_OUT_0 : in std_logic;
       D_OUT_1 : in std_logic;
       D_IN_0 : out std_logic;
       D_IN_1 : out std_logic
       );
       END COMPONENT SB_IO;
       attribute DRIVE_STRENGTH : string;
       attribute DRIVE_STRENGTH of SB_IO : component is "x2";

--COMPONENT INSTANTIATION
       uut : SB_GB_IO
       generic map (
              PIN_TYPE => "000000",
              IO_STANDARD => "SB_LVCMOS",
              NEG_TRIGGER => '0' ,
              PULLUP => '0'
       )
       port map(
       -- enter port connections here.
              PACKAGE_PIN              => Package_Pin,
              CLOCK_ENABLE       => clock_enable,
              LATCH_INPUT_VALUE  => latch_input_value,
              INPUT_CLK                => input_clk,
              OUTPUT_CLK               => output_clk,
              OUTPUT_ENABLE      => output_enable,
              D_OUT_1                => d_out_0,
              D_OUT_0                => d_out_1,
              D_IN_1                   => d_in_1,
              D_IN_0                   => d_in_0
       );
```

Default Signal Values

The iCEcube2 software assigns the logic ‘0’ value to all unconnected input ports except for CLOCK_ENABLE.

Note that explicitly connecting a logic ‘1’ value to port CLOCK_ENABLE results in a non-optimal implementation, since an extra LUT is used to generate the Logic ‘1’. If your intention is to keep the Input and Output registers always enabled, it is recommended that port CLOCK_ENABLE be left unconnected.

Input and Output Pin Function Tables

Input and Output functions are independently selectable through the PIN_TYPE [1:0] and PIN_TYPE [5:2] parameter settings respectively. Specific I/O functions are defined by the combination of both attributes. This means that the complete number of combinations is 64, although some combinations are not valid and not defined below.

Note that the selection of I/O Standards such as SSTL and LVCMOS are not defined by these tables.

Table 4.1. Input Pin Function

#	Pin Function Mnemonic	PIN_TYPE[1:0]		Functional Description of Package Pin Input Operation
1	PIN_INPUT	0	1	Simple input pin (D_IN_0)
2	PIN_INPUT_LATCH	1	1	Disables internal data changes on the physical input pin by latching the value.
3	PIN_INPUT_REGISTERED	0	0	Input data is registered in input cell.
4	PIN_INPUT_REGISTERED_LATCH	1	0	Disables internal data changes on the physical input pin by latching the value on the input register.
5	PIN_INPUT_DDR	0	0	Input 'DDR' data is clocked out on rising and falling clock edges. Use the D_IN_0 and D_IN_1 pins for DDR operation.

Table 4.2. Output Pin Function

#	Pin Function Mnemonic	PIN_TYPE[5:2]				Functional Description of Package Pin Output Operation
1	PIN_NO_OUTPUT	0	0	0	0	Disables the output function
2	PIN_OUTPUT	0	1	1	0	Simple output pin, (no enable)
3	PIN_OUTPUT_TRISTATE	1	0	1	0	The output pin may be tristated using the enable
4	PIN_OUTPUT_ENABLE_REGISTERED	1	1	1	0	The output pin may be tristated using a registered enable signal
5	PIN_OUTPUT_REGISTERED	0	1	0	1	Output registered, (no enable)
6	PIN_OUTPUT_REGISTERED_ENABLE	1	0	0	1	Output registered with enable (enable is not registered)
7	PIN_OUTPUT_REGISTERED_ENABLE_ REGISTERED	1	1	0	1	Output registered and enable registered
8	PIN_OUTPUT_DDR	0	1	0	0	Output 'DDR' data is clocked out on rising and falling clock edges
9	PIN_OUTPUT_DDR_ENABLE	1	0	0	0	Output data is clocked out on rising and falling clock edges
10	PIN_OUTPUT_DDR_ENABLE_ REGISTERED	1	1	0	0	Output 'DDR' data with registered enable signal
11	PIN_OUTPUT_REGISTERED_INVERTE D	0	1	1	1	Output registered signal is inverted
12	PIN_OUTPUT_REGISTERED_ENABLE INVERTED	1	0	1	1	Output signal is registered and inverted, (no enable function)
13	PIN_OUTPUT_REGISTERED_ENABLE_ REGISTERED_INVERTED	1	1	1	1	Output signal is registered and inverted, the enable/tristate control is registered.

Syntax Verilog Use

Output Pin Function is the bit vector associated with PIN_TYPE [5:2] and Input Pin Function is the bit vector associated with PIN_TYPE [1:0], resulting in a 6-bit value PIN_TYPE [5:0]

```
defparam my_generic_IO.PIN_TYPE = 6’b{Output Pin Function, Input Pin Function};
```

4.1.1. DDR I/O Configuration

The following setting configures the SB_IO into a DDR I/O.

```
             defparam my_DDR_IO.PIN_TYPE = 6’b100000;

                          // PIN_TYPE [5:2] = 1000
                          // PIN_TYPE [1:0] = 00
```

This creates a DDR I/O pin whereby the input data is clocked in on both the rising and falling input clock edges.

The output 'DDR' data is clocked out on rising and falling output clock edges, and the output may be tri-stated, using the output enable port of the SB_IO.

4.1.2. High Drive SB_IO

I/O in iCE40/iCE40LM device can be configured with different drive strengths to increase the I/O output current. To configure an SB_IO with specific drive value, the user needs to specify the “DRIVE_STRENGTH” synthesis attribute on the SB_IO instance and the I/O should be configured as output-only registered I/O.

Synthesis Attribute Syntax:

```
/* synthesis DRIVE_STRENGTH = <Drive value> */
```

Table 4.3. Drive Value

Drive Strength Value	Description
x1	Default drive strength. No replication of SB_IO.
x2	Increase default drive strength by 2. SB_IO replicated once.
x3	Increase default drive strength by 3. SB_IO replicated twice.

Note: High drive SB_IO is available only in selected iCE40/iCE40LM packages. Refer to Chapter 12 in iCEcube2_userguide for the list of supported device packages.

4.1.3. Pull Up Resistor Configuration

For iCE40UL device, you can configure the internal pull-up resistor strength of an I/O to a predefined resistor value through the attribute settings. By default, all the I/O are weakly pulled up by 100k internal resistor.

The PULLUP_RESISTOR attribute is effective only when PULLUP parameter is set to 1.

Synthesis Attribute Syntax:

```
/* synthesis PULLUP_RESISTOR = "3P3K" */
```

Table 4.4. Pull-up Resistor Value

Drive Strength Value	Description
"3P3K"	Pull-up resistor level is 3.3 kΩ.
"6P8K"	Pull-up resistor level is 6.8 kΩ.
"10K"	Pull-up resistor level is 10 kΩ.
"100K"	Pull-up resistor level is 100 kΩ (default).

Note: PULLUP_RESISTOR attribute is supported only for iCE40UL devices.

Verilog Instantiation

```
SB_IO         IO_PIN_INST
(
.PACKAGE_PIN (Package_Pin),                   //     User's Pin signal name
.LATCH_INPUT_VALUE (latch_input_value),//     Latches/holds the Input value
.CLOCK_ENABLE (clock_enable),                 //     Clock Enable common to input and
                                              //     output clock
.INPUT_CLK (input_clk),                       //     Clock for the input registers
.OUTPUT_CLK (output_clk),               //     Clock for the output registers
.OUTPUT_ENABLE (output_enable),         //     Output Pin Tristate/Enable
                                              //     control
.D_OUT_0 (d_out_0),                     //     Data 0 – out to Pin/Rising clk
                                              //     edge
.D_OUT_1 (d_out_1),                     //     Data 1 - out to Pin/Falling clk
                                              //     edge
.D_IN_0 (d_in_0),                             //     Data 0 - Pin input/Rising clk
                                              //     edge
.D_IN_1 (d_in_1)                              //     Data 1 – Pin input/Falling clk
                                              //     edge
) /* synthesis DRIVE_STRENGTH= x2 */;

defparam IO_PIN_INST.PIN_TYPE = 6'b000000;
                                  // See Input and Output Pin Function Tables.
                                  // Default value of PIN_TYPE = 6'000000 i.e.
                                  // an input pad, with the input signal
                                  // registered.
defparam IO_PIN_INST.PULLUP = 1'b0;
                                  // By default, the IO will have NO pull up.
                                  // This parameter is used only on bank 0, 1,
                                  // and 2. Ignored when it is placed at bank 3 defparam
IO_PIN_INST.NEG_TRIGGER = 1'b0;
                                  // Specify the polarity of all FFs in the IO to
                                  // be falling edge when NEG_TRIGGER = 1.
                                  // Default is rising edge.
defparam IO_PIN_INST.IO_STANDARD = "SB_LVCMOS";
                                  // Other IO standards are supported in bank 3
                                  // only: SB_SSTL2_CLASS_2, SB_SSTL2_CLASS_1,
                                  // SB_SSTL18_FULL, SB_SSTL18_HALF, SB_MDDR10,
                                  // SB_MDDR8, SB_MDDR4, SB_MDDR2 etc.
```

5. Global Buffer Primitives

5.1. SB_GB_IO

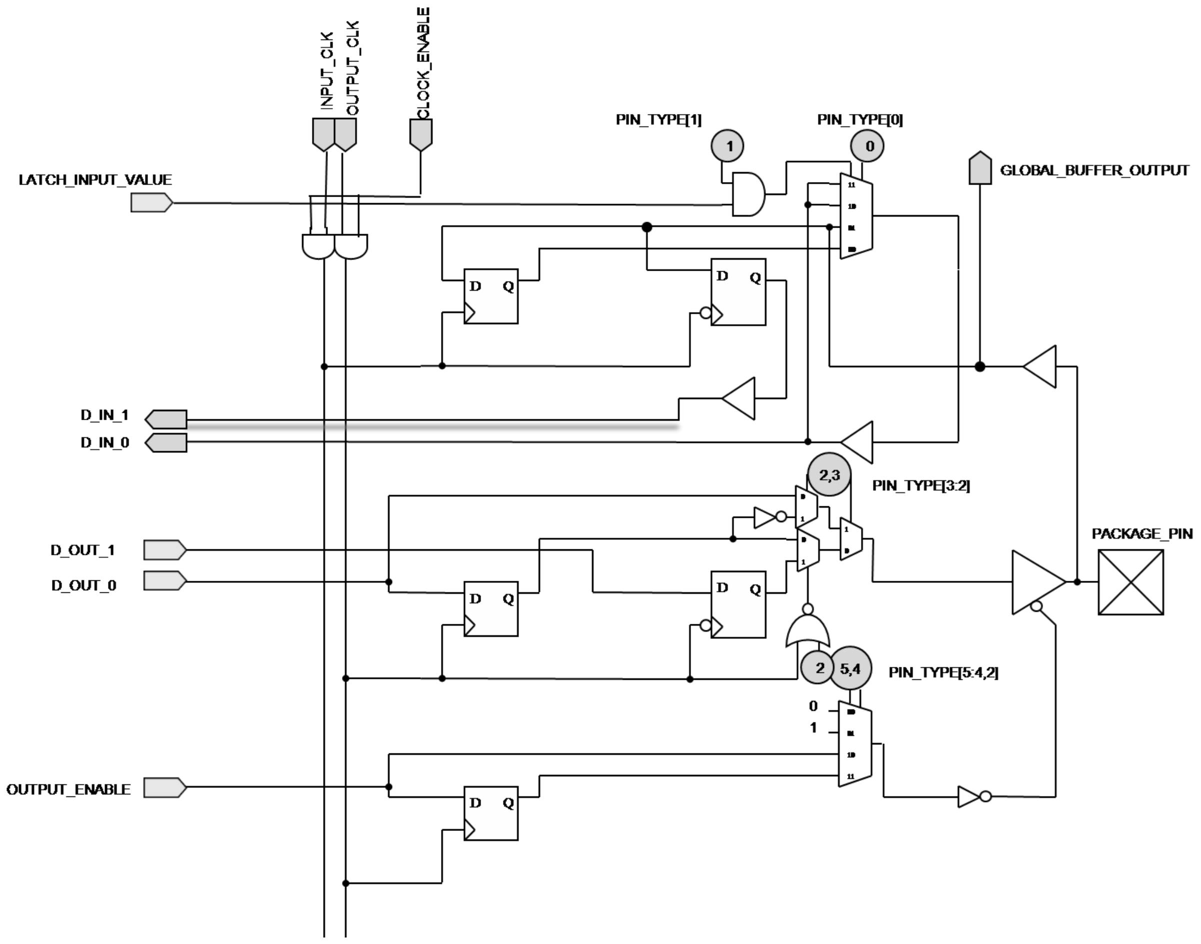

Figure 5.1. SB_GB_IO

VHDL Code

```
--COMPONENT DECLARATION
      COMPONENT SB_GB_IO IS
      GENERIC(
      PIN_TYPE : std_logic_vector(5 downto 0) := "000000"
      );
      PORT(
      PACKAGE_PIN : in std_logic;
      LATCH_INPUT_VALUE : in std_logic;
      CLOCK_ENABLE : in std_logic;
      INPUT_CLK : in std_logic;
      OUTPUT_CLK : in std_logic;
      OUTPUT_ENABLE : in std_logic;
      D_OUT_0 : in std_logic;
      D_OUT_1 : in std_logic;
      D_IN_0 : out std_logic;
      D_IN_1 : out std_logic;
      GLOBAL_BUFFER_OUTPUT: out std_logic
      );
      END COMPONENT SB_IO;


--COMPONENT INSTANTIATION
      uut : SB_GB_IO
      generic map (
            PIN_TYPE => "000000"
      )
      port map(
      -- enter port connections here.
            PACKAGE_PIN               => Package_Pin
            CLOCK_ENABLE        => clock_enable,
            LATCH_INPUT_VALUE   => latch_input_value,
            INPUT_CLK                 => input_clk,
            OUTPUT_CLK                => output_clk,
            OUTPUT_ENABLE       => output_enable,
            D_OUT_1                 => d_out_0,
            D_OUT_0                 => d_out_1,
            D_IN_1                    => d_in_1,
            D_IN_0                    => d_in_0,
            GLOBAL_BUFFER_OUTPUT => Global_Buffered_User_Clock
      );
```

Default Signal Values

The iCEcube2 software assigns the logic ‘0’ value to all unconnected input ports except for CLOCK_ENABLE.

Note: Explicitly connecting a logic ‘1’ value to port CLOCK_ENABLE results in a non-optimal implementation, since an extra LUT is used to generate the Logic ‘1’. If your intention is to keep the Input and Output registers always enabled, it is recommended that port CLOCK_ENABLE be left unconnected.

Verilog Instantiation

```
SB_GB_IO My_Clock_Buffer_Package_Pin (  // A users external Clock reference pin
.PACKAGE_PIN (Package_Pin),              // User’s Pin signal name
.LATCH_INPUT_VALUE (latch_input_value),  // Latches/holds the Input value
.CLOCK_ENABLE (clock_enable),            // Clock Enable common to input and
// output clock
.INPUT_CLK (input_clk),   //     Clock for the input registers
.OUTPUT_CLK (output_clk), //     Clock for the output registers
.OUTPUT_ENABLE (output_enable),  //     Output Pin Tristate/Enable
      //     control
.D_OUT_0 (d_out_0), //     Data 0 – out to Pin/Rising clk
      //     edge
.D_OUT_1 (d_out_1), //     Data 1 - out to Pin/Falling clk
      //     edge
.D_IN_0 (d_in_0),   //     Data 0 - Pin input/Rising clk
      //     edge
.D_IN_1 (d_in_1)    //     Data 1 – Pin input/Falling clk
      //     edge
.GLOBAL_BUFFER_OUTPUT (Global_Buffered_User_Clock)
// Example use – clock buffer
//driven from the input pin
);

defparam     My_Clock_Buffer_Package_Pin.PIN_TYPE = 6'b000000;
// See Input and Output Pin Function Tables.
// Default value of PIN_TYPE = 6’000000 i.e.
// an input pad, with the input signal
// registered
```

Note: This primitive is a superset of the SB_IO primitive, and includes the connectivity to drive a Global Buffer. For example SB_GB_IO pins are likely to be used for external Clocks.

5.2. SB_GB Primitive

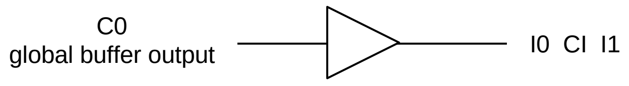

Figure 5.2. SB_GB Primitive

Verilog Instantiation

```
SB_GB My_Global_Buffer_i (//Required for a user’s internally generated
                                 //FPGA signal that is heavily loaded and
                                 //requires global buffering. For example, a
                                 //user’s logic-generated clock.

.USER_SIGNAL_TO_GLOBAL_BUFFER (Users_internal_Clk),
.GLOBAL_BUFFER_OUTPUT ( Global_Buffered_User_Signal)

);
```

VHDL code

```
COMPONENT SB_GB
       PORT(
       USER_SIGNAL_TO_GLOBAL_BUFFER : in std_logic;
       GLOBAL_BUFFER_OUTPUT: out std_logic
       );
       END COMPONENT SB_IO;


--COMPONENT INSTANTIATION
       uut : SB_GB
       port map(
       -- enter port connections here.
              USER_SIGNAL_TO_GLOBAL_BUFFER => Users_internal_clk,
              GLOBAL_BUFFER_OUTPUT => Global_Buffered_User_Clock
       );
```

6. PLL Primitives

There are five primitives that represent the PLL function in the iCEcube2 software viz. SB_PLL40_CORE, SB_PLL40_PAD, SB_PLL40_2_PAD, SB_PLL40_2F_CORE and SB_PLL40_2F_PAD for the iCE40 device family. A short description of each primitive and its ports/parameters is provided in the following sections.

It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

6.1. SB_PLL40_CORE

The SB_PLL40_CORE primitive should be used when the source clock of the PLL is driven by FPGA routing, that is, when the PLL source clock originates on the FPGA or is driven by an input pad that is not in the bottom I/O bank (I/O Bank 2).

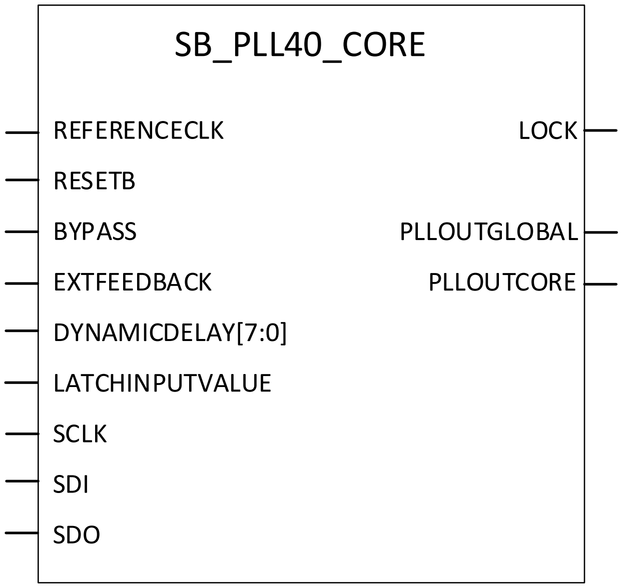

Figure 6.1. SB_PLL40_CORE

Ports

REFERENCECLK: PLL source clock that serves as the input to the SB_PLL40_CORE primitive.

PLLOUTGLOBAL: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCORE: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBAL port.

LOCK: Output port, when HIGH, indicates that the signal on PLLOUTGLOBAL/PLLOUTCORE is locked to the PLL source on REFERENCECLK.

EXTFEEDBACK: External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.

DYNAMICDELAY: 7-bit input bus that enables dynamic control of the delay contributed by the Fine Delay Adjust Block. The Fine Delay Adjust Block is used when there is a need to adjust the phase alignment of PLLOUTGLOBAL/PLLOUTCORE with respect to REFERENCECLK. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.

RESETB: Active low input that asynchronously resets the PLL.

BYPASS: Input signal, when asserted, connects the signal on REFERENCECLK to PLLOUTCORE/PLLOUTGLOBAL pins.

LATCHINPUTVALUE: Active high input, when enabled, forces the PLL into low-power mode. The PLLOUTGLOBAL/PLLOUTCORE pins are held static at their last value. This function is enabled when the parameter ENABLE_ICEGATE is set to ‘1’.

SCLK, SDI, SDO: These pins are used only for internal testing purposes, and need not be instantiated by users.

Parameters

The SB_PLL40_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.1. SB_PLL40_CORE Parameters

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL and PLLOUTCORE signals are delay compensated by (n+1)×150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are additionally delayed by (n+1)×150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0-> Divide by 4 1-> Divide by 7

Parameter Name	Description	Parameter Value	Description
PLLOUT_SELECT	Selects the signal to be output at the PLLOUTCORE and PLLOUTGLOBAL ports	SHIFTREG_0deg	0o phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs without any phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACT OR	Divide-by factor of a divider in external feedback path	User-specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

6.2. SB_PLL40_PAD

The SB_PLL40_PAD primitive should be used when the source clock of the PLL is driven by an input pad that is located in the bottom I/O bank (I/O Bank 2) or the top I/O bank (I/O Bank 0), and the source clock is not required inside the FPGA.

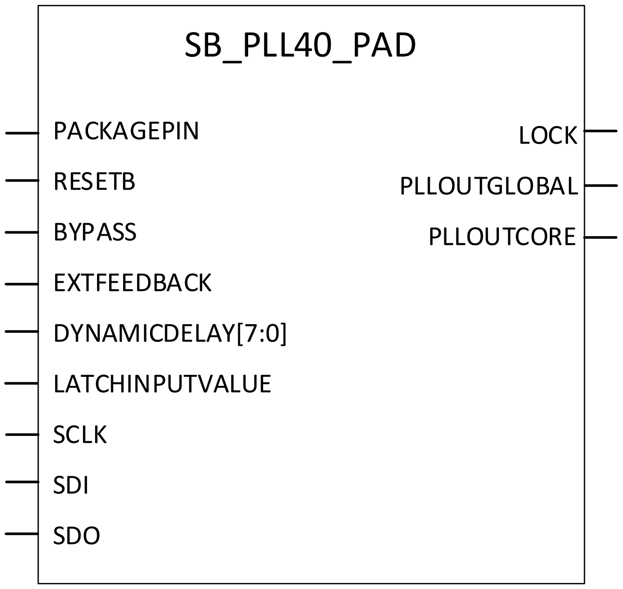

Figure 6.2. SB_PLL40_PAD

Ports

PACKAGEPIN: PLL REFERENCE CLOCK pin that serves as the input to the SB_PLL40_PAD primitive.

PLLOUTGLOBAL: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCORE: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBAL port.

LOCK: Output port, when HIGH, indicates that the signal on PLLOUTGLOBAL/PLLOUTCORE is locked to the PLL source on PACKAGEPIN.

EXTFEEDBACK: External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.

DYNAMICDELAY: 7-bit input bus that enables dynamic control of the delay contributed by the Fine Delay Adjust Block. The Fine Delay Adjust Block is used when there is a need to adjust the phase alignment of PLLOUTGLOBAL/PLLOUTCORE with respect to PACKAGEPIN. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.

RESETB: Active low input that asynchronously resets the PLL.

BYPASS: Input signal, when asserted, connects the signal on PACKAGEPIN to

PLLOUTCORE/PLLOUTGLOBAL pins.

LATCHINPUTVALUE: Active high input, when enabled, forces the PLL into low-power mode. The PLLOUTGLOBAL/PLLOUTCORE pins are held static at their last value. This function is enabled when the parameter ENABLE_ICEGATE is set to '1'.

SCLK, SDI, SDO: These pins are used only for internal testing purposes, and need not be instantiated by users.

Parameters

The SB_PLL40_PAD primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.2. SB_PLL40_PAD Parameters

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY [3:0] pins

Parameter Name	Description	Parameter Value	Description
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL and PLLOUTCORE signals are delay compensated by (n+1) ×150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_F EEDBACK is FIXED.
DELAY_ADJUSTMENT_MO DE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are additionally delayed by (n+1)×150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_ RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0 -> Divide by 4 1 -> Divide by 7
PLLOUT_SELECT	Selects the signal to be output at the PLLOUTCORE and PLLOUTGLOBAL ports	SHIFTREG_0deg	0o phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs without any phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to
DIVF	Feedback divider	0,1,..,63	control the output frequency, depending on the FEEDBACK_PATH setting.
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACT OR	Divide-by factor of a divider in external feedback path	User-specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input
DIVF	Feedback divider	0,1,..,63	Control the output frequency, depending on the FEEDBACK_PATH setting.
DIVQ	VCO Divider	1,2,…,6	

Parameter Name	Description	Parameter Value	Description
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACT OR	Divide-by factor of a divider in external feedback path	User-specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

6.3. SB_PLL40_2_PAD

The SB_PLL40_2_PAD primitive should be used when the source clock of the PLL is driven by an input pad that is located in the bottom I/O bank (I/O Bank 2) or the top I/O bank (I/O Bank 0), and in addition to the PLL output, the source clock is also required inside the FPGA.

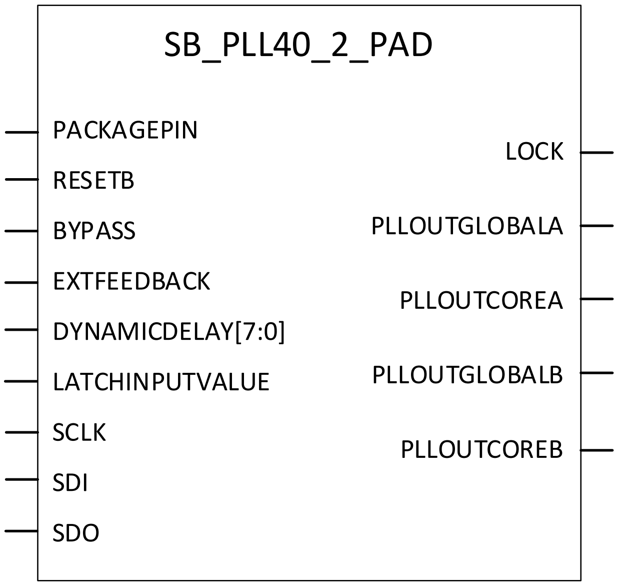

Figure 6.3. SB_PLL40_2_PAD

Ports

PACKAGEPIN: PLL REFERENCE CLOCK pin that serves as the input to the SB_PLL40_2_PAD primitive.

PLLOUTGLOBALA: The signal on PACKAGEPIN appears on the FPGA at this pin, and drives a global clock network on the FPGA. Do not use this pin in an external feedback path to the PLL.

PLLOUTCOREA: The signal on PACKAGEPIN appears on the FPGA at this pin, which drives regular

FPGA routing. Do not use this pin in an external feedback path to the PLL.

PLLOUTGLOBALB: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCOREB: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBAL port.

LOCK: Output port, when HIGH, indicates that the signal on PLLOUTGLOBALB/PLLOUTCOREB is locked to the PLL source on PACKAGEPIN.

EXTFEEDBACK: External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.

DYNAMICDELAY: 4-bit input bus that enables dynamic control of the delay contributed by the Fine Delay Adjust Block. The Fine Delay Adjust Block is used when there is a need to adjust the phase alignment of

PLLOUTGLOBAL/PLLOUTCORE with respect to PACKAGEPIN. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.

RESET: Active low input that asynchronously resets the PLL.

BYPASS: Input signal, when asserted, connects the signal on PACKAGEPIN to

PLLOUTCORE/PLLOUTGLOBAL pins.

LATCHINPUTVALUE: Active high input, when enabled, forces the PLL into low-power mode. The PLLOUTGLOBALA/PLLOUTCOREA pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTA is set to '1', and the LATCHINPUTVALUE signal is asserted. The PLLOUTGLOBALB/PLLOUTCOREB pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTB is set to '1', and the LATCHINPUTVALUE signal is asserted.

SCLK, SDI, SDO: These pins are used only for internal testing purposes, and need not be instantiated by users.

Parameters

The SB_PLL40_2_PAD primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.3. SB_PLL40_2_PAD Parameters

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE _FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL and PLLOUTCORE signals are delay compensated by (n+1)×150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_ FEEDBACK is FIXED.

Parameter Name	Description	Parameter Value	Description
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)×150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0 -> Divide by 4 1-> Divide by 7
PLLOUT_SELECT_PORTB	Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports	SHIFTREG_0deg	0^o phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs to PortB. No phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output to PORTB. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User-specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE_PORTA	Enables the PLL power- down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input
ENABLE_ICEGATE_PORTB	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

6.4. SB_PLL40_2F_CORE

The SB_PLL40_2F_CORE primitive should be used when PLL is used to generate two different output frequencies, and the source clock of the PLL is driven by FPGA routing. For example, when the PLL source clock originates on the FPGA.

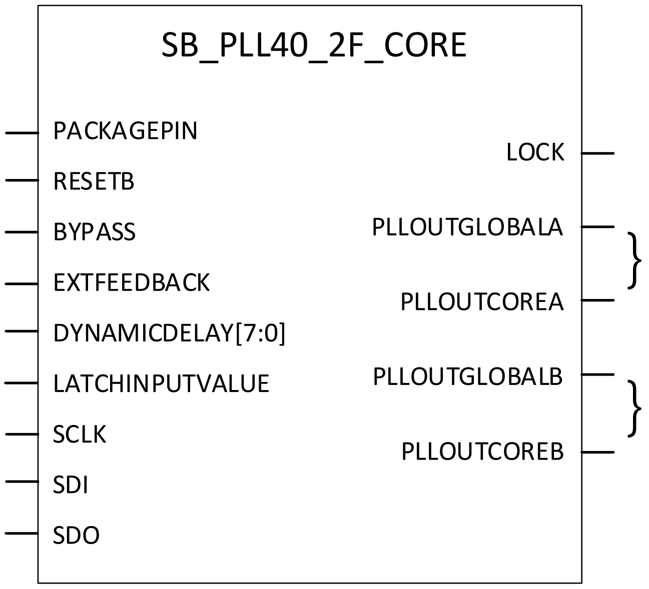

Figure 6.4. SB_PLL40_2F_CORE

Ports

REFERENCECLK: PLL source clock that serves as the input to the SB_PLL40_2F_CORE primitive.

PLLOUTGLOBALA: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCOREA: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBALA port.

PLLOUTGLOBALB: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCOREB: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBALB port.

LOCK: Output port, when HIGH, indicates that the signal on PLLOUTGLOBALB/PLLOUTCOREB is locked to the PLL source on PACKAGEPIN.

EXTFEEDBACK: External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.

DYNAMICDELAY: 4-bit input bus that enables dynamic control of the delay contributed by the Fine Delay Adjust Block. The Fine Delay Adjust Block is used when there is a need to adjust the phase alignment of PLLOUTGLOBAL/PLLOUTCORE with respect to REFERENCECLK. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.

RESETB: Active low input that asynchronously resets the PLL.

BYPASS: Input signal, when asserted, connects the signal on REFERENCECLK to

PLLOUTCORE/PLLOUTGLOBAL pins.

LATCHINPUTVALUE: Active high input, when enabled, forces the PLL into low-power mode. The PLLOUTGLOBALA/PLLOUTCOREA pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTA is set to ‘1’, and the LATCHINPUTVALUE signal is asserted. The PLLOUTGLOBALB/PLLOUTCOREB pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTB is set to ‘1’, and the LATCHINPUTVALUE signal is asserted.

SCLK, SDI, SDO: These pins are used only for internal testing purposes, and need not be instantiated by users.

Parameters

The SB_PLL40_2F_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.4. SB_PLL40_2F_CORE Parameters

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE _FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are delay compensated by (n+1)×150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_ FEEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE _RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)×150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_ RELATIVE is "FIXED".
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is "PHASE_AND_DELAY". 0-> Divide by 4 1-> Divide by 7

Parameter Name	Description	Parameter Value	Description
PLLOUT_SELECT_PORTA	Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports	SHIFTREG_0deg	0o phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY"
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY" and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs to PortA. No phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output to PORTA. No phase shift.
PLLOUT_SELECT_PORTB	Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports	SHIFTREG_0deg	0o phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY"
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY" and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs to PortB. No phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output to PORTB. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User-specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE_PORTA	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input
ENABLE_ICEGATE_PORTB	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

6.5. SB_PLL40_2F_PAD

The SB_PLL40_2F_PAD primitive should be used when the PLL is used to generate two different output frequencies, and the source clock of the PLL is driven by an input pad located in the bottom I/O bank (I/O Bank 2) or the top I/O bank (I/O Bank 0).

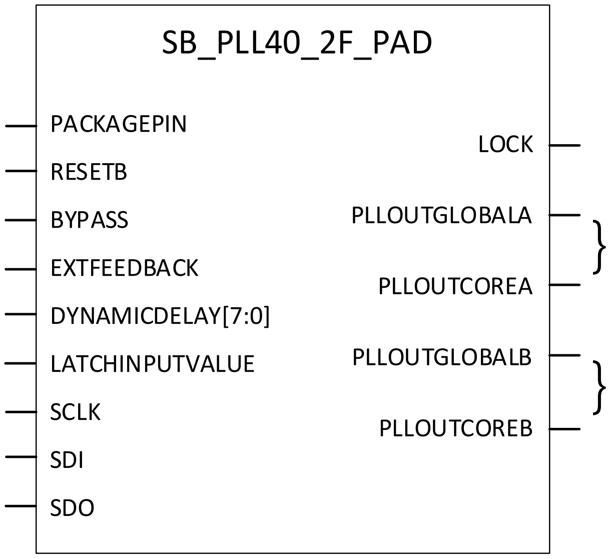

Figure 6.5. SB_PLL40_2F_PAD

Ports

PACKAGEPIN: PLL REFERENCE CLOCK pin that serves as the input to the SB_PLL40_2F_PAD primitive.

PLLOUTGLOBALA: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCOREA: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBALA port.

PLLOUTGLOBALB: Output clock generated by the PLL, drives a global clock network on the FPGA.

PLLOUTCOREB: Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBALB port.

LOCK: Output port, when HIGH, indicates that the signal on PLLOUTGLOBALB/PLLOUTCOREB is locked to the PLL source on PACKAGEPIN.

EXTFEEDBACK: External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.

DYNAMICDELAY: 4-bit input bus that enables dynamic control of the delay contributed by the Fine Delay Adjust Block. The Fine Delay Adjust Block is used when there is a need to adjust the phase alignment of PLLOUTGLOBAL/PLLOUTCORE with respect to PACKAGEPIN. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.

RESETB: Active low input that asynchronously resets the PLL.

BYPASS: Input signal, when asserted, connects the signal on PACKAGEPIN to

PLLOUTCORE/PLLOUTGLOBAL pins.

LATCHINPUTVALUE: Active high input, when enabled, forces the PLL into low-power mode. The PLLOUTGLOBALA/PLLOUTCOREA pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTA is set to ‘1’, and the LATCHINPUTVALUE signal is asserted. The PLLOUTGLOBALB/PLLOUTCOREB pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTB is set to ‘1’, and the LATCHINPUTVALUE signal is asserted.

SCLK, SDI, SDO: These pins are used only for internal testing purposes, and need not be instantiated by users.

Parameters

The SB_PLL40_2F_PAD primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.5. SB_PLL40_2F_PAD Parameters

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE _FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are delay compensated by (n+1)×150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_F EEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE _RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)×150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_ RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0-> Divide by 4 1-> Divide by 7

Parameter Name	Description	Parameter Value	Description
PLLOUT_SELECT_PORTA	Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports	SHIFTREG_0deg	0o phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY"
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY" and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs to PortA. No phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output to PORTA. No phase shift.
PLLOUT_SELECT_PORTB	Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports	SHIFTREG_0deg	0o phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY"
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is "PHASE_AND_DELAY" and SHIFTREG_DIV_MODE=0
		GENCLK	The internally generated PLL frequency outputs to PortB. No phase shift.
		GENCLK_HALF	The internally generated PLL frequency is divided by 2 and then output to PORTB. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User-specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE_PORTA	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input
ENABLE_ICEGATE_PORTB	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

7. Hard Macro Primitives

7.1. iCE40LM Hard Macros

This section describes the following dedicated hard macro primitives available in iCE40LM devices.

- SB_HSOSC (macro primitive for HSSG)
- SB_LSOSC (macro primitive for LPSG)
- SB_I2C
- SB_SPI

7.1.1. SB_HSOSC (For HSSG)

SB_HSOSC primitive can be used to instantiate High Speed Strobe Generator (HSSG), which generates 12 MHz strobe signal. The strobe can drive either the global clock network or fabric routes directly based on the clock network selection.

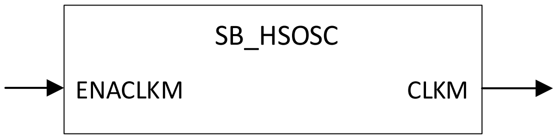

Figure 7.1. SB_HSOSC

Table 7.1. SB_HSOSC Port Signals

Signal Name	Direction	Description
ENACLKM	Input	Enable High Speed Strobe Generator. Active High.
CLKM	Output	Strobe Generator Output (12 MHz).

Clock Network Selection

By default, the strobe generator use one of the dedicated clock networks in the device to drive the elements. You may configure the strobe generator to use the fabric routes instead of global clock network using the synthesis attributes.

Synthesis Attribute

/* synthesis ROUTE_THROUGH_FABRIC=<value> */

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_HSOSC OSCInst0 (
.ENACLKM(ENACLKM),
.CLKM(CLKM)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

7.1.2. SB_LSOSC (For LPSG)

SB_LSOSC primitive can instantiate Low Power Strobe Generator (LPSG), which generates 10 kHz strobe signal. The strobe can drive either the global clock network or fabric routes directly based on the clock network selection.

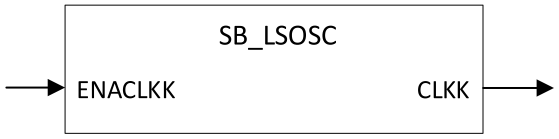

Figure 7.2. SB_LSOSC

Table 7.2. SB_LSOSC Port Signals

Signal Name	Direction	Description
ENACLKK	Input	Enable Low Power Strobe Generator. Active High.
CLKK	Output	Strobe Generator Output (10 kHz).

Clock Network Selection

By default, the strobe generator use one of the dedicated clock networks in the device to drive the elements. You may configure the strobe generator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attribute:

/* synthesis ROUTE_THROUGH_FABRIC=<value> */

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_LSOSC OSCInst0 (
       .ENACLKK(ENACLKK),
       .CLKK(CLKK)
       ) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

7.1.3. SB_I2C

The I2C hard IP provides industry standard two pin communication interface that conforms to the version 2.1 of the I2C bus specification. It could be configured as either Controller or Target port. In Controller mode, it support configurable data transfer rate and perform arbitration detection to allow it to operate in multi-Controller systems. It supports both 7 bits and 10 bits addressing in Target mode with configurable Target address and clock stretching in both Controller and Target mode with enable/disable capability.

iCE40LM device supports two I2C hard IP primitives , located at upper left corner and upper right corner of the chip.

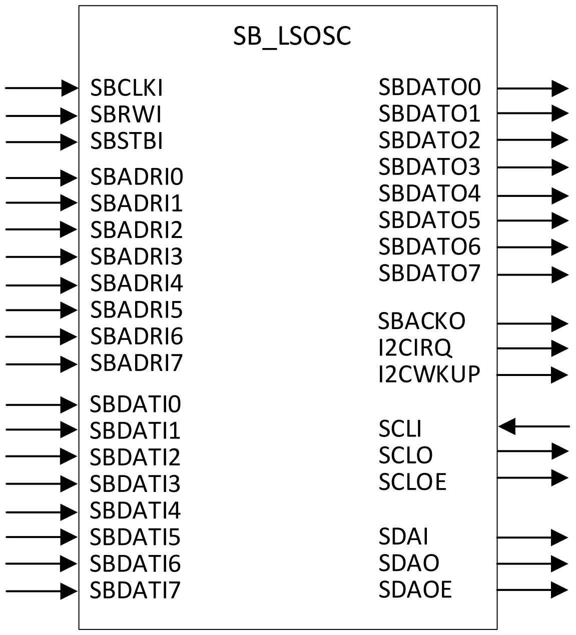

Figure 7.3. SB_I2C

Table 7.3. SB_I2C Port Signals

Signal Name	Direction	Description
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.

Signal Name	Direction	Description
SBDATO0	Output	System Data Output. Bit 0.
SBDATO1	Output	System Data Output. Bit 1.
SBDATO2	Output	System Data Output. Bit 2.
SBDATO3	Output	System Data Output. Bit 3.
SBDATO4	Output	System Data Output. Bit 4.
SBDATO5	Output	System Data Output. Bit 5.
SBDATO6	Output	System Data Output. Bit 6.
SBDATO7	Output	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement.
I2CIRQ	Output	I2C Interrupt output.
I2CWKUP	Output	I2C Wake Up from Standby signal.
SCLI	Input	Serial Clock Input.
SCLO	Output	Serial Clock Output
SCLOE	Output	Serial Clock Output Enable. Active High.
SDAI	Input	Serial Data Input
SDAO	Output	Serial Data Output
SDAOE	Output	Serial Data Output Enable. Active High.

Parameters

I2C Primitive requires configuring certain parameters for Target initial address and selecting I2C IP location.

Table 7.4. SB_I2C Parameters

I2C Location	Parameter	Parameter Default Value	Description
Upper Left Corner	I2C_SLAVE_INIT_ADDR	0b1111100001	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
	BUS_ADDR74	0b0001	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.
Upper Right Corner	I2C_SLAVE_INIT_ADDR	0b1111100010	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
	BUS_ADDR74	0b0011	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.

Synthesis Attribute

Synthesis attribute "I2C_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCLO output with respect to the SBCLKI input clock frequency.

/* synthesis I2C_CLK_DIVIDER=[Divide Range] */ Divide Range: 0, 1, 2, 3 … 1023. Default is 0.

Verilog Instantiation

```
SB_I2C i2cInst0 (
       .SBCLKI(sbclki),
       .SBRWI(sbrwi),
       .SBSTBI(sbstbi),
       .SBADRI7(sbadri[7]),
       .SBADRI6(sbadri[6]),
       .SBADRI5(sbadri[5]),
       .SBADRI4(sbadri[4]),
       .SBADRI3(sbadri[3]),
       .SBADRI2(sbadri[2]),
       .SBADRI1(sbadri[1]),
       .SBADRI0(sbadri[0]),
       .SBDATI7(sbdati[7]),
       .SBDATI6(sbdati[6]),
       .SBDATI5(sbdati[5]),
       .SBDATI4(sbdati[4]),
       .SBDATI3(sbdati[3]),
       .SBDATI2(sbdati[2]),
       .SBDATI1(sbdati[1]),
       .SBDATI0(sbdati[0]),
       .SCLI(scli),
       .SDAI(sdai),
       .SBDATO7(sbdato[7]),
       .SBDATO6(sbdato[6]),
       .SBDATO5(sbdato[5]),
       .SBDATO4(sbdato[4]),
       .SBDATO3(sbdato[3]),
       .SBDATO2(sbdato[2]),
       .SBDATO1(sbdato[1]),
       .SBDATO0(sbdato[0]),
       .SBACKO(sbacko),
       .I2CIRQ(i2cirq),
       .I2CWKUP(i2cwkup),
       .SCLO(sclo),
       .SCLOE(scloe),
       .SDAO(sdao),
       .SDAOE(sdaoe)
    )/* synthesis I2C_CLK_DIVIDER= 1 */;

defparam i2cInst0.I2C_SLAVE_INIT_ADDR = "0b1111100001";
defparam i2cInst0.BUS_ADDR74 = "0b0001";
```

7.1.4. SB_SPI

The SPI hard IP provide industry standard four-pin communication interface with 8-bit wide System Bus to communicate with System Host. It could be configured as Controller or Target SPI port with separate Chip Select Pin. In Controller mode, it provides programmable baud rate, and supports CS HOLD capability for multiple transfers. It provides variety status flags, such as Mode Fault Error flag, Transmit/Receive status flag, and others for easy communicate with system host.

The iCE40LM device supports two SPI hard IP primitives, located at lower left corner and lower right corner of the chip.

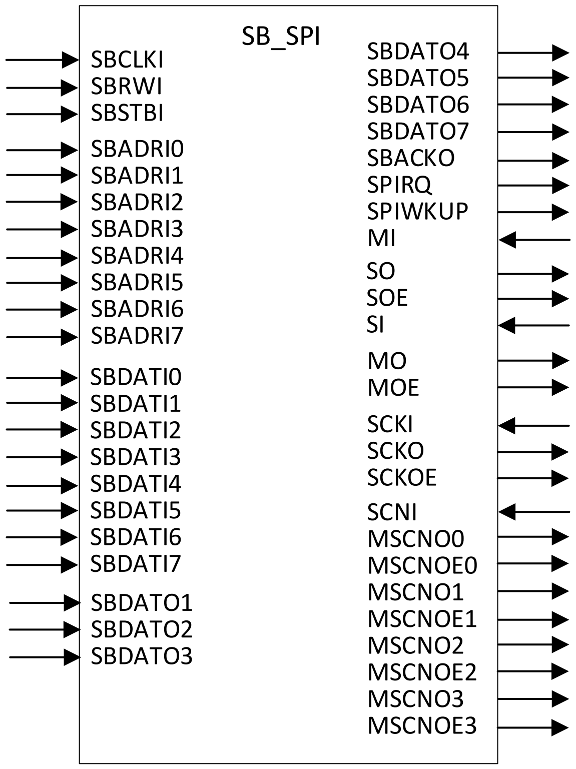

Figure 7.4. SB_SPI

Table 7.5. SB_SPI Port Signals

Signal Name	Direction	Description
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.

Signal Name	Direction	Description
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Input	System Data Output. Bit 0.
SBDATO1	Input	System Data Output. Bit 1.
SBDATO2	Input	System Data Output. Bit 2.
SBDATO3	Input	System Data Output. Bit 3.
SBDATO4	Input	System Data Output. Bit 4.
SBDATO5	Input	System Data Output. Bit 5.
SBDATO6	Input	System Data Output. Bit 6.
SBDATO7	Input	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement
SPIIRQ	Output	SPI Interrupt output.
SPIWKUP	Output	SPI Wake Up from Standby signal.
MI	Input	Controller Input from PAD
SO	Output	Target Output to PAD
SOE	Output	Target Output Enable to PAD. Active High.
SI	Input	Target Input from PAD
MO	Output	Controller Output to PAD
MOE	Output	Controller Output Enable to PAD. Active High
SCKO	Output	Target Clock Output to PAD
SCKOE	Output	Target Clock Output Enable to PAD. Active High.
SCSNI	Input	Target Chip Select Input From PAD
MCSNO0	Output	Controller Chip Select Output to PAD. Line 0.
MCSNO1	Output	Controller Chip Select Output to PAD. Line 1.
MCSNO2	Output	Controller Chip Select Output to PAD. Line 2.
MCSNO3	Output	Controller Chip Select Output to PAD. Line 3.
MCSNOE0	Output	Controller Chip Select Output Enable to PAD. Active High. Line 0.
MCSNOE1	Output	Controller Chip Select Output Enable to PAD. Active High. Line 1
MCSNOE2	Output	Controller Chip Select Output Enable to PAD. Active High. Line 2
MCSNOE3	Output	Controller Chip Select Output Enable to PAD. Active High. Line 3

Parameters

SPI Primitive requires configuring a parameter for selecting the SPI IP location.

Table 7.6. SB_SPI Parameters

SPI Location	Parameter	Parameter Default Value	Description
Lower Left Corner	BUS_ADDR74	0b0000	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.
Lower Right Corner	BUS_ADDR74	0b0010	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.

Synthesis Attribute

Synthesis attribute "SPI_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCKO output with respect to the SBCLKI input clock frequency.

/* synthesis SPI_CLK_DIVIDER= [Divide Range] */ Divide Range : 0, 1, 2, 3….63. Default is 0.

Verilog Instantiation

```
SB_SPI spiInst0 (
       .SBCLKI(sbclki),
       .SBRWI(sbrwi),
       .SBSTBI(sbstbi),
       .SBADRI7(sbadri[7]),
       .SBADRI6(sbadri[6]),
       .SBADRI5(sbadri[5]),
       .SBADRI4(sbadri[4]),
       .SBADRI3(sbadri[3]),
       .SBADRI2(sbadri[2]),
       .SBADRI1(sbadri[1]),
       .SBADRI0(sbadri[0]),
       .SBDATI7(sbdati[7]),
       .SBDATI6(sbdati[6]),
       .SBDATI5(sbdati[5]),
       .SBDATI4(sbdati[4]),
       .SBDATI3(sbdati[3]),
       .SBDATI2(sbdati[2]),
       .SBDATI1(sbdati[1]),
       .SBDATI0(sbdati[0]),
       .MI(mi),
       .SI(si),
       .SCKI(scki),
       .SCSNI(scsni),
       .SBDATO7(sbdato[7]),
       .SBDATO6(sbdato[6]),
       .SBDATO5(sbdato[5]),
       .SBDATO4(sbdato[4]),
       .SBDATO3(sbdato[3]),
       .SBDATO2(sbdato[2]),
       .SBDATO1(sbdato[1]),
       .SBDATO0(sbdato[0]),
       .SBACKO(sbacko),
       .SPIIRQ(spiirq),
       .SPIWKUP(spiwkup),
       .SO(so),
       .SOE(soe),
       .MO(mo),
       .MOE(moe),
       .SCKO(scko),
       .SCKOE(sckoe),
       .MCSNO3(mcsno_hi[3]),
       .MCSNO2(mcsno_hi[2]),
       .MCSNO1(mcsno_lo[1]),
       .MCSNO0(mcsno_lo[0]),
       .MCSNOE3(mcsnoe_hi[3]),
        .MCSNOE2(mcsnoe_hi[2]),
        .MCSNOE1(mcsnoe_lo[1]),
        .MCSNOE0(mcsnoe_lo[0])
) /* synthesis SPI_CLK_DIVIDER = "1" */;
defparam spiInst0.BUS_ADDR74 = "0b0000";
```

7.2. iCE40 Ultra (iCE5LP) Hard Macros

This section describes the following dedicated hard macro primitives available in iCE5LP (iCE40 Ultra) devices.

- SB_HFOSC
- SB_LFOSC
- SB_LED_DRV_CUR
- SB_RGB_DRV
- SB_IR_DRV
- SB_IO_OD
- SB_I2C
- SB_SPI
- SB_MAC16

7.2.1. SB_HFOSC

SB_HFOSC primitive generates 48 MHz nominal clock frequency within +/- 10% variation, with user-programmable divider value of 1, 2, 4, and 8. The HFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.

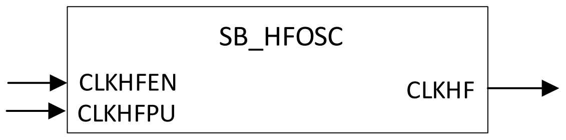

Figure 7.5. SB_HFOSC

Table 7.7. SB_HFOSC Port Signals

Signal Name	Direction	Description
CLKHFPU	Input	Power up the HFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.
CLKHFEN	Input	Enable the clock output. Enable should be low for the 100 µs power up period. Active High.
CLKHF	Output	HF Oscillator output.

Table 7.8. SB_HFOSC Parameters

Parameter	Parameter Values	Description
CLKHF_DIV	"0b00"	Sets 48 MHz HFOSC output.
	"0b01"	Sets 24 MHz HFOSC output.
	"0b10"	Sets 12 MHz HFOSC output.
	"0b11"	Sets 6 MHz HFOSC output

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CLKHFEN: Logic “0”

Input CLKHFPU: Logic “0”

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */ Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_HFOSC OSCInst0 (
.CLKHFEN(ENCLKHF),
.CLKHFPU(CLKHF_POWERUP),
.CLKHF(CLKHF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;

defparam OSCInst0.CLKHF_DIV = "0b00";
```

7.2.2. SB_LFOSC

SB_LFOSC primitive generates 10 kHz nominal clock frequency within +/- 10% variation. There is no divider on the LFOSC.

The LFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.

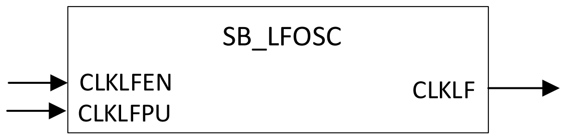

Figure 7.6. SB_LFOSC

Table 7.9. SB_LFOSC Port Signals

Signal Name	Direction	Description
CLKLFPU	Input	Power up the LFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.
CLKLFEN	Input	Enable the clock output. Enable should be low for the 100 µs power up period. Active High.
CLKLF	Output	LF Oscillator output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CLKLFEN: Logic “0”

Input CLKLFPU: Logic “0”

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_LFOSC OSCInst1 (
       .CLKLFEN(ENCLKLF),
       .CLKLFPU(CLKLF_POWERUP),
       .CLKLF(CLKLF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

7.2.3. SB_LED_DRV_CUR

SB_LED_DRV_CUR primitive generates the constant reference current required to power up the SB_RGB_DRV and SB_IR_DRV primitives.

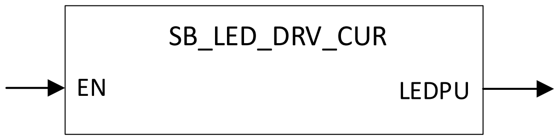

Figure 7.7. SB_LED_DRV_CUR

Table 7.10. SB_LED_DRV_CUR Port Signals

Signal Name	Direction	Description
EN	Input	Enable to generate constant current source for SB_RGB_DRV and SB_IR_DRV primitives. After Enable, the reference current value is stable after 100 µs. Active High.
LEDPU	Output	Power up signal for SB_RGB_DRV and SB_IR_DRV primitives. This port should be connected only to RGBPU/IRPU pins of SB_RGB_DRV and SB_IR_DRV primitives.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input EN : Logic "0"

Verilog Instantiation

```
SB_LED_DRV_CUR      LED_CUR_inst
(
       .EN(enable_led_current),
       .LEDPU(led_power_up)
);
```

7.2.4. SB_RGB_DRV

SB_RGB_DRV primitive contains three dedicated open drain I/O pins for RGB LED outputs. Each of the RGB LED output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_RGB_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

The primitive allows configuration of each of the 3 RGB LED outputs individually. When the RGBx_CURRENT parameter of RGBx output is set to "0b000000", then SB_IO_OD can be used to drive the package pin.

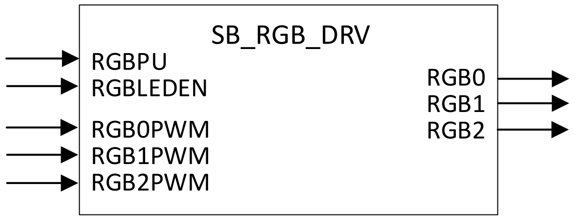

Figure 7.8. SB_RGV_DRV

Table 7.11. SB_RGB_DRV Port Signals

Signal Name	Direction	Description
RGBLEDEN	Input	Enable the SB_RGB_DRV primitive. Active High.
RGBPU	Input	Power Up Signal. Connect to LEDPU port of SB_LED_DRV_CUR primitive.
RGB0PWM	Input	Input data to drive RGB0 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB1PWM	Input	Input data to drive RGB1 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB2PWM	Input	Input data to drive RGB2 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB0	Output	RGB0 LED output
RGB1	Output	RGB1 LED output
RGB2	Output	RGB2 LED output

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input RGBLEDEN : Logic "0"

Input RGB0PWM : Logic "0"

Input RGB1PWM : Logic "0"

Input RGB2PWM : Logic "0"

Input RGBPU : Logic "0"

Parameters

The SB_RGB_DRV primitive contains the following parameter and their default values:

Parameter RGB0_CURRENT = "0b000000";

Parameter RGB1_CURRENT = "0b000000";

Parameter RGB2_CURRENT = "0b000000";

Parameter values:

"0b000000" = 0 mA. // Set this value to use the associated SB_IO_OD instance at RGB

// LED location.

"0b000001" = 4 mA.

"0b000011" = 8 mA.

"0b000111" = 12 mA.

"0b001111" = 16 mA.

"0b011111" = 20 mA.

"0b111111" = 24 mA.

Verilog Instantiation

```
SB_RGB_DRV RGB_DRIVER (
       .RGBLEDEN(ENABLE_LED),
       .RGB0PWM(RGB0),
       .RGB1PWM(RGB1),
       .RGB2PWM(RGB2),
       .RGBPU(led_power_up),
       .RGB0(LED0),
       .RGB1(LED1),
       .RGB2(LED2)
);

defparam RGB_DRIVER.RGB0_CURRENT = "0b111111"; defparam RGB_DRIVER.RGB1_CURRENT =
"0b111111" ; defparam RGB_DRIVER.RGB2_CURRENT = "0b111111";
```

7.2.5. SB_IR_DRV

SB_IR_DRV primitive contains a single dedicated open drain I/O pin for IRLED output. The IRLED output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_IR_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

When the IR_CURRENT parameter is set to "0b0000000000", then SB_IO_OD can be used to drive the package pin.

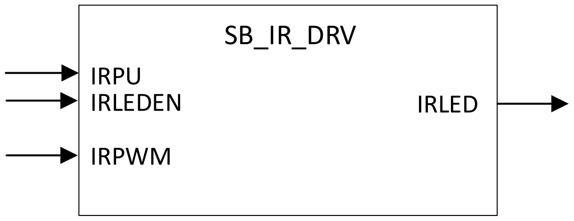

Figure 7.9. SB_IR_DRV

Table 7.12. SB_IR_DRV Port Signals

Signal Name	Direction	Description
IRLEDEN	Input	Enable the SB_IR_DRV primitive. Active High.
IRPU	Input	Power Up Signal. Connect to LEDPU port of SB_LED_DRV_CUR primitive.
IRPWM	Input	PWM Input data to drive IRLED pin.
IRLED	Output	IR LED output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input IRLEDEN	: Logic "0"
Input IRPWM	: Logic "0"

Parameter

The SB_IR_DRV primitive contains the following parameter and their default values:

Parameter IR_CURRENT = "0b0000000000";

Parameter Values:

"0b0000000000"; = 0 mA. // Set this value to use the associated SB_IO_OD instance at // IR LED location.

"0b0000000001"; = 50 mA

"0b0000000011"; = 100 mA

"0b0000000111"; = 150 mA

"0b0000001111"; = 200 mA

"0b0000011111"; = 250 mA

"0b0000111111"; = 300 mA

"0b0001111111"; = 350 mA

"0b0011111111"; = 400 mA

"0b0111111111"; = 450 mA

"0b1111111111"; = 500 mA

Verilog Instantiation

```
SB_IR_DRV    IRDRVinst (
       .IRLEDEN(ENABLE_IRLED),
       .IRPWM(IR_INPUT),
       .IRPU(led_power_up),
       .IRLED(IR_LED)
);

defparam IRDRVinst.IR_CURRENT = "0b1111111111";
```

7.2.6. SB_RGB_IP

SB_RGB_IP primitive generates the three RGB PWM outputs, to be connected to the LED drivers.

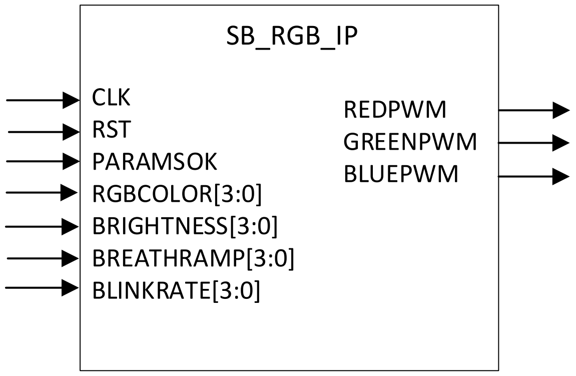

Figure 7.10. SB_RGB_IP

Table 7.13. SB_RGB_IP Port Signals

Signal Name	Direction	Description
CLK	Input	Clock Input
RST	Input	Asynchronous Reset Input
PARAMSOK	Input	Enable signal to sample the RGB data
RGBCOLOR[3:0]	Input	4-bit RGB color data input
BRIGHTNESS[3:0]	Input	4-bit RGB brightness data input
BREATHRAMP[3:0]	Input	4-bit breathramp data input
BLINKRATE[3:0]	Input	4-bit blinkrate data input
REDPWM	Output	RED PWM output
GREENPWM	Output	GREEN PWM output
BLUEPWM	Output	BLUE PWM output

Default Signal Values

The iCEcube2 software assigns the logic "0" value to all unconnected input ports.

Verilog Instantiation

```
SB_RGB_IP RGBPWMIP_inst(
       .CLK (CLK),
       .RST (RST),  // Async rst
       .PARAMSOK (PARAMSOK),
       .RGBCOLOR (RGBCOLOR),
       .BRIGHTNESS (BRIGHTNESS),
       .BREATHRAMP (BREATHRAMP),
       .BLINKRATE (BLINKRATE),
       .REDPWM (REDPWM),
       .GREENPWM (GREENPWM),
       .BLUEPWM (BLUEPWM)
);
```

7.2.7. SB_IO_OD

The SB_IO_OD is the open drain I/O primitive. When the Tristate output is enabled, the I/O pulls down the package pin signal to zero. The following figure and Verilog template illustrate the complete user accessible logic diagram, and its Verilog instantiation.

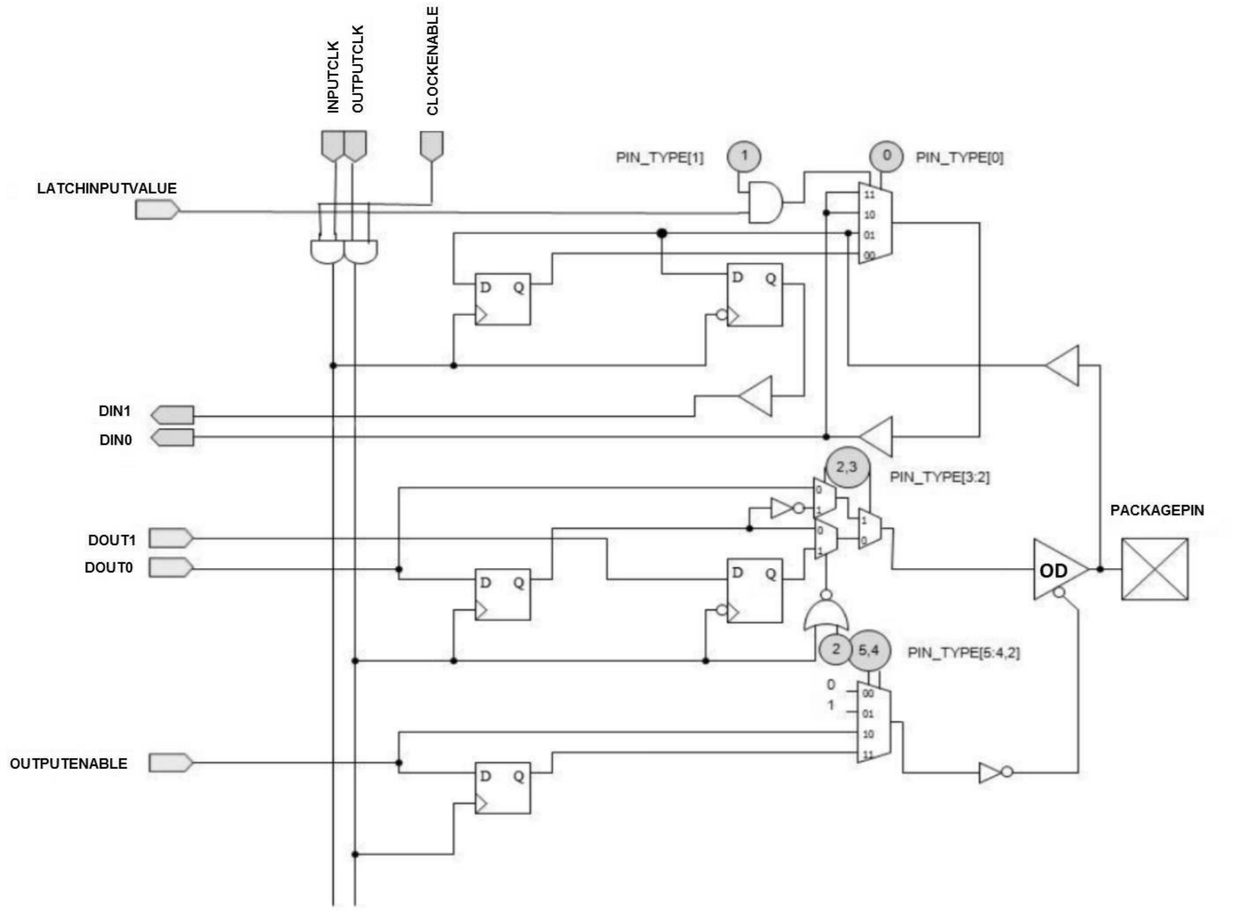

Figure 7.11. SB_IO_OD

VHDL Code

```
COMPONENT SB_IO_OD IS

       GENERIC(
       PIN_TYPE : std_logic_vector(5 downto 0) := "000000";
       NEG_TRIGGER: std_logic_vector := '0'
       );
       PORT(
       PACKAGE_PIN : in std_logic;
       LATCH_INPUT_VALUE : in std_logic;
       CLOCK_ENABLE : in std_logic;
       INPUT_CLK : in std_logic;
       OUTPUT_CLK : in std_logic;
       OUTPUT_ENABLE : in std_logic;
       D_OUT_0 : in std_logic;
       D_OUT_1 : in std_logic;
       D_IN_0 : out std_logic;
       D_IN_1 : out std_logic
       );
       END COMPONENT SB_IO;
       attribute DRIVE_STRENGTH : string;
       attribute DRIVE_STRENGTH of SB_IO : component is "x2";

--COMPONENT INSTANTIATION
       uut : SB_IO_OD
       generic map (
              PIN_TYPE => "000000",
              NEG_TRIGGER => '0'
       )
       port map(
       -- enter port connections here.
              PACKAGEPIN                => PackagePin
              CLOCKENABLE             => clockenable,
              LATCHINPUT_VALUE    => latchinputvalue,
              INPUTCLK                  => inputclk,
              OUTPUTCLK                 => outputclk,
              OUTPUTENABLE        => outputenable,
              DOUT1                  => dout0,
              DOUT0                  => dout1,
              DIN1                        => din1,
              DIN0                        => din0
       );
```

Table 7.14. SB_IO_OD Port Signals

Signal Name	Direction	Description
PACKAGEPIN	Bidirectional	Bidirectional I/O pin
LATCHINPUTVALUE	Input	Latches/Holds the input data
CLOCKENABLE	Input	Clock enable signal
INPUTCLK	Input	Clock for the input registers
OUTPUTCLK	Input	Clock for the output registers
OUTPUTENABLE	Input	Enable Tri-state output. Active high.

Signal Name	Direction	Description
DOUT0	Input	Data to package pin
DOUT1	Input	Data to package pin
DIN0	Output	Data from package pin
DIN1	Output	Data from package pin

Default Signal Values

The iCEcube2 software assigns the logic “0” value to all unconnected input ports except for

CLOCKENABLE.

Note that explicitly connecting logic “1” value to port CLOCKENABLE results in a non-optimal implementation, since extra LUT is used to generate the Logic “1”. If your intention is to keep the Input and Output registers always enabled, it is recommended that port CLOCKENABLE to be left unconnected.

Parameter Values

Parameter PIN_TYPE = 6’b000000;

    // See Input and Output Pin Function Tables in SB_IO.

    // Default value of PIN_TYPE = 6’b000000

Parameter NEG_TRIGGER = 1’b0;

    // Specify the polarity of all FFs in the I/O to be falling edge when
    NEG_TRIGGER = 1.   Default is 1’b0, rising edge.

Input and Output Pin Function Tables

Refer SB_IO Input and Output Pin Functional table for the PIN_TYPE settings. Some of the output pin configurations are not applicable for SB_IO_OD primitive.

Verilog Instantiation

```
SB_IO_OD     OpenDrainInst0
(
.PACKAGEPIN (PackagePin),        // User’s Pin signal name
.LATCHINPUTVALUE (latchinputvalue),    // Latches/holds the Input value
.CLOCKENABLE (clockenable),            // Clock Enable common to input and
                                       // output clock
.INPUTCLK (inputclk),                  // Clock for the input registers
.OUTPUTCLK (outputclk),                // Clock for the output registers
.OUTPUTENABLE (outputenable),          // Output Pin Tristate/Enable
                                       // control
.DOUT0 (dout0),                        // Data 0 – out to Pin/Rising clk
                                       // edge
.DOUT1 (dout1),                        // Data 1 - out to Pin/Falling clk
                                       // edge
.DIN0 (din0),                    // Data 0 - Pin input/Rising clk
                                       // edge
.DIN1 (din1)                     // Data 1 – Pin input/Falling clk
                                       // edge
);

defparam OpenDrainInst0.PIN_TYPE = 6'b000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
```

7.2.8. SB_I2C

The iCE5LP device supports two I2C hard IP primitives , located at upper left corner and upper right corner of the chip.

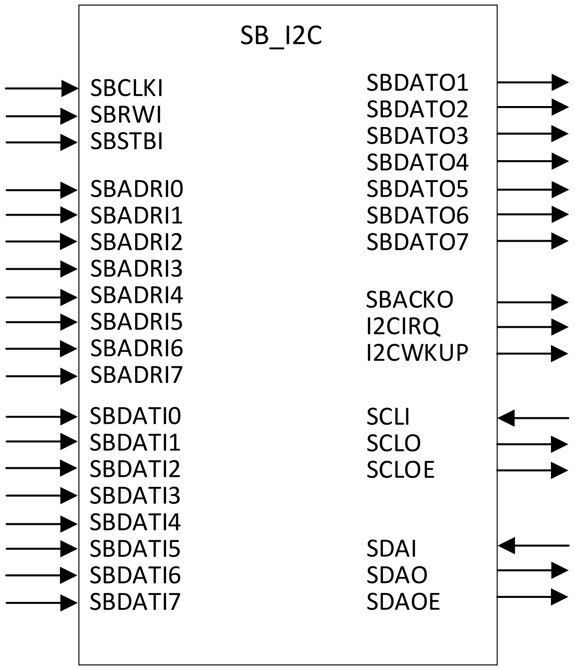

Figure 7.12. SB_I2C

Table 7.15. SB_I2C Port Signals

Signal Name	Direction	Description
SBCLKI	Input	System Clock input
SBRWI	Input	System Read/Write Input
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Output	System Data Output. Bit 0.
SBDATO1	Output	System Data Output. Bit 1.
SBDATO2	Output	System Data Output. Bit 2.
SBDATO3	Output	System Data Output. Bit 3.

Signal Name	Direction	Description
SBDATO4	Output	System Data Output. Bit 4.
SBDATO5	Output	System Data Output. Bit 5.
SBDATO6	Output	System Data Output. Bit 6.
SBDATO7	Output	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement.
I2CIRQ	Output	I2C Interrupt output.
I2CWKUP	Output	I2C Wake Up from Standby signal.
SCLI	Input	Serial Clock Input.
SCLO	Output	Serial Clock Output
SCLOE	Output	Serial Clock Output Enable. Active High.
SDAI	Input	Serial Data Input
SDAO	Output	Serial Data Output
SDAOE	Output	Serial Data Output Enable. Active High.

Parameters

I2C Primitive requires configuring certain parameters for Target initial address and selecting I2C IP location.

Table 7.16. SB_I2C Parameters

I2C Location	Parameters	Parameter Default Value	Description
Upper Left Corner	I2C_SLAVE_INIT_ADDR	0b1111100001	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
	BUS_ADDR74	0b0001	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.
Upper Right Corner	I2C_SLAVE_INIT_ADDR	0b1111100010	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
	BUS_ADDR74	0b0011	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.

Synthesis Attribute

I2C_CLK_DIVIDER

Synthesis attribute "I2C_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCLO output with respect to the SBCLKI input clock frequency.

```
/* synthesis I2C_CLK_DIVIDER= Divide Value *
/ Divide Value: 0, 1, 2, 3 … 1023. Default is 0.
```

SDA_INPUT_DELAYED

SDA_INPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAI signal.

```
/* synthesis SDA_INPUT_DELAYED= value */
Value:
0:  No delay.
1:  Add 50 ns delay. (Default value).
```

SDA_OUTPUT_DELAYED

SDA_OUTPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAO signal.

/* synthesis SDA_OUTPUT_DELAYED= value */

Value:

0: No delay (Default value).

1: Add 50 ns delay.

7.2.9. SB_SPI

The iCE5LP device supports two SPI hard IP primitives, located at lower left corner and lower right corner of the chip.

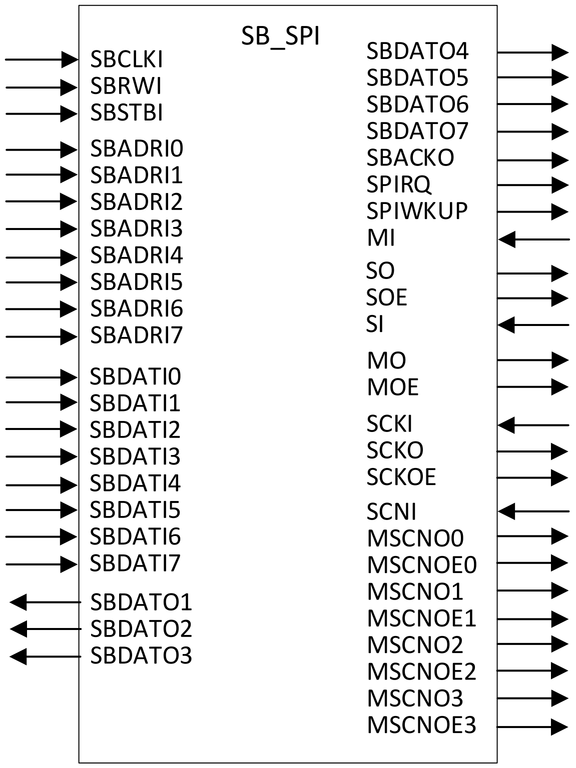

Figure 7.13. SB_SPI

Table 7.17. SB_SPI Port Signals

Signal Name	Direction	Description
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.

Signal Name	Direction	Description
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Input	System Data Output. Bit 0.
SBDATO1	Input	System Data Output. Bit 1.
SBDATO2	Input	System Data Output. Bit 2.
SBDATO3	Input	System Data Output. Bit 3.
SBDATO4	Input	System Data Output. Bit 4.
SBDATO5	Input	System Data Output. Bit 5.
SBDATO6	Input	System Data Output. Bit 6.
SBDATO7	Input	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement
SPIIRQ	Output	SPI Interrupt output.
SPIWKUP	Output	SPI Wake Up from Standby signal.
MI	Input	Controller Input from PAD
SO	Output	Target Output to PAD
SOE	Output	Target Output Enable to PAD. Active High.
SI	Input	Target Input from PAD
MO	Output	Controller Output to PAD
MOE	Output	Controller Output Enable to PAD. Active High
SCKO	Output	Target Clock Output to PAD
SCKOE	Output	Target Clock Output Enable to PAD. Active High.
SCSNI	Input	Target Chip Select Input From PAD
MCSNO0	Output	Controller Chip Select Output to PAD. Line 0.
MCSNO1	Output	Controller Chip Select Output to PAD. Line 1.
MCSNO2	Output	Controller Chip Select Output to PAD. Line 2.
MCSNO3	Output	Controller Chip Select Output to PAD. Line 3.
MCSNOE0	Output	Controller Chip Select Output Enable to PAD. Active High. Line 0.
MCSNOE1	Output	Controller Chip Select Output Enable to PAD. Active High. Line 1
MCSNOE2	Output	Controller Chip Select Output Enable to PAD. Active High. Line 2
MCSNOE3	Output	Controller Chip Select Output Enable to PAD. Active High. Line 3

Parameters

SPI Primitive requires configuring a parameter for selecting the SPI IP location.

Table 7.18. SB_SPI Parameters

I2C Location	Parameters	Parameter Default Value	Description
Lower Left Corner	BUS_ADDR74	0b0000	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.
Lower Right Corner	BUS_ADDR74	0b0001	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.

Synthesis Attribute

Synthesis attribute "SPI_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCKO output with respect to the SBCLKI input clock frequency.

/* synthesis SPI_CLK_DIVIDER= [Divide Range] */ Divide Range : 0, 1, 2, 3….63. Default is 0.

7.2.10. SB_MAC16

The SB_MAC16 primitive is the dedicated configurable DSP block available in iCE5LP devices. The SB_MAC16 can be configured into a multiplier, adder, subtracter, accumulator, multiply-add and multiply- sub through the instance parameters. The SB_MAC16 blocks can be cascaded to implement wider functional units.

iCEcube2 supports a set of predefined SB_MAC16 functional configurations. Refer to the SB_MAC16 Configurations section for the list of supported configurations.

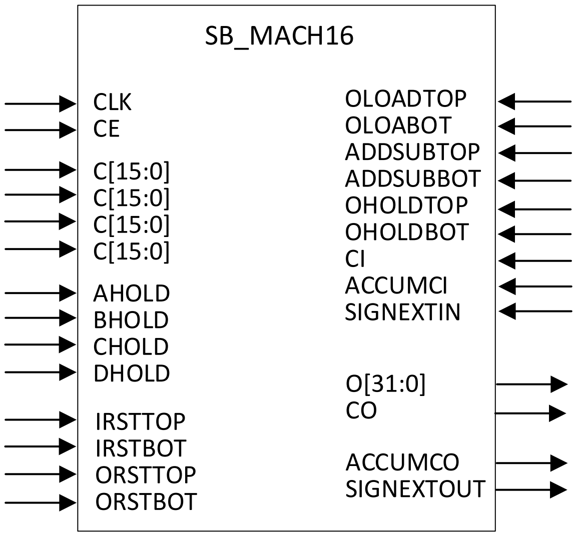

Figure 7.14. SB_MAC16

Port A (Source Clock)

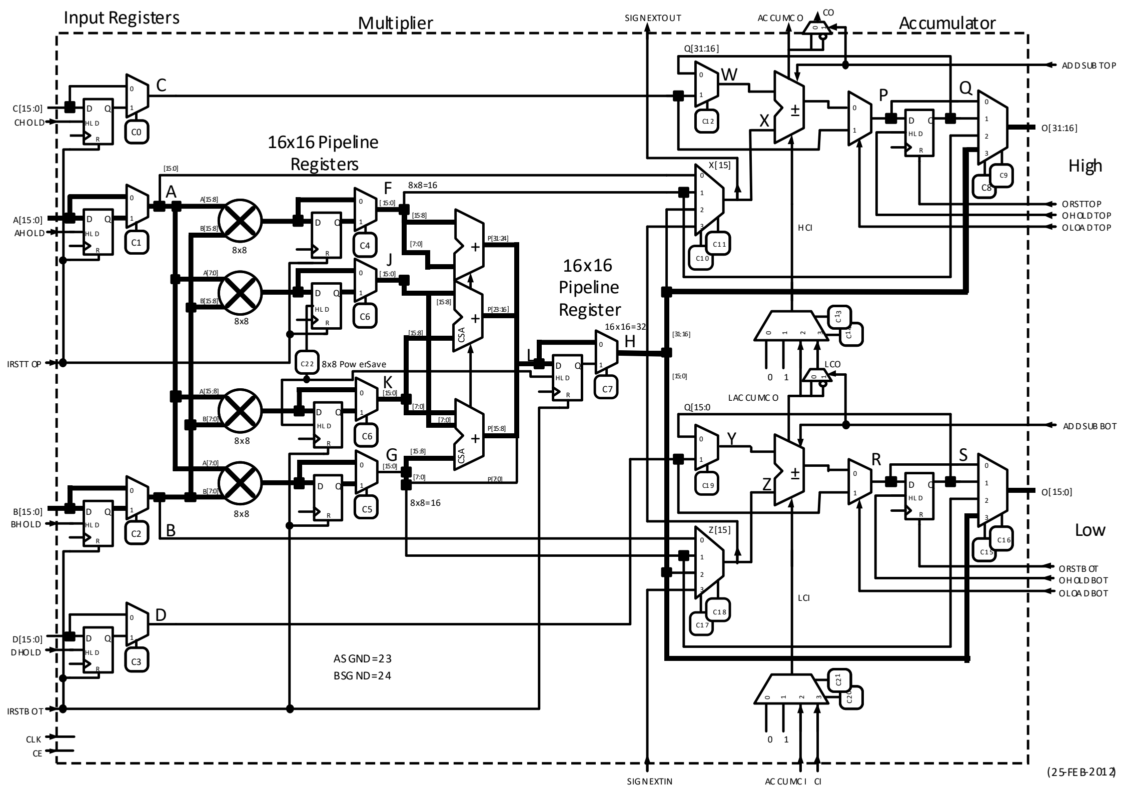

Figure 7.15. SB_MAC16

Port A (Generated Clock)

Table 7.19. SB_MAC16 Port Signals

Signal Name	Direction	Description	Default Values
CLK	Input	Clock input. Applies to all clocked elements in the MAC16 Block.	
CE	Input	Clock Enable Input. Active High.	1'b1
C[15:0]	Input	Input to the C Register / Direct input to the adder accumulator.	16'b0
A[15:0]	Input	Input to the A Register / Direct input to the multiplier blocks /Direct input to the adder accumulator.	16'b0
B[15:0]	Input	Input to the B Register / Direct input to the multiplier blocks /Direct input to the adder accumulator.	16'b0
D[15:0]	Input	Input to the D Register / Direct input to the adder accumulator.	16'b0
AHOLD	Input	Hold A registers Data .Controls data flow into the input register A. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.	1'b0
BHOLD	Input	Hold B registers Data .Controls data flow into the B input register. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.	1'b0

Signal Name	Direction	Description	Default Values
CHOLD	Input	Hold C registers Data. Controls data flow into the A input register. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.	1'b0
DHOLD	Input	Hold D registers Data. Controls data flow into the A input register. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.	1'b0
IRSTTOP	Input	Reset input to the A and C input registers, and the pipeline registers in the upper half of the multiplier block. Active High	1'b0
IRSTBOT	Input	Reset input to the B and C input registers, and the pipeline registers in the lower half of the multiplier block, and the 32-bit multiplier result pipeline register. Active High.	1'b0
ORSTTOP	Input	Reset the high-order bits of the accumulator register ([31:16]). Active High.	1'b0
ORSTBOT	Input	Reset the low-order accumulator register bits ([15:0]). Active High.	1'b0
OLOADTOP	Input	High-order Accumulator Register Accumulate/Load. Controls whether the accumulator register accepts the output of the adder/subtracter or whether the register is loaded with the value from Input C (or Register C, if configured). 0: Accumulator Register [31:16] loaded with output from adder/subtracter. 1: Accumulator Register [31:16] loaded with Input C or Register C, depending on primitive parameter value.	1'b0
OLOADBOT	Input	Low-order Accumulator Register Accumulate/Load. Controls whether the low-order accumulator register bits (15:0] accepts the output of the adder/subtracter or whether the register is loaded with the value from Input D (or Register D, if configured). 0: Accumulator Register [15:0] loaded with output from adder/subtracter. 1: Accumulator Register [15:0] loaded with Input D or Register D, depending on primitive parameter value.	1'b0
ADDSUBTOP	Input	High-order Add/Subtract. Controls whether the adder/subtracter adds or subtracts. 0: Add: W+X+HCI 1: Subtract: W-X-HCI	1'b0: Add
ADDSUBBOT	input	Low-order Add/Subtract. Controls whether the adder/subtracter adds or subtracts. 0: Add: Y+Z+LCI 1: Subtract: Y-Z-LCI	1'b0: Add
OHOLDTOP	Input	High-order Accumulator Register Hold. Controls data flow into the high-order ([31:16]) bits of the accumulator. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.	
OHOLDBOT	Input	Low-order Accumulator Register Hold. Controls data flow into the high-order ([15:0]) bits of the accumulator. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.	
CI	Input	Carry/borrow input from lower logic tile.	
ACCUMCI	Input	Cascade Carry/borrow input from lower MAC16 block.	
SIGNEXTIN	Input	Sign extension input from lower MAC16 block.	

Signal Name	Direction	Description	Default Values
O[31:0]	Output	32-bit MAC16 Output. O[31:0]: 32-bit result of a 16x16 multiply operation or a 32-bit adder/accumulate function. O[31:16]: 16-bit result of an 8x8 multiply operation or a 32-bit adder/accumulate function. O[15:0]: 16-bit result of an 8x8 multiply operation or a 32-bit adder/accumulate function.	
CO	Output	Carry/borrow output to higher logic tile.	
ACCUMCO	Output	Cascade Carry/borrow output to higher MAC16 block.	
SIGNEXTOUT	Output	Sign extension output to higher MAC16 block.	

Parameters

The parameter table below shows the list of parameters to configure the SB_MAC16 block. This table also maps the parameter to the configuration bits shown in the SB_MAC16 Functional diagram.

Table 7.20. SB_MAC16 Parameters

Parameter Name	Configuration Bits	Parameter Description	Value	Description
NEG_TRIGGER		Controls input clock polarity	0	All the registers are rising_edge triggered.
			1	All the registers are falling_edge triggered.
C_REG	C0	Input C register Control.	0	Input C not registered
			1	Input C registered
A_REG	C1	Input A register Control	0	Input A not registered
			1	Input A registered
B_REG	C2	Input B register Control	0	Input B not registered
			1	Input B registered
D_REG	C3	Input D register Control	0	Input D not registered
			1	Input D registered
TOP_8x8_MULT_REG	C4	Top 8x8 multiplier output register control (point F)	0	Top 8x8 multiplier output is not registered.
			1	Top 8x8 multiplier output is registered.
BOT_8x8_MULT_REG	C5	Bottom 8x8 multiplier output register control. (point G)	0	Bottom 8x8 multiplier output is not registered.
			1	Bottom 8x8 multiplier output is registered
PIPELINE_16x16_MULT_REG1	C6	Intermediate 8x8 multiplier pipeline register controls (points J and K). These multipliers are only used for 16x16 multiply operations. For 8x8 multiply operations, set C6 and C7 to 1, which reduces power consumption.	0	Intermediate 8x8 multiplier outputs are not registered.
			1	Intermediate 8x8 multiplier outputs are registered.

Parameter Name	Configuration Bits	Parameter Description	Value	Description
PIPELINE_16x16_MULT_REG2	C7	16x16 multiplier Pipeline registers control. (point H)	0	32-bit output from 16x16 multiplier is not registered.
			1	32-bit output from 16x16 multiplier is registered.
TOPOUTPUT_SELECT	C9,C8	Selects Top SB_MAC16 output O[31:16].	00	16-bit output of Multiplexer P, from top adder/subtracter,
			01	16-bit output from upper accumulator register, Q
			10	16-bit output from upper 8x8 multiplier, F
			11	Upper 16-bit output from 16x16 multiplier, H
TOPADDSUB_LOWERINPUT	C11,C10	Selects input X for the upper adder/subtracter	00	16-bit input from A input or associated input register.
			01	16-bit output from upper 8x8 multiplier, F
			10	Upper 16-bit output from 16x16 multiplier, H
			11	Sign extension input from lower adder/subtracter, Z[15]. Duplicate 16 bits.
TOPADDSUB_UPPERINPUT	C12	Selects input W for the upper adder/subtracter	0	16-bit feedback from upper accumulator register, Q
			1	16-bit input from C input or associated pipeline register
TOPADDSUB_CARRYSELECT	C14,C13	Carry/borrow input select to upper adder/subtracter.	00	Constant 0
			01	Constant 1
			10	Cascade Carry/borrow output from lower adder/subtracter
			11	Carry/borrow output from lower adder/subtracter
BOTOUTPUT_SELECT	C16,C15	Selects Lower SB_MAC16 output O[15:0].	00	16-bit output of multiplexer R from lower adder/subtracter
			01	16-bit output from lower accumulator register, S
			10	16-bit output from lower 8x8 multiplier, G
			11	Lower 16-bit output from 16x16 multiplier, H
BOTADDSUB_LOWERINPUT	C18,C17	Selects Input Z for the lower dder/subtracter	00	16-bit input from B input or associated pipeline register
			01	16-bit output from lower 8x8 multiplier, G
			10	Lower 16-bit output from 16x16 multiplier, H
			11	Sign extension input SIGNEXTIN. Duplicate 16 bits.
BOTADDSUB_UPPERINPUT	C19	Selects Input Y for the lower adder/subtracter	0	16-bit feedback from lower accumulator register, S
			1	16-bit input from D input or associated input register.

Parameter Name	Configuration Bits	Parameter Description	Value	Description
BOTADDSUB_CARRYSELECT	C21,C20	Carry/borrow input select to lower adder/subtracter	00	Constant 0
			01	Constant 1
			10	Cascade Carry/borrow input from ACCUMCI
			11	Carry/borrow input from CI
MODE_8x8	C22	Selects 8x8 Multiplier mode and 8x8 Low-Power Multiplier Blocking Option	0	No effect
			1	Selects 8x8 Multiplier mode. Holds the pipelining registers associate with a 16x16 multiplier in clock disable mode. Helps reduce the dynamic power consumption within the multiplier function. Used in conjunction with C6, C7 settings.
A_SIGNED	C23	Indicates whether multiplier input A is signed or unsigned. Applies regardless if input A is 16-or 32- bits wide.	0	Multiplier input A is unsigned.
			1	Multiplier input A is signed.
B_SIGNED	C24	Indicates whether multiplier input B is signed or unsigned. Applies regardless if input B is 16-or 32- bits wide.	0	Multiplier input B is unsigned.
			1	Multiplier input B is signed.

7.2.10.1. SB_MAC16 Configurations

The following SB_MAC16 functional blocks are supported in iCEcube2.

- Multiplier
- Multiply and Accumulate (MAC)
- Accumulator (ACC)
- Add/Subtract (ADD/SUB)
- Multiply and Add/Subtract (MULTADDSUB)

The valid configuration parameter settings for each functional block are listed below.

Note: Read the configuration settings from left to right while filling the SB_MAC16 instance parameter values.

Multiplier Configurations

8×8 Multiplier	: 8×8 input multiplier with 16-bit output.
16×16 Multiplier	: 16×16 input multiplier with 32-bit output.

Table 7.21. Multiplier Configurations

SB_MAC16 Function/ Parameter Settings	**B_SIGNED**	**A_SIGNED**	**MODE_8x8**	**BOTADDSUB_CARRYSELECT**		**BOTADDSUB_UPPERINPUT**	**BOTADDSUB_LOWERINPUT**		**BOTOUTPUT_SELECT**		**TOPADDSUB_CARRYSELECT**		**TOPADDSUB_UPPERINPUT**	**TOPADDSUB_LOWERINPUT**		**TOPOUTPUT_SELECT**		**PIPELINE_16x16_MULT_REG2**	**PIPELINE_16x16_MULT_REG1**	**BOT_8x8_MULT_REG**	**TOP_8x8_MULT_REG**	**D_REG**	**B_REG**	**A_REG**	**C_REG**
	CBIT[24]	**CBIT[23]**	**CBIT[22]**	**CBIT[21]**	**CBIT[20]**	**CBIT[19]**	**CBIT[18]**	**CBIT[17]**	**CBIT[16]**	**CBIT[15]**	**CBIT[14]**	**CBIT[13]**	**CBIT[12]**	**CBIT[11]**	**CBIT[10]**	**CBIT[9]**	**CBIT[8]**	**CBIT[7]**	**CBIT[6]**	**CBIT[5]**	**CBIT[4]**	**CBIT[3]**	**CBIT[2]**	**CBIT[1]**	**CBIT[0]**
8×8 Multiplier																									
mult_8x8_all_pipelined_unsigned	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	1	0	1	1	0
mult_8x8_all_pipelined_signed	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	1	0	1	1	0
mult_8x8_bypass_unsigned	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
mult_8x8_bypass_signed	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
16×16 Multiplier																									
mult_16x16_all_pipelined_unsigned	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0
mult_16x16_all_pipelined_signed	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0
mult_16x16_intermediate_register_bypassed_unsigned	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0
mult_16x16_intermediate_register_bypassed_signed	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
mult_16x16_bypass_unsigned	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
mult_16x16_bypass_signed	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0

MAC Configurations

16-bit MAC: 8×8 input, 16-bit output MAC.

32-bit MAC: 16×16 input, 32-bit output MAC.

Table 7.22. MAC Configurations

SB_MAC16 Function/Parameter Settings	B_SIGNED	A_SIGNED	MODE_8x8	BOTADDSUB_CARRYSELECT		BOTADDSUB_UPPERINPUT	BOTADDSUB_LOWERINPUT		BOTOUTPUT_SELECT		TOPADDSUB_CARRYSELECT		TOPADDSUB_UPPERINPUT	TOPADDSUB_LOWERINPUT		TOPOUTPUT_SELECT		PIPELINE_16x16_MULT_REG2	PIPELINE_16x16_MULT_REG1	BOT_8x8_MULT_REG	TOP_8x8_MULT_REG	D_REG	B_REG	A_REG	C_REG
	CBIT[24]	CBIT[23]	CBIT[22]	CBIT[21]	CBIT[20]	CBIT[19]	CBIT[18]	CBIT[17]	CBIT[16]	CBIT[15]	CBIT[14]	CBIT[13]	CBIT[12]	CBIT[11]	CBIT[10]	CBIT[9]	CBIT[8]	CBIT[7]	CBIT[6]	CBIT[5]	CBIT[4]	CBIT[3]	CBIT[2]	CBIT[1]	CBIT[0]
16-bit MAC																									
mac_16_all_pipelined_unsigned	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1
mac_16_intermediate_register_ bypassed_unsigned	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1
mac_16_bypassed_unsigned	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
32-bit MAC																									
mac_32_all_pipelined_ unsigned	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1
mac_32_all_pipelined_cascaded_ unsigned	0	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1
mac_32_all_pipelined_cin_ unsigned	0	0	0	1	1	0	1	0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1
mac_32_intermediate_register_ bypassed_unsigned	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0	1	1	1	1
mac_32_intermediate_register_bypasse d_cascaded_unsigned	0	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0	1	1	1	1
mac_32_intermediate_register_ bypassed_cin_unsigned	0	0	0	1	1	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0	1	1	1	1
mac_32_bypassed_unsigned	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
mac_32_bypassed_cascaded_ unsigned	0	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
mac_32_bypassed_cin_ unsigned	0	0	0	1	1	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
mac_32_intermediate_register_ bypassed_signed	1	1	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0	1	1	1	1

ACCUMULATOR Configurations

16-bit ACCUMULATOR: 16-bit input, 16-bit output Accumulator.

32-bit ACCUMULATOR: 32-bit input, 32-bit output Accumulator.

Table 7.23. ACCUMULATOR Configurations

SB_MAC16 Function/ Parameter Settings	**B_SIGNED**	**A_SIGNED**	**MODE_8x8**	**BOTADDSUB_CARRYSELECT**		**BOTADDSUB_UPPERINPUT**	**BOTADDSUB_LOWERINPUT**		**BOTOUTPUT_SELECT**		**TOPADDSUB_CARRYSELECT**		**TOPADDSUB_UPPERINPUT**	**TOPADDSUB_LOWERINPUT**		**TOPOUTPUT_SELECT**		**PIPELINE_16x16_MULT_REG2**	**PIPELINE_16x16_MULT_REG1**	**BOT_8x8_MULT_REG**	**TOP_8x8_MULT_REG**	**D_REG**	**B_REG**	**A_REG**	**C_REG**
	CBIT[24]	**CBIT[23]**	**CBIT[22]**	**CBIT[21]**	**CBIT[20]**	**CBIT[19]**	**CBIT[18]**	**CBIT[17]**	**CBIT[16]**	**CBIT[15]**	**CBIT[14]**	**CBIT[13]**	**CBIT[12]**	**CBIT[11]**	**CBIT[10]**	**CBIT[9]**	**CBIT[8]**	**CBIT[7]**	**CBIT[6]**	**CBIT[5]**	**CBIT[4]**	**CBIT[3]**	**CBIT[2]**	**CBIT[1]**	**CBIT[0]**
16-bit ACCUMULATOR																									
acc_16_all_pipelined_unsigned	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1
acc_16_bypassed_unsigned	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
32-bit ACCUMULATOR																									
acc_32_all_pipelined_unsigned	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	1	1	1
acc_32_all_pipelined_cascaded_unsigned	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	1	1	1
acc_32_all_pipelned_cin_unsigned	0	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	1	1	1
acc_32_bypassed_unsigned	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
acc_32_bypassed_cascaded_unsigned	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
acc_32_bypassed_cin_unsigned	0	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Add/Subtract Configurations

16-bit ADDSUB: 16-bit input, 16-bit output ADDSUB.

32-bit ADDSUB: 32-bit input, 32-bit output ADDSUB.

Table 7.24. Add/Subtract Configurations

SB_MAC16 Function/ Parameter Settings	**B_SIGNED**	**A_SIGNED**	**MODE_8x8**	**BOTADDSUB_CARRYSELECT**		**BOTADDSUB_UPPERINPUT**	**BOTADDSUB_LOWERINPUT**		**BOTOUTPUT_SELECT**		**TOPADDSUB_CARRYSELECT**		**TOPADDSUB_UPPERINPUT**	**TOPADDSUB_LOWERINPUT**		**TOPOUTPUT_SELECT**		**PIPELINE_16x16_MULT_REG2**	**PIPELINE_16x16_MULT_REG1**	**BOT_8x8_MULT_REG**	**TOP_8x8_MULT_REG**	**D_REG**	**B_REG**	**A_REG**	**C_REG**
	CBIT[24]	**CBIT[23]**	**CBIT[22]**	**CBIT[21]**	**CBIT[20]**	**CBIT[19]**	**CBIT[18]**	**CBIT[17]**	**CBIT[16]**	**CBIT[15]**	**CBIT[14]**	**CBIT[13]**	**CBIT[12]**	**CBIT[11]**	**CBIT[10]**	**CBIT[9]**	**CBIT[8]**	**CBIT[7]**	**CBIT[6]**	**CBIT[5]**	**CBIT[4]**	**CBIT[3]**	**CBIT[2]**	**CBIT[1]**	**CBIT[0]**
16-bit ADDSUB																									
add_sub_16_all_pipelined_ unsigned	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1
add_sub_16_ bypassed_unsigned	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
32-bit ADDSUB																									
add_sub_32_all_pipelined_ unsigned	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	1	1	1	1
add_sub_32_all_pipelined_ cascaded_unsigned	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	1	1	1	1
add_sub_32_all_pipelined_cin_ unsigned	0	0	1	1	1	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	1	1	1	1
add_sub_32_bypassed_ unsigned	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
add_sub_32_bypassed_cascade d_unsigned	0	0	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
add_sub_32_bypassed_cin_ unsigned	0	0	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Multiply Add/Subtract Configurations

16-bit Mult-Add/Sub: 8×8 input, 16-bit output Mult-Add/Sub.

32-bit Mult-Add/Sub: 16×16 input, 32-bit output Mult-Add/Sub.

Table 7.25. Multiply Add/Subtract Configurations

SB_MAC16 Function/ Parameter Settings	B SIGNED	A SIGNED	MODE 8x8	BOTADDSUB_CARRYSELECT		BOTADDSUB UPPERINPUT	BOTADDSUB_LOWERINPUT		BOTOUTPUT_SELECT		TOPADDSUB_CARRYSELECT		TOPADDSUB UPPERINPUT	TOPADDSUB_LOWERINPUT		TOPOUTPUT_SELECT		PIPELINE 16x16 MULT REG2	PIPELINE 16x16 MULT REG1	BOT 8x8 MULT REG	TOP 8x8 MULT REG	D REG	B REG	A REG	C REG
	CBIT[24]	CBIT[23]	CBIT[22]	CBIT[21]	CBIT[20]	CBIT[19]	CBIT[18]	CBIT[17]	CBIT[16]	CBIT[15]	CBIT[14]	CBIT[13]	CBIT[12]	CBIT[11]	CBIT[10]	CBIT[9]	CBIT[8]	CBIT[7]	CBIT[6]	CBIT[5]	CBIT[4]	CBIT[3]	CBIT[2]	CBIT[1]	CBIT[0]
16-bit Mult-Add/Sub																									
mult_add_sub_16_all_pipelined_unsigned	0	0	1	0	0	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1
mult_add_sub_16_intermediate_register_bypassed_unsigned	0	0	1	0	0	1	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	1	1	1	1
mult_add_sub_16_bypassed_ unsigned	0	0	1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
32-bit Mult-Add/Sub																									
mult_add_sub_32_all_pipelined_ unsigned	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1
mult_add_sub_32_all_pipelined_ cascaded_unsigned	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1
mult_add_sub_32_all _pipelined_cin_unsigned	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	1	1	1	1	1	1	1	1	1
mult_add_sub_32_intermediate_register_bypassed_unsigned	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	0	1	1	1	1
mult_add_sub_32_intermediate_register_bypassed_cascaded_unsigned	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	0	1	1	1	1
mult_add_sub_32_intermediate_register_bypassed_cin_unsigned	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	0	1	1	1	1
mult_add_sub_32_bypassed_ unsigned	0	0	0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
mult_add_sub_32_bypassed_ cascaded_unsigned	0	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
mult_add_sub_32_bypassed_ cin_unsigned	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
mult_add_sub_32_intermediate_register_ bypassed_signed	1	1	0	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	0	1	1	1	1

Verilog Instantiation

```
SB_MAC16 i_sbmac16
       (
                  .A(A_i),
                  .B(B_i),
                  .C(C_i),
                  .D(D_i),
                  .O(O),
                  .CLK(SYSCLK_i),
                  .CE(CE_i),
                  .IRSTTOP(RST_i),
                  .IRSTBOT(RST_i),
                  .ORSTTOP(RST_i),
                  .ORSTBOT(RST_i),
                  .AHOLD(AHOLD_i),
                  .BHOLD(BHOLD_i),
                  .CHOLD(CHOLD_i),
                  .DHOLD(DHOLD_i),
                  .OHOLDTOP(HLDTOPOUT_i),
                  .OHOLDBOT(HLDBOTOUT_i),
                  .OLOADTOP(LOADTOP_i),
                  .OLOADBOT(LOADBOT_i),
                  .ADDSUBTOP(ADDSUBTOP_i),
                  .ADDSUBBOT(ADDSUBBOT_i),
                  .CO(CO),
                  .CI(CI),
                  //MAC cascading ports.
                  .ACCUMCI(),
                  .ACCUMCO(),
                  .SIGNEXTIN(),
                  .SIGNEXTOUT()
       );

// mult_8x8_all_pipelined_unsigned  [24:0] = 001_0000010_0000010_0111_0110
// Read configuration settings [24:0] from left to right while filling the instance
parameters.

defparam i_sbmac16. B_SIGNED                              = 1'b0 ;
defparam i_sbmac16. A_SIGNED                              = 1'b0 ;
defparam i_sbmac16. MODE_8x8                              = 1'b1 ;

defparam i_sbmac16. BOTADDSUB_CARRYSELECT                 = 2'b00 ;
defparam i_sbmac16. BOTADDSUB_UPPERINPUT                  = 1'b0 ;
defparam i_sbmac16. BOTADDSUB_LOWERINPUT                  = 2'b00 ;
defparam i_sbmac16. BOTOUTPUT_SELECT               = 2'b10 ;

defparam i_sbmac16. TOPADDSUB_CARRYSELECT                 = 2'b00 ;
defparam i_sbmac16. TOPADDSUB_UPPERINPUT                  = 1'b0 ;
defparam i_sbmac16. TOPADDSUB_LOWERINPUT                  = 2'b00 ;
defparam i_sbmac16. TOPOUTPUT_SELECT               = 2'b10 ;
defparam i_sbmac16. PIPELINE_16x16_MULT_REG2        = 1'b0 ;
defparam i_sbmac16. PIPELINE_16x16_MULT_REG1        = 1'b1 ;
defparam i_sbmac16. BOT_8x8_MULT_REG                = 1'b1 ;
defparam i_sbmac16. TOP_8x8_MULT_REG                = 1'b1 ;
defparam i_sbmac16. D_REG                           = 1'b0 ;
defparam i_sbmac16. B_REG                           = 1'b1 ;
defparam i_sbmac16. A_REG                           = 1'b1 ;
defparam i_sbmac16. C_REG                           = 1'b0 ;
```

7.3. iCE40 UltraLite (iCE40UL) Hard Macros

This section describes the following dedicated hard macro primitives available in the iCE40 UltraLite device.

- SB_HFOSC
- SB_LFOSC
- SB_RGBA_DRV
- SB_IR400_DRV
- SB_BARCODE_DRV
- SB_IR500_DRV
- SB_LEDDA_IP
- SB_IR_IP
- SB_IO_OD
- SB_I2C_FIFO

7.3.1. SB_HFOSC

SB_HFOSC primitive generates 48 MHz nominal clock frequency within ± 10% variation, with user-programmable divider value of 1, 2, 4, and 8. The HFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.

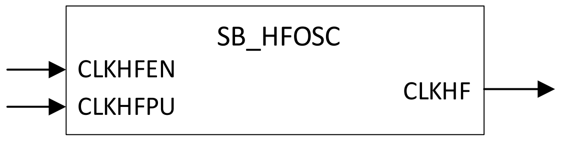

Figure 7.16. SB_HFOSC

Table 7.26. SB_HFOSC Port Signals

Signal Name	Direction	Description
CLKHFPU	Input	Power up the HFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.
CLKHFEN	Input	Enable the clock output. Enable should be low for the 100 µs power up period. Active High.
CLKHF	Output	HF Oscillator output.

Parameters

Table 7.27. SB_HFOSC Parameters

Parameter	Parameter Values	Description
CLKHF_DIV	"0b00"	Sets 48 MHz HFOSC output.
	"0b01"	Sets 24 MHz HFOSC output.
	"0b10"	Sets 12 MHz HFOSC output.
	"0b11"	Sets 6 MHz HFOSC output

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

- Input CLKHFEN: Logic "0"
- Input CLKHFPU: Logic "0"

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */ Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_HFOSC OSCInst0 (
       .CLKHFEN(ENCLKHF),
       .CLKHFPU(CLKHF_POWERUP),
       .CLKHF(CLKHF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;

defparam OSCInst0.CLKHF_DIV = "0b00";
```

7.3.2. SB_LFOSC

SB_LFOSC primitive generates 10 kHz nominal clock frequency within ±10% variation. There is no divider on the LFOSC. The LFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.

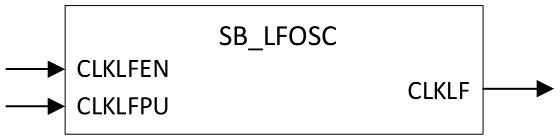

Figure 7.17. SB_LFOSC

Table 7.28. SB_LFOSC Port Signals

Signal Name	Direction	Description
CLKLFPU	Input	Power up the LFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.
CLKLFEN	Input	Enable the clock output. Enable should be low for the 100 µs power up period. Active High.
CLKLF	Output	LF Oscillator output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CLKLFEN: Logic "0"

Input CLKLFPU: Logic "0"

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */ Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_LFOSC OSCInst1 (
       .CLKLFEN(ENCLKLF),
       .CLKLFPU(CLKLF_POWERUP),
       .CLKLF(CLKLF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

7.3.3. SB_RGBA_DRV

SB_RGBA_DRV primitive is the RGB LED drive module which contains three dedicated open drain I/O pins for RGB LED outputs. Each of the RGB LED output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_RGB_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

The primitive allows configuration of each of the three RGB LED outputs individually. When the RGBx_CURRENT parameter of RGBx output is set to "0b000000", then SB_IO_OD can be used to drive the package pin.

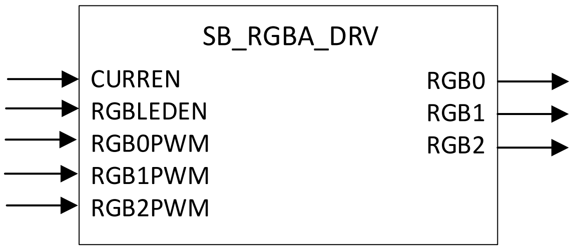

Figure 7.18. SB_RGBA_DRV

Table 7.29. SB_RGBA_DRV Port Signals

Signal Name	Direction	Description
CURREN	Input	Enable the mixed signal control block to supply reference current to the IR drivers. When it is not enabled, (CURREN=0), no current is supplied, and the IR drivers are powered down. Enabling the mixed signal control block takes 100 µs to reach a stable reference current value.
RGBLEDEN	Input	Enable the SB_RGB_DRV primitive. Active High.
RGB0PWM	Input	Input data to drive RGB0 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB1PWM	Input	Input data to drive RGB1 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB2PWM	Input	Input data to drive RGB2 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB0	Output	RGB0 LED output.
RGB1	Output	RGB1 LED output.
RGB2	Output	RGB2 LED output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CURREN	: Logic “0”
Input RGBLEDEN	: Logic “0”
Input RGB0PWM	: Logic “0”
Input RGB1PWM	: Logic “0”
Input RGB2PWM	: Logic “0”

Parameters

The SB_RGBA_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0 "; Parameter values:

"0b0" = Full Current Mode (Default).

"0b1" = Half Current Mode.

Parameter RGB0_CURRENT = "0b000000";

Parameter RGB1_CURRENT = "0b000000";

Parameter RGB2_CURRENT = "0b000000";

Parameter values:

"0b000000" = 0 mA. // Set this value to use the associated SB_IO_OD instance at RGB // LED location.

"0b000001" = 4 mA for Full Mode; 2 mA for Half Mode

"0b000011" = 8 mA for Full Mode; 4 mA for Half Mode

"0b000111" = 12 mA for Full Mode; 6 mA for Half Mode.

"0b001111" = 16 mA for Full Mode; 8 mA for Half Mode

"0b011111" = 20 mA for Full Mode; 10 mA for Half Mode.

"0b111111" = 24 mA for Full Mode; 12 mA for Half Mode.

Verilog Instantiation

```
SB_RGBA_DRV RGBA_DRIVER (
        .CURREN(ENABLE_CURR),
        .RGBLEDEN(ENABLE_RGBDRV),
        .RGB0PWM(RGB0),
        .RGB1PWM(RGB1),
        .RGB2PWM(RGB2),
        .RGB0(LED0),
        .RGB1(LED1),
        .RGB2(LED2)
);

defparam RGBA_DRIVER.CURRENT_MODE = "0b0"; defparam RGBA_DRIVER.RGB0_CURRENT =
"0b111111"; defparam RGBA_DRIVER.RGB1_CURRENT = "0b111111" ; defparam
RGBA_DRIVER.RGB2_CURRENT = "0b111111";
```

7.3.4. SB_IR400_DRV

SB_IR400_DRV primitive is the IR driver which contains a single dedicated open drain I/O pin for IRLED output. The IRLED output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_IR400_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

When the IR400_CURRENT parameter is set to "0b00000000", then SB_IO_OD can be used to drive the package pin.

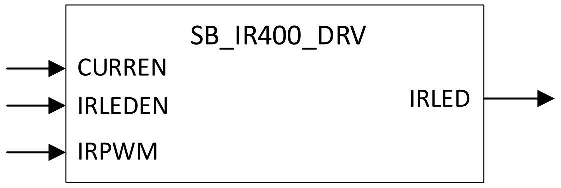

Figure 7.19. SB_IR400_DRV

Table 7.30. SB_IR400_DRV Port Signals

Signal Name	Direction	Description
CURREN	Input	Enable the mixed signal control block to supply reference current to the IR drivers. When it is not enabled (CURREN=0), no current is supplied, and the IR drivers are powered down. Enabling the mixed signal control block takes 100 µs to reach a stable reference current value.
IRLEDEN	Input	Enable the SB_IR400_DRV primitive. Active High.
IRPWM	Input	PWM Input data to drive IRLED pin.
IRLED	Output	IR LED output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CURREN	: Logic “0”
Input IRLEDEN	: Logic “0”
Input IRPWM	: Logic “0”

Parameter

The SB_IR400_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0 ";

Parameter values:

"0b0" = Full Current Mode (Default).

"0b1" = Half Current Mode.

Parameter IR400_CURRENT = "0b00000000";

Parameter Values:

"0b0000000000"; = 0 mA. //This is the setting to tristate the IR output to allow it to be used as GPIO (SB_IO_OD)

"0b00000001"; = 50 mA for Full Mode; 25 mA for Half Mode.

"0b00000011"; = 100 mA for Full Mode; 50 mA for Half Mode.

"0b00000111"; = 150 mA for Full Mode; 75 mA for Half Mode.

"0b00001111"; = 200 mA for Full Mode; 100 mA for Half Mode.

"0b00011111"; = 250 mA for Full Mode; 125 mA for Half Mode.

"0b00111111"; = 300 mA for Full Mode; 150 mA for Half Mode.

"0b01111111"; = 350 mA for Full Mode; 175 mA for Half Mode.

"0b11111111"; = 400 mA for Full Mode; 200 mA for Half Mode.

Verilog Instantiation

```
SB_IR400_DRV IRDRVinst (
       .CURREN(ENABLE_CURRENT),
       .IRLEDEN(ENABLE_IRDRV),
       .IRPWM(IR_PWMINPUT),
       .IRLED(IR_LEDOUT)
);

defparam IRDRVinst.CURRENT_MODE = "0b0";
defparam IRDRVinst.IR400_CURRENT = "0b1111111111";
```

7.3.5. SB_BARCODE_DRV

SB_BARCODE_DRV primitive contains a single dedicated open drain I/O pin for BARCODE output. The BARCODE output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_BARCODE_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

When the BARCODE_CURRENT parameter is set to "0b0000", SB_IO_OD can be used to drive the package pin.

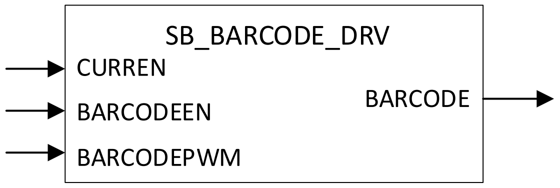

Figure 7.20. SB_BARCODE_DRV

Table 7.31. SB_BARCODE_DRV Port Signals

Signal Name	Direction	Description
CURREN	Input	Enable the mixed signal control block to supply reference current to the IR drivers. When it is not enabled (CURREN=0), no current is supplied, and the IR drivers are powered down. Enabling the mixed signal control block takes 100 µs to reach a stable reference current value.
BARCODEN	Input	Enable the SB_BARCODE_DRV primitive. Active High.
BARCODEPWM	Input	PWM Input data to drive BARCODE pin.
BARCODE	Output	BARCODE output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CURREN	: Logic “0”
Input BARCODEEN	: Logic “0”
Input BARCODEPWM	: Logic “0”

Parameter

The SB_BARCODE_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0";

Parameter values:

"0b0" = Full Current Mode (Default).

"0b1" = Half Current Mode.

Parameter BARCODE_CURRENT = "0b0000";

Parameter values:

"0b0000" = 0 mA.//This is the setting to tri-state the BARCODE output to allow it to be used // as GPIO (SB_IO_OD)

"0b0001" = 16.6 mA for Full Mode; 8.3 mA for Half Mode,

"0b0011" = 33.3 mA for Full Mode; 16.6 mA for Half Mode,

"0b1001" = 66.6 mA for Full Mode; 33.3 mA for Half Mode,

"0b1010" = 83.3 mA for Full Mode; 41.6 mA for Half Mode,

"0b0111" = 50 mA for Full Mode; 25 mA for Half Mode,

"0b1111" = 100 mA for Full Mode; 50 mA for Half Mode

Verilog Instantiation

```
SB_BARCODE_DRV BARCODEDRVinst (
       .CURREN(ENABLE_CURRENT),
       .BARCODEEN(ENABLE_BARCODEDRV),
       .BARCODEPWM(BARCODE_PWMINPUT),
       .BARCODE(BARCODEOUT)
);
defparam BARCODEDRVinst.CURRENT_MODE = "0b0";
defparam BARCODEDRVinst.BARCODE_CURRENT = "0b1111";
```

7.3.6. SB_IR500_DRV

SB_IR500_DRV primitive is the IR driver which contains a two dedicated open drain I/O pin for IRLED1, IRLED2 outputs. The IRLED outputs are bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_IR500_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

When the IR4500_CURRENT parameter is set to "0b00000000", then SB_IO_OD can be used to drive the package pin.

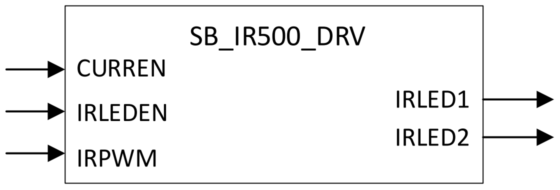

Figure 7.21. SB_IR500_DRV

SB_IR500_DRV DRC Rule

This primitive cannot be instantiated along with SB_BARCODE_DRV or SB_IR400_DRV instance.

Table 7.32. SB_IR500_DRV Port Signals

Signal Name	Direction	Description
CURREN	Input	Enable the mixed signal control block to supply reference current to the IR drivers. When it is not enabled (CURREN=0), no current is supplied, and the IR drivers are powered down. Enabling the mixed signal control block takes 100 μs to reach a stable reference current value.
IRLEDEN	Input	Enable the SB_IR400_DRV primitive. Active High.
IRPWM	Input	PWM Input data to drive IRLED pin.
IRLED1	Output	IR LED output 1.
IRLED2	Output	IR LED output 2.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CURREN	: Logic “0”
Input IRLEDEN	: Logic “0”
Input IRPWM	: Logic “0”

Parameters

The SB_IR500_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0";

Parameter values:

"0b0"; = Full Current Mode (Default).

"0b1"; = Half Current Mode.

Parameter IR500_CURRENT = “0b000000000000”;

Parameter values:

"0b000000000000"; = 0 mA. // This is the setting to tristate the BARCODE output to allow it to // be used as GPIO (SB_IO_OD).

"0b000000000111"; = 50 mA for Full Mode; 25 mA for Half Mode.

"0b000000001111"; = 100 mA for Full Mode; 50 mA for Half Mode.

"0b000000011111"; = 150 mA for Full Mode; 75 mA for Half Mode.

"0b000000111111"; = 200 mA for Full Mode; 100 mA for Half Mode.

"0b000001111111"; = 250 mA for Full Mode; 125 mA for Half Mode.

"0b000011111111"; = 300 mA for Full Mode; 150 mA for Half Mode.

"0b000111111111"; = 350 mA for Full Mode; 175 mA for Half Mode.

"0b001111111111"; = 400 mA for Full Mode; 200 mA for Half Mode.

"0b011111111111"; = 450 mA for Full Mode; 225 mA for Half Mode.

"0b111111111111"; = 500 mA for Full Mode; 250 mA for Half Mode.

Verilog Instantiation

```
SB_IR500_DRV IRDRVinst (
       .CURREN(ENABLE_CURRENT),
       .IRLEDEN(ENABLE_IRDRV),
       .IRPWM(IR_PWMINPUT),
       .IRLED(IR_LEDOUT)
);

defparam IRDRVinst.CURRENT_MODE = "0b0";
defparam IRDRVinst.IR500_CURRENT = "0b1111111111";
```

7.3.7. SB_LEDDA_IP

SB_LEDDA_IP primitive generates the RGB PWM outputs for the RGB LED drivers. The IP contains registers that are programmed in by the SCI bus interface signals.

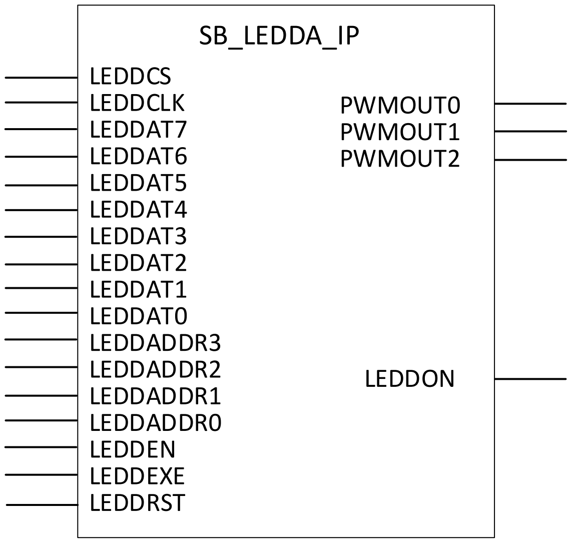

Figure 7.22. SB_LEDDA_IP

Table 7.33. SB_LEDDA_IP Port Signals

Signal Name	Direction	Description
LEDDCS	Input	CS to write LEDD IP registers
LEDDCLK	Input	Clock to write LEDD IP registers
LEDDDAT7	Input	Bit 7 data to write into the LEDD IP registers
LEDDDAT6	Input	Bit 6 data to write into the LEDD IP registers
LEDDDAT5	Input	Bit 5 data to write into the LEDD IP registers
LEDDDAT4	Input	Bit 4 data to write into the LEDD IP registers
LEDDDAT3	Input	Bit 3 data to write into the LEDD IP registers
LEDDDAT2	Input	Bit 2 data to write into the LEDD IP registers
LEDDDAT1	Input	Bit 1 data to write into the LEDD IP registers
LEDDDAT0	Input	Bit 0 data to write into the LEDD IP registers
LEDDADDR3	Input	LEDD IP register address bit 3
LEDDADDR2	Input	LEDD IP register address bit 2
LEDDADDR1	Input	LEDD IP register address bit 1
LEDDADDR0	Input	LEDD IP register address bit 0
LEDDDEN	Input	Data enable input to indicate data and address are stable.
LEDDEXE	Input	Enable the IP to run the blinking sequence. When it is LOW, the sequence stops at the nearest OFF state.
LEDDRST	Input	Device level reset signal to reset all internal registers during the device configuration. This port is not accessible to user signals.
PWMOUT0	Output	PWM output 0
PWMOUT1	Output	PWM output 1
PWMOUT2	Output	PWM output 2
LEDDON	Output	Indicating the LED is on

Default Signal Values

The iCEcube2 software assigns the logic “0” value to all unconnected input ports.

Verilog Instantiation

```
SB_LEDDA_IP  PWMgen_inst (
       .LEDDCS(ledd_cs),
       .LEDDCLK(led_clk),
       .LEDDDAT7(led_ip_data[7]),
       .LEDDDAT6(led_ip_data[6]),
       .LEDDDAT5(led_ip_data[5]),
       .LEDDDAT4(led_ip_data[4]),
       .LEDDDAT3(led_ip_data[3]),
       .LEDDDAT2(led_ip_data[2]),
       .LEDDDAT1(led_ip_data[1]),
       .LEDDDAT0(led_ip_data[0]),
       .LEDDADDR3(led_ip_addr[3]),
       .LEDDADDR2(led_ip_addr[2]),
       .LEDDADDR1(led_ip_addr[1]),
       .LEDDADDR0(led_ip_addr[0]),
       .LEDDDEN(led_ip_den),
       .LEDDEXE(led_ip_exe),
       .LEDDRST(led_ip_rst),
       .PWMOUT0(LED0),
       .PWMOUT1(LED1),
       .PWMOUT2(LED2),
       .LEDDON(led_on)
);
```

7.3.8. SB_IR_IP

SB_IR_IP primitive is the IR transceiver module. It generates or receives the modulated pulse for the IR driver primitives.

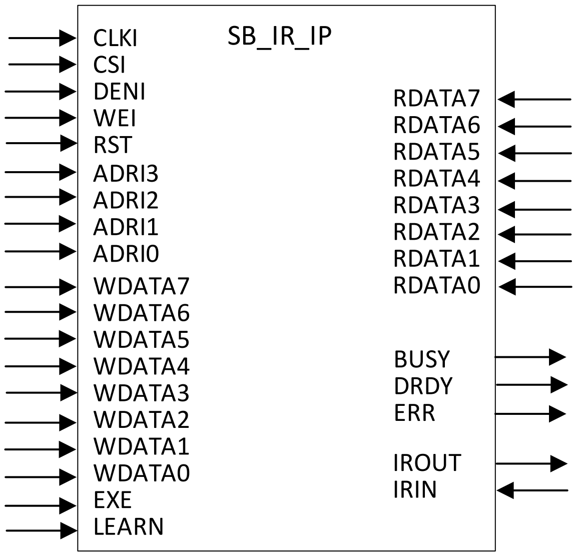

Figure 7.23. SB_IR_IP

Table 7.34. SB_IR_IP Port Signals

Signal Name	Direction	Description
CLKI	Input	Clock input for IR IP
CSI	Input	Select Signal. Active High to activate the IP. This usually connects to the output of the decoding logic from MSB of the address bus
DENI	Input	Data Enable. When asserted, indicates that the data and address on the IR Transceiver Control Bus are stabilized and ready to be captured.
WEI	Input	Data Write Enable. Asserted during WRITE and deasserted during READ cycle.
ADRI3	Input	Control Register Address Bit 3
ADRI2	Input	Control Register Address Bit 2
ADRI1	Input	Control Register Address Bit 1
ADRI0	Input	Control Register Address Bit 0
WDATA7	Input	Write Data Input Bit 7
WDATA6	Input	Write Data Input Bit 6
WDATA5	Input	Write Data Input Bit 5
WDATA4	Input	Write Data Input Bit 4
WDATA3	Input	Write Data Input Bit 3
WDATA2	Input	Write Data Input Bit 2
WDATA1	Input	Write Data Input Bit 1
WDATA0	Input	Write Data Input Bit 0
RDATA7	Output	Read Data Output Bit 7
RDATA6	Output	Read Data Output Bit 6
RDATA5	Output	Read Data Output Bit 5
RDATA4	Output	Read Data Output Bit 4
RDATA3	Output	Read Data Output Bit 3
RDATA2	Output	Read Data Output Bit 2
RDATA1	Output	Read Data Output Bit 1
RDATA0	Output	Read Data Output Bit 0
EXE	Input	Execute. When asserted, starts the IR Transceiver Hard IP to transmit or receive IR data
LEARN	Input	Learning Mode control. When asserted, the IR Transceiver is in learning mode. The IR Transceiver receives data instead of transmit data.
BUSY	Output	Busy status output
DRDY	Output	Data Buffer Ready status output
ERR	Output	Data Error status
RST	Input	Device level reset signal to reset all internal registers and IROUT signal to OFF state during the device configuration. This port is not accessible to user signals.
IRIN	Input	Modulated ON/OFF pulse from IR sensor.
IROUT	Output	Modulated ON/OFF pulse for IR Transmit.

Default Signal Values

The iCEcube2 software assigns the logic “0” value to all unconnected input ports.

Verilog Instantiation

```
SB_IR_IP IRIP_inst (
       .CLKI(sysclk_i),
       .CSI(csi_i),
       .DENI(deni_i),
       .WEI(wei_i),
       .ADRI3(addri_i[3]),
       .ADRI2(addri_i[2]),
       .ADRI1(addri_i[1]),
       .ADRI0(addri_i[0]),
       .WDATA7(wdata_i[7]),
       .WDATA6(wdata_i[6]),
       .WDATA5(wdata_i [5]),
       .WDATA4(wdata_i [4]),
       .WDATA3(wdata_i [3]),
       .WDATA2(wdata_i [2]),
       .WDATA1(wdata_i [1]),
       .WDATA0(wdata_i [0]),
       .RDATA7(rdata_o[7]),
       .RDATA6(rdata_o[6]),
       .RDATA5(rdata_o[5]),
       .RDATA4(rdata_o[4]),
       .RDATA3(rdata_o[3]),
       .RDATA2(rdata_o[2]),
       .RDATA1(rdata_o[1]),
       .RDATA0(rdata_o[0]),
       .EXE(exe_i),
       .LEARN(learn_i),
       .BUSY(busy_o),
       .DRDY(drdy_o),
       .ERR(err_o),
       .RST(rst_i),
       .IRIN(irin_i),
       .IROUT(irpulse_w)
);
```

7.3.9. SB_IO_OD

The SB_IO_OD is the open drain I/O primitive. When the Tristate output is enabled, the I/O pulls down the package pin signal to zero. The following figure and Verilog template illustrate the complete user accessible logic diagram, and its Verilog instantiation.

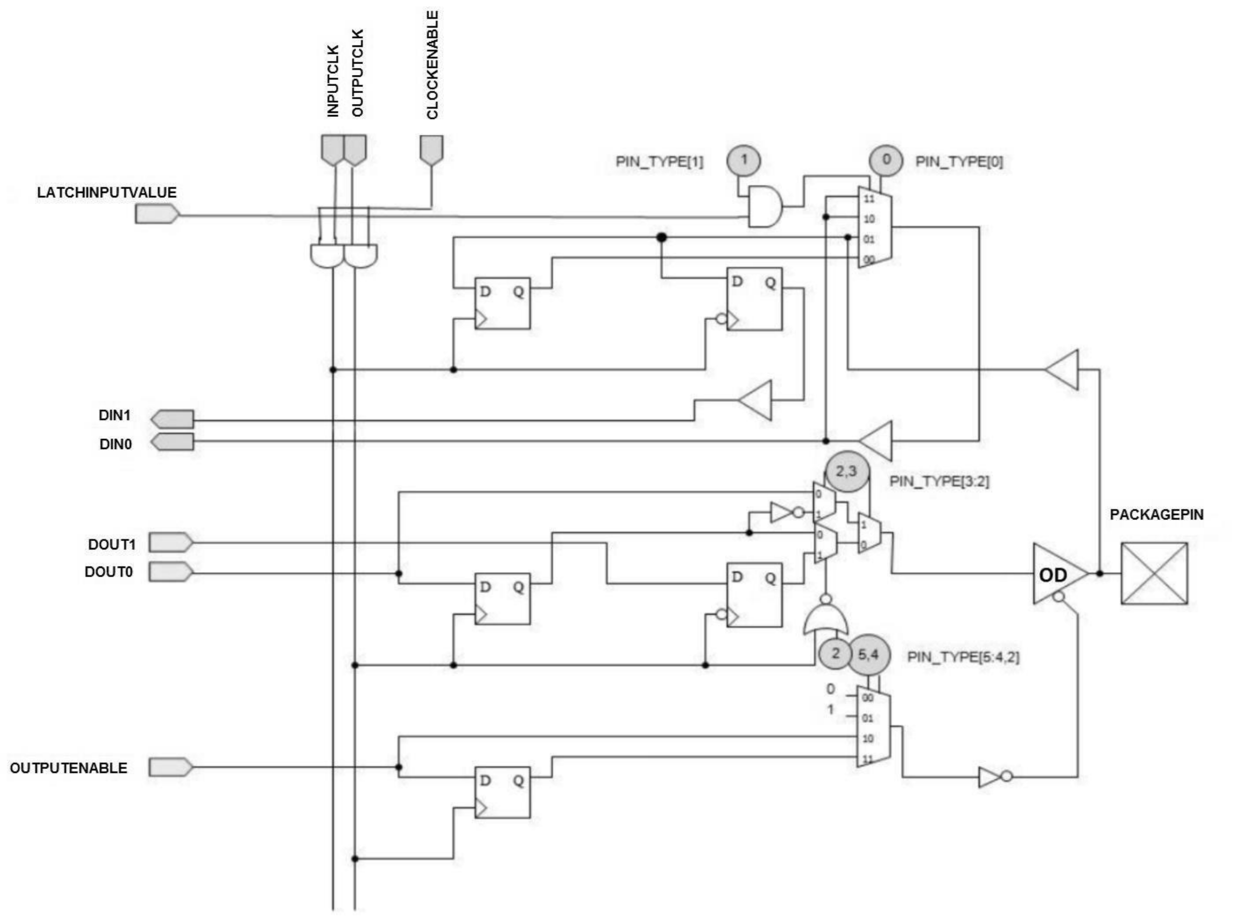

Figure 7.24. SB_IO_OD

Table 7.35. SB_IO_OD Port Signals

Signal Name	Direction	Description
PACKAGEPIN	Bidirectional	Bidirectional I/O pin.
LATCHINPUTVALUE	Input	Latches/Holds the input data.
CLOCKENABLE	Input	Clock enable signal.
IRLED1	Output	IR LED output 1.
IRLED2	Output	IR LED output 2.
INPUTCLK	Input	Clock for the input registers.
OUTPUTCLK	Input	Clock for the output registers.
OUTPUTENABLE	Input	Enable Tristate output. Active high.
DOUT0	Input	Data to package pin.
DOUT1	Input	Data to package pin.
DIN0	Output	Data from package pin.
DIN1	Output	Data from package pin.

Default Signal Values

The iCEcube2 software assigns the logic “0” value to all unconnected input ports except for CLOCKENABLE.

Note that explicitly connecting logic “1” value to port CLOCKENABLE results in a non-optimal implementation, since extra LUT is used to generate the Logic “1”. If your intention is to keep the Input and Output registers always enabled, it is recommended that port CLOCKENABLE to be left unconnected.

Parameters

Parameter PIN_TYPE = 6’b000000;

// See Input and Output Pin Function Tables in SB_IO.

// Default value of PIN_TYPE = 6’b000000

Parameter NEG_TRIGGER = 1’b0;

// Specify the polarity of all FFs in the I/O to be falling edge when

NEG_TRIGGER = 1. Default is 1’b0, rising edge.

Input and Output Pin Function Tables

Refer SB_IO Input and Output Pin Functional Table for the PIN_TYPE settings. Some of the output pin configurations are not applicable for SB_IO_OD primitive.

Verilog Instantiation

```
SB_IO_OD     OpenDrainInst0 (
.PACKAGEPIN (PackagePin),              // User’s Pin signal name
.LATCHINPUTVALUE (latchinputvalue),          // Latches/holds the Input value
.CLOCKENABLE (clockenable),                  // Clock Enable common to input and
                                             // output clock
.INPUTCLK (inputclk),                        // Clock for the input registers
.OUTPUTCLK (outputclk),                      // Clock for the output registers
.OUTPUTENABLE (outputenable),                // Output Pin Tristate/Enable
                                             // control
.DOUT0 (dout0),                              // Data 0 – out to Pin/Rising clk
                                             // edge
.DOUT1 (dout1),                              // Data 1 - out to Pin/Falling clk
                                             // edge
.DIN0 (din0),                          // Data 0 - Pin input/Rising clk
                                             // edge
.DIN1 (din1)                           // Data 1 – Pin input/Falling clk
                                             // edge
);

defparam OpenDrainInst0.PIN_TYPE = 6'b000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
```

7.3.10. SB _I2C_FIFO

The iCE40UL device supports two I2C hard IP primitives , located at bottom left corner and bottom right corner of the chip.

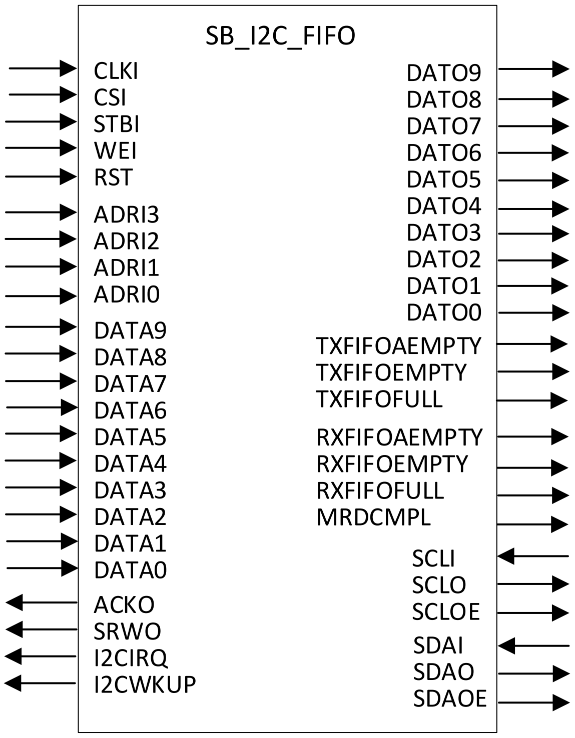

Figure 7.25. SB_I2C_FIFO

Table 7.36. SB_I2C_FIFO Port Signals

Signal Name	Direction	Description
CLKI	Input	System Clock input
CSI	Input	Select Signal. Active High to activate the IP. This usually connects to the output of the decoding logic from MSB of the address bus
STBI	Input	Strobe Signal
WEI	Input	Write Enable Signal
ADRI3	Input	Control Register Address Bit 3
ADRI2	Input	Control Register Address Bit 2
ADRI1	Input	Control Register Address Bit 1
ADRI0	Input	Control Register Address Bit 0
DATI9	Input	Data Input Bit 9
DATI8	Input	Data Input Bit 8
DATI7	Input	Data Input Bit 7
DATI6	Input	Data Input Bit 6
DATI5	Input	Data Input Bit 5
DATI4	Input	Data Input Bit 4
DATI3	Input	Data Input Bit 3
DATI2	Input	Data Input Bit 2
DATI1	Input	Data Input Bit 1
DATI0	Input	Data Input Bit 0
DATO9	Output	Data Output Bit 9

Signal Name	Direction	Description
DATO8	Output	Data Output Bit 8
DATO7	Output	Data Output Bit 7
DATO6	Output	Data Output Bit 6
DATO5	Output	Data Output Bit 5
DATO4	Output	Data Output Bit 4
DATO3	Output	Data Output Bit 3
DATO2	Output	Data Output Bit 2
DATO1	Output	Data Output Bit 1
DATO0	Output	Data Output Bit 0
ACKO	Output	Acknowledgement
I2CIRQ	Output	I2C Interrupt output
I2CWKUP	Output	I2C Wake up from Standby signal
SRWO	Output	Target RW "1" Controller receiving / Target transmitting "0" Controller transmitting / Target receiving
SCLI	Input	Serial Clock Input
SCLO	Output	Serial Clock Output
SCLOE	Output	Serial Clock Output Enable. Active High
SDAI	Input	Serial Data Input
SDAO	Output	Serial Data Output
SDAOE	Output	Serial Data Output Enable. Active High
FIFORST*	Input	RESET for the FIFO logic
TXFIFOAEMPTY*	Output	TX FIFO Status Signal
TXFIFOEMPTY*	Output	TX FIFO Status Signal
TXFIFOFULL*	Output	TX FIFO Status Signal
RXFIFOAFULL*	Output	RX FIFO Status Signal
RXFIFOFULL*	Output	RX FIFO Status Signal
RXFIFOEMPTY*	Output	RX FIFO Status Signal
MRDCMPL*	Output	Controller Read Complete (only valid for Controller Read Mode)

*__Note__: Only available if I2C_FIFO_ENB = ENABLED.

Parameters

Table 7.37. SB_I2C_FIFO Parameters

I2C Location	Parameter	Parameter Default	Description
Left Side Corner	I2C_SLAVE _ADDR	0b1111100001	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
Right Side	I2C_SLAVE _ADDR	0b1111100010	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.

Synthesis Attribute

I2C_CLK_DIVIDER

Synthesis attribute "I2C_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCLO output with respect to the SBCLKI input clock frequency.

/* synthesis I2C_CLK_DIVIDER= Divide Value */

Divide Value: 0, 1, 2, 3 … 1023. Default is 0.

SDA_INPUT_DELAYED

SDA_INPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAI signal.

/* synthesis SDA_INPUT_DELAYED= value */

Value:

0: No delay.

1: Add 50 ns delay. (Default value).

SDA_OUTPUT_DELAYED

SDA_OUTPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAO signal.

/* synthesis SDA_OUTPUT_DELAYED= value */

Value:

0: No delay (Default value).

1: Add 50 ns delay.

I2C_FIFO_ENB

"I2C_FIFO_ENB" attribute is used to enable or disable FIFO

/* synthesis I2C_FIFO_ENB= [value] */

Value:

ENABLED: FIFO mode is enabled.

DISABLED: FIFO mode is disabled.

7.4. iCE40 UltraPlus Hard Macros

This section describes the following dedicated hard macro primitives available in iCE40 UltraPlus device.

- SB_HFOSC
- SB_LFOSC
- SB_RGBA_DRV
- SB_LEDDA_IP
- SB_IO_OD
- SB_I2C
- SB_SPI
- SB_MAC16
- SB_SPRAM256KA
- SB_IO_I3C

7.4.1. SB_HFOSC

SB_HFOSC primitive generates 48 MHz nominal clock frequency within +/- 10% variation, with user-programmable divider value of 1, 2, 4, and 8. The HFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.

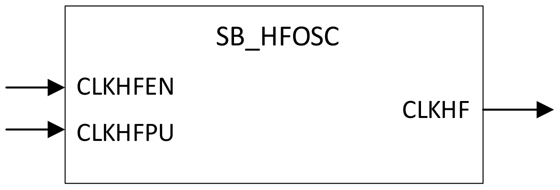

Figure 7.26. SB_HFOSC

Table 7.38. SB_HFOSC Port Signals

Signal Name	Direction	Description
CLKHFPU	Input	Power up the HFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.
CLKHFEN	Input	Enable the clock output. Enable should be low for the 100 µs power up period. Active High.
CLKHF	Output	HF Oscillator output.

Parameters

Table 7.39. SB_PLL40_CORE Parameters

Parameter	Parameter Values	Description
CLKHF_DIV	"0b00"	Sets 48 MHz HFOSC output.
	"0b01"	Sets 24 MHz HFOSC output.
	"0b10"	Sets 12 MHz HFOSC output.
	"0b11"	Sets 6 MHz HFOSC output

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CLKHFEN: Logic “0”

Input CLKHFPU: Logic “0”

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_HFOSC OSCInst0 (
.CLKHFEN(ENCLKHF),
.CLKHFPU(CLKHF_POWERUP),
.CLKHF(CLKHF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;

defparam OSCInst0.CLKHF_DIV = "0b00";
```

7.4.2. SB_LFOSC

SB_LFOSC primitive generates 10 kHz nominal clock frequency within ±10% variation. There is no divider on the LFOSC. The LFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.

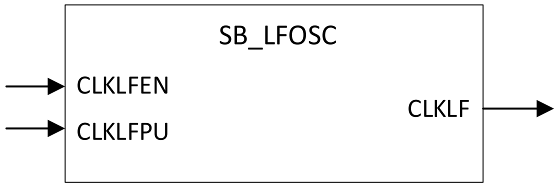

Figure 7.27. SB_LFOSC

Table 7.40. SB_LFOSC Port Signals

Signal Name	Direction	Description
CLKLFPU	Input	Power up the LFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.
CLKLFEN	Input	Enable the clock output. Enable should be low for the 100 µs power up period. Active High.
CLKLF	Output	LF Oscillator output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CLKLFEN: Logic “0”

Input CLKLFPU: Logic “0”

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

Verilog Instantiation

```
SB_LFOSC OSCInst1 (
       .CLKLFEN(ENCLKLF),
       .CLKLFPU(CLKLF_POWERUP),
       .CLKLF(CLKLF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

7.4.3. SB_RGBA_DRV

SB_RGBA_DRV primitive is the RGB LED drive module which contains three dedicated open drain I/O pins for RGB LED outputs. Each of the RGB LED output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_RGB_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.

The primitive allows configuration of each of the three RGB LED outputs individually. When the RGBx_CURRENT parameter of RGBx output is set to "0b000000", then SB_IO_OD can be used to drive the package pin.

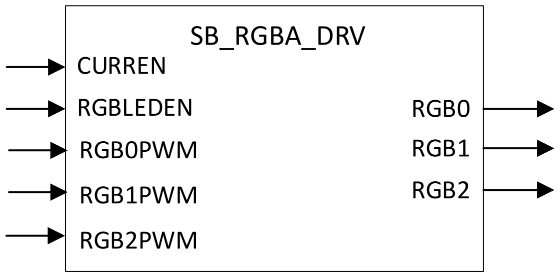

Figure 7.28. SB_RGBA_DRV

Table 7.41. SB_RGBA_DRV Port Signals

Signal Name	Direction	Description
CURREN	Input	Enable the mixed signal control block to supply reference current to the IR drivers. When it is not enabled (CURREN=0), no current is supplied, and the IR drivers are powered down. Enabling the mixed signal control block takes 100 µs to reach a stable reference current value.
RGBLEDEN	Input	Enable the SB_RGB_DRV primitive. Active High.
RGB0PWM	Input	Input data to drive RGB0 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB1PWM	Input	Input data to drive RGB1 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB2PWM	Input	Input data to drive RGB2 LED pin. This input is usually driven from the SB_LEDD_IP.
RGB0	Output	RGB0 LED output.
RGB1	Output	RGB1 LED output.
RGB2	Output	RGB2 LED output.

Default Signal Values

The iCEcube2 software assigns the following signal value to unconnected port:

Input CURREN	: Logic “0”
Input RGBLEDEN	: Logic “0”
Input RGB0PWM	: Logic “0”
Input RGB1PWM	: Logic “0”
Input RGB2PWM	: Logic “0”

Parameters

The SB_RGBA_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0 ";

Parameter values:

"0b0" = Full Current Mode (Default).

"0b1" = Half Current Mode.

Parameter RGB0_CURRENT = "0b000000";

Parameter RGB1_CURRENT = "0b000000";

Parameter RGB2_CURRENT = "0b000000";

Parameter values:

"0b000000" = 0 mA. // Set this value to use the associated SB_IO_OD instance at RGB // LED location.

"0b000001" = 4 mA for Full Mode; 2 mA for Half Mode

"0b000011" = 8 mA for Full Mode; 4 mA for Half Mode

"0b000111" = 12 mA for Full Mode; 6 mA for Half Mode.

"0b001111" = 16 mA for Full Mode; 8 mA for Half Mode

"0b011111" = 20 mA for Full Mode; 10 mA for Half Mode.

"0b111111" = 24 mA for Full Mode; 12 mA for Half Mode.

Verilog Instantiation

```
SB_RGBA_DRV RGBA_DRIVER (
       .CURREN(ENABLE_CURR),
       .RGBLEDEN(ENABLE_RGBDRV),
       .RGB0PWM(RGB0),
       .RGB1PWM(RGB1),
       .RGB2PWM(RGB2),
       .RGB0(LED0),
       .RGB1(LED1),
       .RGB2(LED2)
);
defparam RGBA_DRIVER.CURRENT_MODE = "0b0";
defparam RGBA_DRIVER.RGB0_CURRENT = "0b111111";
defparam RGBA_DRIVER.RGB1_CURRENT = "0b111111" ;
defparam RGBA_DRIVER.RGB2_CURRENT = "0b111111";
```

7.4.4. SB_LEDDA_IP

SB_LEDDA_IP primitive generates the RGB PWM outputs for the RGB LED drivers. The IP contains registers that are programmed in by the SCI bus interface signals.

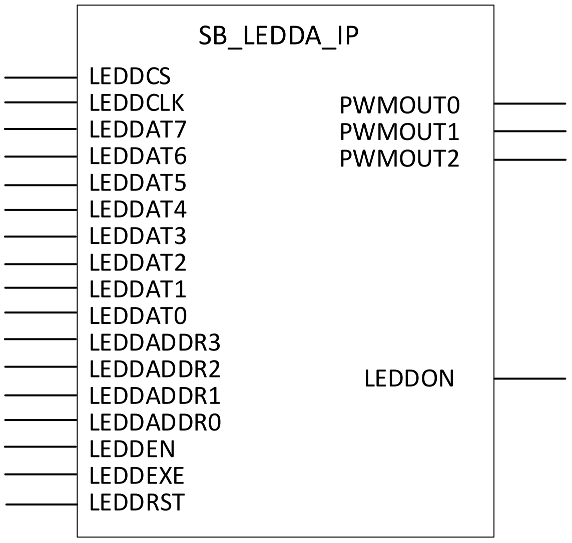

Figure 7.29. SB_LEDDA_IP

Table 7.42. SB_LEDDA_IP Port Signals

Signal Name	Direction	Description
LEDDCS	Input	CS to write LEDD IP registers
LEDDCLK	Input	Clock to write LEDD IP registers
LEDDDAT7	Input	Bit 7 data to write into the LEDD IP registers
LEDDDAT6	Input	Bit 6 data to write into the LEDD IP registers
LEDDDAT5	Input	Bit 5 data to write into the LEDD IP registers
LEDDDAT4	Input	Bit 4 data to write into the LEDD IP registers
LEDDDAT3	Input	Bit 3 data to write into the LEDD IP registers
LEDDDAT2	Input	Bit 2 data to write into the LEDD IP registers
LEDDDAT1	Input	Bit 1 data to write into the LEDD IP registers
LEDDDAT0	Input	Bit 0 data to write into the LEDD IP registers
LEDDADDR3	Input	LEDD IP register address bit 3
LEDDADDR2	Input	LEDD IP register address bit 2
LEDDADDR1	Input	LEDD IP register address bit 1
LEDDADDR0	Input	LEDD IP register address bit 0
LEDDDEN	Input	Data enable input to indicate data and address are stable.
LEDDEXE	Input	Enable the IP to run the blinking sequence. When it is LOW, the sequence stops at the nearest OFF state.
LEDDRST	Input	Device level reset signal to reset all internal registers during the device configuration. This port is not accessible to user signals.
PWMOUT0	Output	PWM output 0
PWMOUT1	Output	PWM output 1
PWMOUT2	Output	PWM output 2
EDDON	Output	Indicating the LED is on

Default Signal Values

The iCEcube2 software assigns the logic “0” value to all unconnected input ports.

Verilog Instantiation

```
SB_LEDDA_IP  PWMgen_inst (
       .LEDDCS(ledd_cs),
       .LEDDCLK(led_clk),
       .LEDDDAT7(led_ip_data[7]),
       .LEDDDAT6(led_ip_data[6]),
       .LEDDDAT5(led_ip_data[5]),
       .LEDDDAT4(led_ip_data[4]),
       .LEDDDAT3(led_ip_data[3]),
       .LEDDDAT2(led_ip_data[2]),
       .LEDDDAT1(led_ip_data[1]),
       .LEDDDAT0(led_ip_data[0]),
       .LEDDADDR3(led_ip_addr[3]),
       .LEDDADDR2(led_ip_addr[2]),
       .LEDDADDR1(led_ip_addr[1]),
       .LEDDADDR0(led_ip_addr[0]),
       .LEDDDEN(led_ip_den),
       .LEDDEXE(led_ip_exe),
       .LEDDRST(led_ip_rst),
       .PWMOUT0(LED0),
       .PWMOUT1(LED1),
       .PWMOUT2(LED2),
       .LEDDON(led_on)
);
```

7.4.5. SB_IO_OD

The SB_IO_OD is the open drain I/O primitive. When the tri-state output is enabled, the I/O pulls down the package pin signal to zero. The following figure and Verilog template illustrate the complete user accessible logic diagram, and its Verilog instantiation.

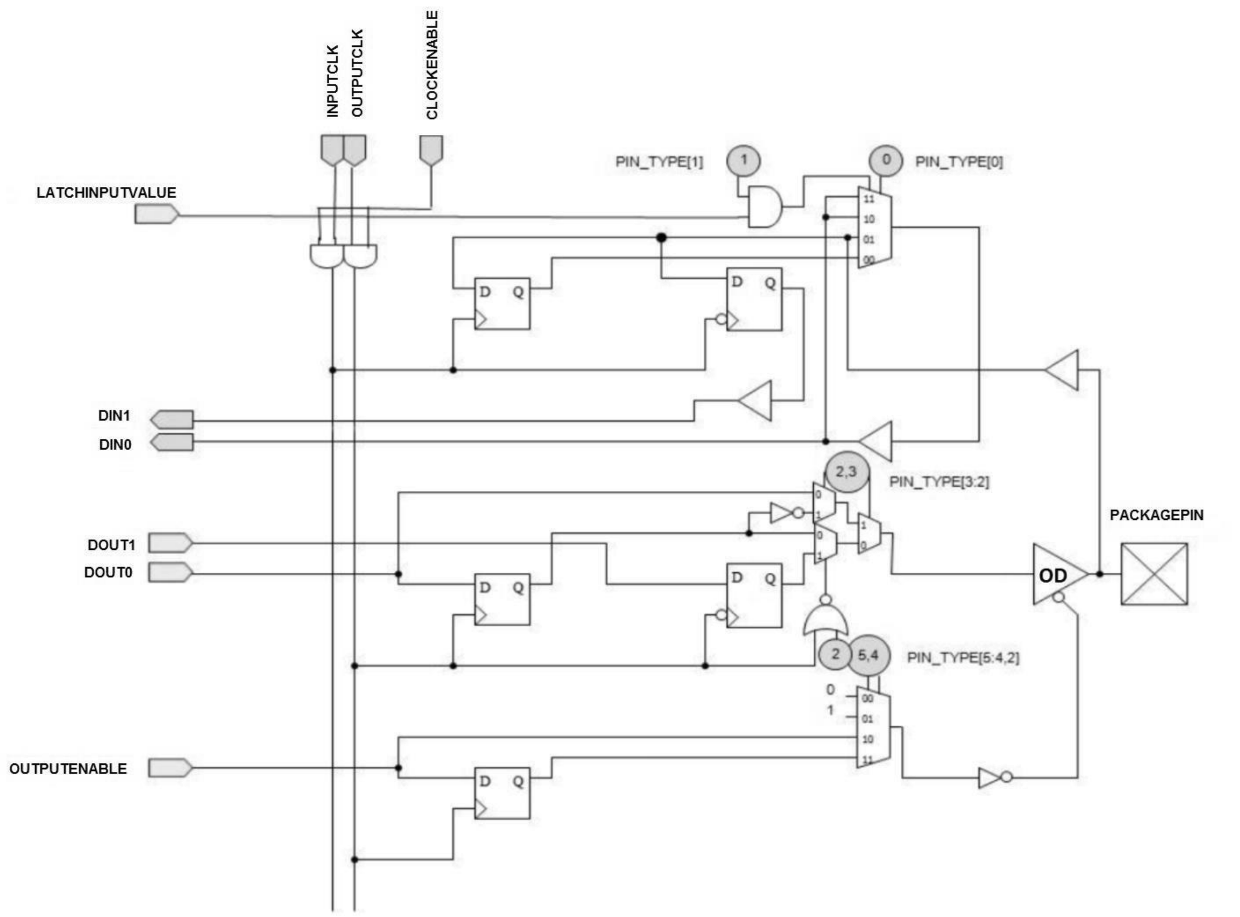

Figure 7.30. SB_IO_OD

Table 7.43. SB_IO_OD Port Signals

Signal Name	Direction	Description
PACKAGEPIN	Bidirectional	Bidirectional I/O pin.
LATCHINPUTVALUE	Input	Latches/Holds the input data.
CLOCKENABLE	Input	Clock enable signal.
INPUTCLK	Input	Clock for the input registers.
OUTPUTCLK	Input	Clock for the output registers.
OUTPUTENABLE	Input	Enable Tristate output. Active high.
DOUT0	Input	Data to package pin.
DOUT1	Input	Data to package pin.
DIN0	Output	Data from package pin.
DIN1	Output	Data from package pin.

Default Signal Values

The iCEcube2 software assigns the logic “0” value to all unconnected input ports except for CLOCKENABLE.

Note that explicitly connecting logic “1” value to port CLOCKENABLE results in a non-optimal implementation, since extra LUT is used to generate the Logic “1”. If your intention is to keep the Input and Output registers always enabled, it is recommended that port CLOCKENABLE to be left unconnected.

Parameter Values

Parameter PIN_TYPE = 6’b000000;

// See Input and Output Pin Function Tables in SB_IO.

// Default value of PIN_TYPE = 6’b000000

Parameter NEG_TRIGGER = 1’b0;

// Specify the polarity of all FFs in the I/O to be falling edge when

NEG_TRIGGER = 1.   Default is 1’b0, rising edge.

Input and Output Pin Function Tables

Refer SB_IO Input and Output Pin Functional Table for the PIN_TYPE settings. Some of the output pin configurations are not applicable for SB_IO_OD primitive.

Verilog Instantiation

SB_IO_OD   OpenDrainInst0

```
(
PACKAGEPIN (PackagePin),                //User’s Pin signal name
LATCHINPUTVALUE (latchinputvalue),            //Latches/holds the Input value
CLOCKENABLE (clockenable),              //Clock Enable common to input and
                                              //output clock
INPUTCLK (inputclk),                          //Clock for the input registers
OUTPUTCLK (outputclk),                        //Clock for the output registers
OUTPUTENABLE (outputenable),                  //Output Pin Tristate/Enable
                                              //control
DOUT0 (dout0),                                //Data 0 – out to Pin/Rising clk
                                              //edge
DOUT1 (dout1),                                //Data 1 - out to Pin/Falling clk
                                              //edge
DIN0 (din0),                            // Data 0 - Pin input/Rising clk
                                              // edge
DIN1 (din1)                                   // Data 1 – Pin input/Falling clk
                                              // edge
);

defparam OpenDrainInst0.PIN_TYPE = 6'b000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
```

7.4.5.1. SB_I2C

The iCE40 UltraPlus device supports two I2C hard IP primitives, located at upper left corner and upper right corner of the chip.

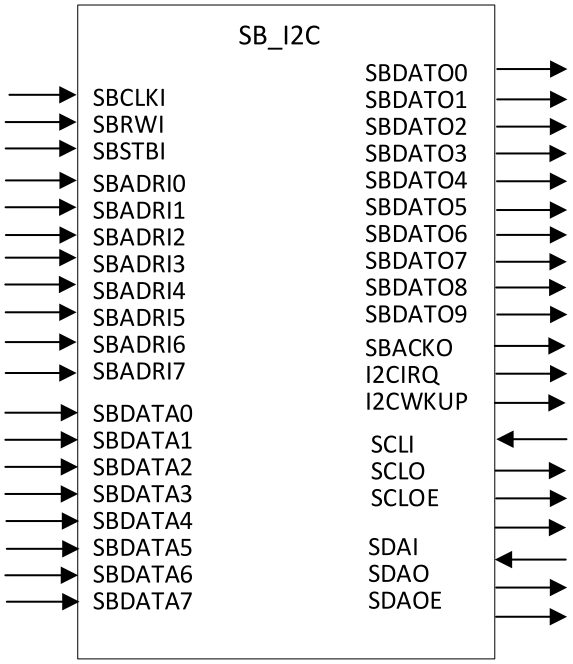

Figure 7.31. SB_I2C

Table 7.44. SB_I2C Port Signals

Signal Name	Direction	Description
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Output	System Data Output. Bit 0.
SBDATO1	Output	System Data Output. Bit 1.
SBDATO2	Output	System Data Output. Bit 2.

Signal Name	Direction	Description
SBDATO3	Output	System Data Output. Bit 3.
SBDATO4	Output	System Data Output. Bit 4.
SBDATO5	Output	System Data Output. Bit 5.
SBDATO6	Output	System Data Output. Bit 6.
SBDATO7	Output	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement.
I2CIRQ	Output	I2C Interrupt output.
I2CWKUP	Output	I2C Wake Up from Standby signal.
SCLI	Input	Serial Clock Input.
SCLO	Output	Serial Clock Output
SCLOE	Output	Serial Clock Output Enable. Active High.
SDAI	Input	Serial Data Input
SDAO	Output	Serial Data Output
SDAOE	Output	Serial Data Output Enable. Active High.

Parameters

Table 7.45. SB_I2C Parameters

I2C Location	Parameter	Parameter Default Value	Description
Upper Left Corner	I2C_SLAVE_INIT_ADDR	0b1111100001	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
	BUS_ADDR74	0b0001	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.
Upper Right Corner	I2C_SLAVE_INIT_ADDR	0b1111100010	Upper Bits <9:2> can be changed through control registers. Lower bits <1:0> are fixed.
	BUS_ADDR74	0b0011	Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.

Synthesis Attribute

I2C_CLK_DIVIDER

Synthesis attribute "I2C_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCLO output with respect to the SBCLKI input clock frequency.

/* synthesis I2C_CLK_DIVIDER= Divide Value */ Divide Value: 0, 1, 2, 3 … 1023. Default is 0.

SDA_INPUT_DELAYED

SDA_INPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAI signal.

/* synthesis SDA_INPUT_DELAYED= value */

Value:

0: No delay.

1: Add 50 ns delay. (Default value).

SDA_OUTPUT_DELAYED

SDA_OUTPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAO signal.

/* synthesis SDA_OUTPUT_DELAYED= value */

Value:

0: No delay (Default value).

1: Add 50 ns delay.

SCL_INPUT_FILTERED

SCL_INPUT_FILTERED attribute is used to add 50 ns glitch filter to the SCLI signal.

/* synthesis SCL_INPUT_FILTERED= value */

Value:

0: No Filter delay (Default value).

1: Add 50 ns glitch filter delay.

7.4.6. SB_SPI

The iCE40 UltraPlus device supports two SPI hard IP primitives, located at lower left corner and lower right corner of the chip.

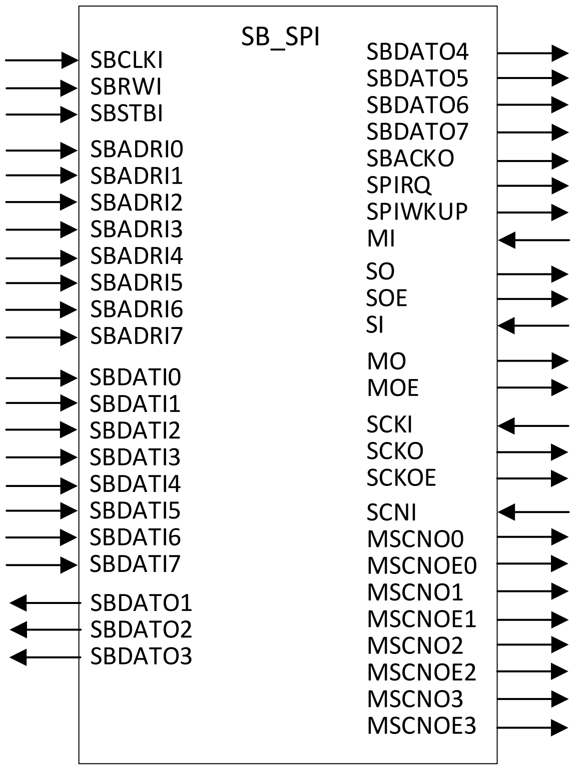

Figure 7.32. SB_SPI

Table 7.46. SB_SPI Port Signals

Signal Name	Direction	Description
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Input	System Data Output. Bit 0.
SBDATO1	Input	System Data Output. Bit 1.
SBDATO2	Input	System Data Output. Bit 2.
SBDATO3	Input	System Data Output. Bit 3.
SBDATO4	Input	System Data Output. Bit 4.
SBDATO5	Input	System Data Output. Bit 5.
SBDATO6	Input	System Data Output. Bit 6.
SBDATO7	Input	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement
SPIIRQ	Output	SPI Interrupt output.
SPIWKUP	Output	SPI Wake Up from Standby signal.
MI	Input	Controller Input from PAD
SO	Output	Target Output to PAD
SOE	Output	Target Output Enable to PAD. Active High.
SI	Input	Target Input from PAD
MO	Output	Controller Output to PAD
MOE	Output	Controller Output Enable to PAD. Active High.
SCKO	Output	Target Clock Output to PAD
SCKOE	Output	Target Clock Output Enable to PAD. Active High.
SCSNI	Input	Target Chip Select Input From PAD
MCSNO0	Output	Controller Chip Select Output to PAD. Line 0.
MCSNO1	Output	Controller Chip Select Output to PAD. Line 1.
MCSNO2	Output	Controller Chip Select Output to PAD. Line 2.
MCSNO3	Output	Controller Chip Select Output to PAD. Line 3.
MCSNOE0	Output	Controller Chip Select Output Enable to PAD. Active High. Line 0.
MCSNOE1	Output	Controller Chip Select Output Enable to PAD. Active High. Line 1.
MCSNOE2	Output	Controller Chip Select Output Enable to PAD. Active High. Line 2.
MCSNOE3	Output	Controller Chip Select Output Enable to PAD. Active High. Line 3.

Parameters

SPI Primitive requires configuring a parameter for selecting the SPI IP location.

Table 7.47. SB_SPI Parameters

SPI Location	Parameter	Parameter Default Value	Description
Lower Left Corner	BUS_ADDR74	0b0000	Fixed value. SBADRI[7:4] bits also should match with this value to activate the IP.
Lower Right Corner	BUS_ADDR74	0b0010	Fixed value. SBADRI[7:4] bits also should match with this value to activate the IP.

Synthesis Attribute

Synthesis attribute "SPI_CLK_DIVIDER" is used by PNR and STA tools for optimization and deriving the appropriate clock frequency at SCKO output with respect to the SBCLKI input clock frequency.

/* synthesis SPI_CLK_DIVIDER= [Divide Range] */ Divide Range : 0, 1, 2, 3….63. Default is 0.

7.4.7. SB_MAC16

The SB_MAC16 primitive is the dedicated configurable DSP block available in iCE5LP/iCE40 UltraPlus devices. The SB_MAC16 can be configured into a multiplier, adder, subtracter, accumulator, multiply-add and multiply-sub through the instance parameters. The SB_MAC16 blocks can be cascaded to implement wider functional units.

iCEcube2 supports a set of predefined SB_MAC16 functional configurations. Refer to SB_MAC16 section for the SB_MAC16 functional model, port details, and the list of supported configurations.

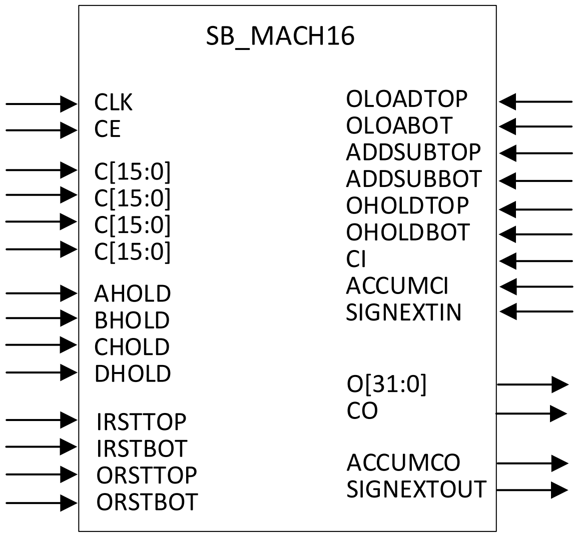

Figure 7.33. SB_MAC16

Port B (Generated Clock)

7.4.8. SB_SPRAM256KA

The iCE40 UltraPlus devices offer four embedded memory blocks of Single Port RAM. Each of these blocks can be configured in 16 kb x 16 mode. These RAM blocks can be cascaded using logic implemented in the fabric to create larger memories.

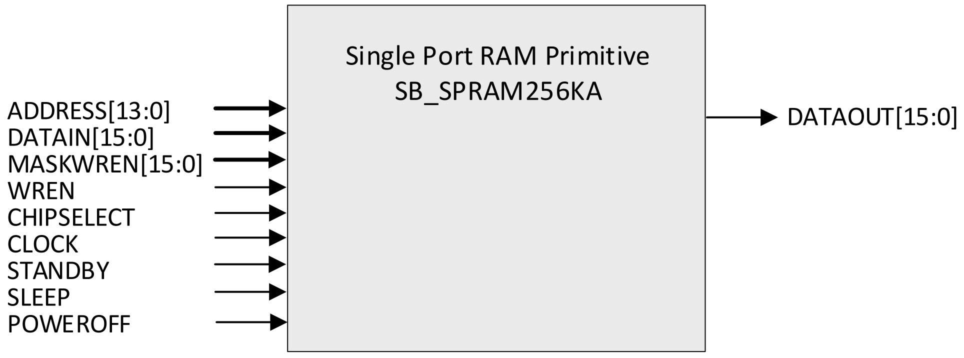

Figure 7.34. SB_SPRAM256KA

Table 7.48. SB_SPRAM256KA Ports

Port Name	Port Width	Default Value	Description
ADDRESS	[13:0]	14b’000000 00000000	This Address input port is used to address the location to be written during the write cycle and read during the read cycle.
DATAIN	[15:0]	16b’000000 0000000000	The data input bus used to write the data into the memory location specified by address input port during the write cycle.
MASKWREN	[3:0]	4b’1111	The Bit Write feature where selective write to individual I/O can be done using the Maskable Write Enable signals. Each bit masks a nibble of DATAIN.
WREN	[0:0]	1b’0	When the Write Enable input is Logic High, the memory is in the write cycle. When the Write enable is Logic Low, the memory is in the Read Cycle.
CHIPSELECT	[0:0]	1b’0	When the memory enable input is Logic High, the memory is enabled and read/write operations can be performed. When memory Enable input is logic Low, the memory is deactivated.
CLOCK	[0:0]	-	This is the external clock for the memory.
STANDBY	[0:0]	1b’0	When this pin is active then memory goes into low leakage mode, there is no change in the output state.
SLEEP	[0:0]	1b’0	This pin shut down power to periphery and maintain memory contents. The outputs of the memory are pulled low.
POWEROFF	[0:0]	1b’1	This pin turns off the power to the memory core when it is driven low (1’b0). There is no memory data retention when it is driven low. When POWEROFF is driven high (1’b1), the SPRAM is powered on.
DATAOUT	[15:0]	-	It outputs the contents of the memory location addressed by the Address Input signals.

The default value of each MASKWREN is 1. In order to mask a nibble of DATAIN, the MASKWREN needs to be pulled low (0).

The following shows which MASKWREN bits enables the mask for the DATAIN nibbles.

- MASKWREN(3) enables mask for DATAIN(15:12)
- MASKWREN(2) enables mask for DATAIN (11:8)
- MASKWREN(1) enables mask for DATAIN (7:4)
- MASKWREN(0) enables mask for DATAIN(3:0)

7.4.9. SB_IO_I3C

The iCE40 UltraPlus devices contain two special purpose I/O blocks. These blocks are same as SB_IO, but have two additional user control signals PU_ENB, WEAK_PU_ENB to enable/disable the resistor networks. These blocks are useful in I3C applications.

The SB_IO_I3C block contains five registers. The following figure and Verilog template illustrate the complete user accessible logic diagram, and its Verilog instantiation.

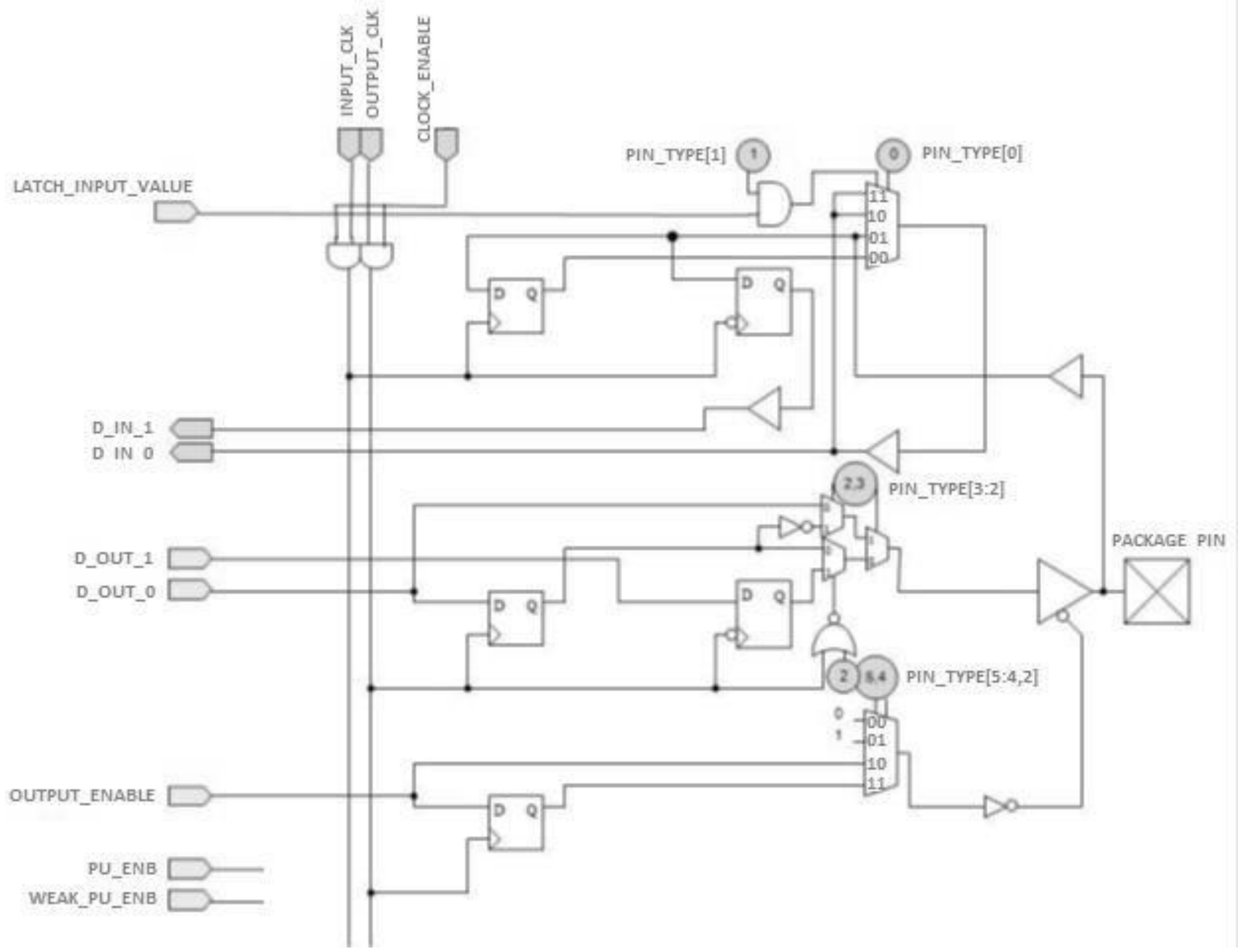

Figure 7.35. SB_IO_I3C

Default Signal Values

The iCEcube2 software assigns the logic ‘0’ value to all unconnected input ports except for CLOCK_ENABLE.

Note that explicitly connecting a logic ‘1’ value to port CLOCK_ENABLE results in a non-optimal implementation, since an extra LUT is used to generate the Logic ‘1’. If your intention is to keep the Input and Output registers always enabled, it is recommended that port CLOCK_ENABLE be left unconnected.

Explicitly connecting a logic “0” value to port PU_ENB and/or WEAK_PU_ENB results and extra LUTs are used to generate the Logic “0”. If it is intended to always enable the pull-up and/or weak pull-up, it is recommended the associated port be left unconnected.

Input and Output Pin Function Tables

Input and Output functions are independently selectable through the PIN_TYPE [1:0] and PIN_TYPE [5:2] parameter settings respectively. Specific I/O functions are defined by the combination of both attributes. This means that the complete number of combinations is 64, although some combinations are not valid and not defined below.

Note that the selection of I/O Standards such as LVCMOS is not defined by these tables.

Table 7.49. Input Pin Function

#	Pin Function Mnemonic	PIN_TYPE[1:0]		Functional Description of Package Pin Input Operation
1	PIN_INPUT	0	1	Simple input pin (D_IN_0)
2	PIN_INPUT_LATCH	1	1	Disables internal data changes on the physical input pin by latching the value.
3	PIN_INPUT_REGISTERED	0	0	Input data is registered in input cell
4	PIN_INPUT_REGISTERED_LATCH	1	0	Disables internal data changes on the physical input pin by latching the value on the input register
5	PIN_INPUT_DDR	0	0	Input 'DDR' data is clocked out on rising and falling clock edges. Use the D_IN_0 and D_IN_1 pins for DDR operation.

Table 7.50. Output Pin Function

#	Pin Function Mnemonic	PIN_TYPE[5:2]				Functional Description of Package Pin Output Operation
1	PIN_NO_OUTPUT	0	0	0	0	Disables the output function
2	PIN_OUTPUT	0	1	1	0	Simple output pin, (no enable)
3	PIN_OUTPUT_TRISTATE	1	0	1	0	The output pin may be tri-stated using the enable
4	PIN_OUTPUT_ENABLE_REGISTERED	1	1	1	0	The output pin may be tri-stated using a registered enable signal
5	PIN_OUTPUT_REGISTERED	0	1	0	1	Output registered, (no enable)
6	PIN_OUTPUT_REGISTERED_ENABLE	1	0	0	1	Output registered with enable (enable is not registered)
7	PIN_OUTPUT_REGISTERED_ENABLE _REGISTERED	1	1	0	1	Output registered and enable registered
8	PIN_OUTPUT_DDR	0	1	0	0	Output 'DDR' data is clocked out on rising and falling clock edges
9	PIN_OUTPUT_DDR_ENABLE	1	0	0	0	Output data is clocked out on rising and falling clock edges
10	PIN_OUTPUT_DDR_ENABLE_REGIST ERED	1	1	0	0	Output 'DDR' data with registered enable signal
11	PIN_OUTPUT_REGISTERED_INVERT ED	0	1	1	1	Output registered signal is inverted
12	PIN_OUTPUT_REGISTERED_ENABLE INVERTED	1	0	1	1	Output signal is registered and inverted, (no enable function)
13	PIN_OUTPUT_REGISTERED_ENABLE _REGISTERED_INVERTED	1	1	1	1	Output signal is registered and inverted, the enable/tristate control is registered.

Syntax Verilog Use

Output Pin Function is the bit vector associated with PIN_TYPE [5:2] and Input Pin Function is the bit vector associated with PIN_TYPE [1:0], resulting in a 6-bit value PIN_TYPE [5:0]

defparam my_generic_IO.PIN_TYPE = 6’b{Output Pin Function, Input Pin Function};

Pull-Up Resistor Configuration.

You can configure the internal pull-up resistor strength of an I/O to a predefined resistor value through the parameter settings and synthesis attributes. The PULLUP parameter setting enables one of the resistor values as given in table 1 and WEAKPULLUP parameter setting enables the 100k pull-up resistor.

The specific resistor value associated with the PULLUP parameter needs to specify through the PULLUP_RESISTOR attribute. The PULLUP_RESISTOR value is effective only when PULLUP parameter is set to 1.

Both the PULLUP and WEAKPULLUP parameters are allowed to set simultaneously. When both parameters are set the effective pull-up resistor values is [100K || {3P3K/6P8K/10K}].

Once the required parameters, attribute values are set, the active low PU_ENB, WEAK_PU_ENB signals allows you to control the pull-up resistor network dynamically.

Synthesis Attribute Syntax:

/* synthesis PULLUP_RESISTOR = "3P3K" */

Resistor Value

Table 7.51. Resistor Value

Resistor Value	Description
"3P3K"	Pull-up resistor level is 3.3 kΩ.
"6P8K"	Pull-up resistor level is 6.8 kΩ.
"10K"	Pull-up resistor level is 10 kΩ.

Note: The PULLUP_RESISTOR value is effective only when PULLUP parameter is set to 1.

Verilog Instantiation

```
SB_IO_I3C    I3CIO_INST
(
.PACKAGE_PIN (Package_Pin),                    //User’s Pin signal name
.LATCH_INPUT_VALUE (latch_input_value),//Latches/holds the Input value
.CLOCK_ENABLE (clock_enable),                  //Clock Enable common to input and
                                               //output clock
.INPUT_CLK (input_clk),                        //Clock for the input registers
.OUTPUT_CLK (output_clk),               //Clock for the output registers
.OUTPUT_ENABLE (output_enable),         //Output Pin Tristate/Enable
                                               //control
.D_OUT_0 (d_out_0),                     //Data 0 – out to Pin/Rising clk
                                               //edge
.D_OUT_1 (d_out_1),                     //Data 1 - out to Pin/Falling clk
                                               //edge
.D_IN_0 (d_in_0),                              //Data 0 - Pin input/Rising clk
                                               //edge
.D_IN_1 (d_in_1),                              //Data 1 – Pin input/Falling clk
                                               //edge
.PU_ENB (pu_enb),                              // Active low Pull Up resistor
Enable –Default to GND

.WEAK_PU_ENB (weak_pu_enb)              //Active low Weak Pull Up Enable
 for 100k resistor. Default to GND

) /* synthesis PULLUP_RESISTOR "3P3K" */  ;

defparam I3CIO_INST.PIN_TYPE = 6'b000000;
                                    // See Input and Output Pin Function Tables.
                                    // Default value of PIN_TYPE = 6’000000 i.e.
                                    // an input pad, with the input signal
                                    // registered.
defparam I3CIO_INST.PULLUP = 1'b0;
                                    // By default, the IO will have NO pull up.
defparam I3CIO_INST.WEAK_PULLUP = 1'b0;
                                   // By default, the IO will have NO Weak pull up.
defparam I3CIO_INST.NEG_TRIGGER = 1'b0;
defparam I3CIO_INST.IO_STANDARD = "SB_LVCMOS";
```

7.5. Device Configuration Primitives

7.5.1. SB_WARMBOOT

The iCE FPGA devices permit you to load a different configuration image during regular operation. Through the use of the Warm Boot Primitive, you can load one of four pre-defined configuration images into the iCE FPGA device.

Note that this Warm Boot mode is different from the Cold Boot operation, which is executed during the initial device boot-up sequence.

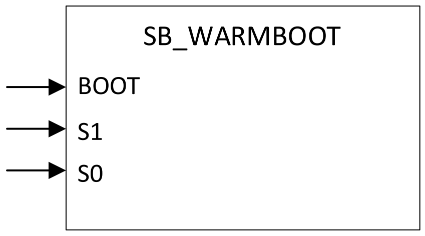

Figure 7.36. SB_WARMBOOT

The selection of one of these four images is accomplished through two input signals, S1 and S0. In order to trigger the selection of a new image, an additional signal, BOOT, is provided. It should be noted that this signal is level-triggered, and should be used for every Warm Boot operation. For example, every time you want to load a new image into the device.

The successful instantiation of this primitive also requires the user to specify the address locations of the four images. These addresses should be specified in the iCEcube2 software as per the Warm Boot Application Note.

Verilog Instantiation

```
SB_WARMBOOT  my_warmboot_i(
       .BOOT (my_boot),          // Level-sensitive trigger signal
       .S1 (my_sel1),            // S1, S0 specify selection of the
                                 // configuration image
       .S0 (my_sel0)
       );
```

References

- iCE40 UltraPlus web page
- iCE40 Ultra/Ultra Lite web page
- iCE40 LP/HX web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 3.3, September 2024

Section	Change Summary
All	• Minor editorial fixes. • Changed the word *Master* to *Controller*. • Changed the word *Slave* to *Target*.
Disclaimers	Updated this section.
Inclusive Language	Added this section.
Abbreviations in This Document	Changed *Acronyms* to *Abbreviations*.
I/O Primitives	Added the VHDL code under SB_IO subsection.
Global Buffer Primitives	Added the VDHL code under SB_GB_IO and SB_GB Primitive subsection.
Hard Macro Primitives	Added VHDL code under SB_IO_OD subsection.
References	Added this section.
Technical Support Assistance	Added reference to the Lattice Answer Database on the Lattice website.

Revision 3.2, November 2021

Section	Change Summary
All	Adjustments in formatting across the document.
Acronyms in This Document	Added this section.
I/O Primitives	Changed table name to Pull-up Resistor Value and K to k unit in Table 4.4.
Hard Macro Primitives	• Changed 16K x 16 mode to 16 kb x 16 mode in SB_SPRAM256KA.. • Updated Table 7.6 and table 7.47 to change I2C column to SPI Location and parameter value of Lower Right Corner to 0b0010. • Changed K to k unit in Table 7.51.

Revision 3.1, July 2021

Section	Change Summary
All	• Assigned document number FPGA-TN-02026. • Updated document template.
Disclaimers	Added this section.
Hard Macro Primitives	Updated Figure 7.15.

Revision 3.0, August 2016

Section	Change Summary
All	Previous Lattice release.

www.latticesemi.com