LATTICE ICE™ Technology Library

Version 3.0
August, 2016
Copyright
Copyright © 2007-2016 Lattice Semiconductor Corporation. All rights reserved. This
document may not, in whole or part, be reproduced, modified, distributed, or publicly
displayed without prior written consent from Lattice Semiconductor Corporation
(“Lattice”).

Trademarks
All Lattice trademarks are as listed at www.latticesemi.com/legal. Synopsys and
Synplify Pro are trademarks of Synopsys, Inc. Aldec and Active-HDL are trademarks of
Aldec, Inc. All other trademarks are the property of their respective owners.

Disclaimers
NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS “AS IS”
WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING
WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY,
NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY
PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE OR ITS SUPPLIERS BE
LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT,
SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT
LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR
LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE
THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LATTICE HAS BEEN
ADvised OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME
JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN
LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Lattice may make changes to these materials, specifications, or information, or to the
products described herein, at any time without notice. Lattice makes no commitment to
update this documentation. Lattice reserves the right to discontinue any product or
service without notice and assumes no obligation to correct any errors contained herein
or to advise any user of this document of any correction if such be made. Lattice
recommends its customers obtain the latest version of the relevant information to
establish that the information being relied upon is current and before ordering any
products.
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>Added Version Number to document. Added sections on Default Signal Values for unconnected ports.</td>
</tr>
<tr>
<td>2.1</td>
<td>Added PLL primitives</td>
</tr>
<tr>
<td>2.2</td>
<td>Corrected SB_CARRY connections to LUT inputs</td>
</tr>
<tr>
<td>2.3</td>
<td>Added iCE40 RAM, PLL primitives.</td>
</tr>
<tr>
<td>2.4</td>
<td>Added PLL_DS, SB_MIPI_RX_2LANE, SB_TMDS_deserializer primitives.</td>
</tr>
<tr>
<td>2.5</td>
<td>Added SB_MAC16 Primitive details.</td>
</tr>
<tr>
<td>2.6</td>
<td>Added iCE40LM Hard Macro details. Removed PLL_DS, SB_MIPI, SB_TMDS, SB_MAC16 primitive details.</td>
</tr>
<tr>
<td>2.7</td>
<td>Added iCE5LP (iCE40 Ultra) primitive details.</td>
</tr>
<tr>
<td>2.8</td>
<td>Removed iCE65 RAM, PLL details.</td>
</tr>
<tr>
<td>2.9</td>
<td>Added iCE40UL (iCE40 Ultra Lite) primitive details</td>
</tr>
<tr>
<td>3.0</td>
<td>Added iCE40UP (iCE40 Ultra Plus) primitives.</td>
</tr>
</tbody>
</table>
# Table of Contents

Register Primitives

<table>
<thead>
<tr>
<th>SB_DFF</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_DFFE</td>
<td>9</td>
</tr>
<tr>
<td>SB_DFFSR</td>
<td>11</td>
</tr>
<tr>
<td>SB_DFFR</td>
<td>13</td>
</tr>
<tr>
<td>SB_DFFSS</td>
<td>15</td>
</tr>
<tr>
<td>SB_DFFS</td>
<td>17</td>
</tr>
<tr>
<td>SB_DFFESR</td>
<td>19</td>
</tr>
<tr>
<td>SB_DFFER</td>
<td>21</td>
</tr>
<tr>
<td>SB_DFFESS</td>
<td>23</td>
</tr>
<tr>
<td>SB_DFFES</td>
<td>25</td>
</tr>
<tr>
<td>SB_DFFN</td>
<td>27</td>
</tr>
<tr>
<td>SB_DFFNE</td>
<td>29</td>
</tr>
<tr>
<td>SB_DFFNSR</td>
<td>31</td>
</tr>
<tr>
<td>SB_DFFNR</td>
<td>33</td>
</tr>
<tr>
<td>SB_DFFNSS</td>
<td>35</td>
</tr>
<tr>
<td>SB_DFFNS</td>
<td>37</td>
</tr>
<tr>
<td>SB_DFFNESR</td>
<td>39</td>
</tr>
<tr>
<td>SB_DFFNER</td>
<td>41</td>
</tr>
<tr>
<td>SB_DFFNESS</td>
<td>43</td>
</tr>
<tr>
<td>SB_DFFNES</td>
<td>45</td>
</tr>
</tbody>
</table>

Combinational Logic Primitives

| SB_LUT4             | 47|
| SB_CARRY            | 49|

Block RAM Primitives

iCE40 Block RAM

<p>| SB_RAM256x16        | 51|
| SB_RAM256x16NR      | 52|
| SB_RAM256x16NW      | 54|
| SB_RAM256x16NWRNW   | 55|
| SB_RAM512x8         | 57|
| SB_RAM512x8NR       | 60|
| SB_RAM512x8NW       | 62|
| SB_RAM512x8NWRNW    | 63|
| SB_RAM512x8NWR      | 65|
| SB_RAM1024x4        | 68|
| SB_RAM1024x4NR      | 70|
| SB_RAM1024x4NW      | 71|
| SB_RAM1024x4NWRNW   | 73|
| SB_RAM2048x2        | 76|
| SB_RAM2048x2NR      | 78|
| SB_RAM2048x2NW      | 79|
| SB_RAM2048x2NWRNW   | 81|</p>
<table>
<thead>
<tr>
<th>Hard Macro Primitives</th>
<th>IO Primitives</th>
<th>Global Buffer Primitives</th>
<th>PLL Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM40_4K</td>
<td>SB_IO</td>
<td>SB_GB_IO</td>
<td>SB_PLL40_2F_CORE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SB_GB_Primitive</td>
<td>SB_PLL40_2F_PAD</td>
</tr>
<tr>
<td>iCE40UP (iCE40 Ultra Plus) Hard Macros</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_PLL40_2F_PAD</td>
</tr>
<tr>
<td>iCE5LP (iCE40 Ultra) Hard Macros</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_LSOSC (For LPSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_I2C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_SPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_RGB_DRV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_UNIQUE_DRV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_UNIQUE_DRV</td>
</tr>
<tr>
<td>iCE40UltraLite Hard Macros</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HFOSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_RGB_DRV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_UNIQUE_DRV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_UNIQUE_DRV</td>
</tr>
<tr>
<td>iCE40UltraPlus Hard Macros</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HFOSC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_RGB_DRV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_UNIQUE_DRV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_HSOSC (For HSSG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SB_UNIQUE_DRV</td>
</tr>
<tr>
<td>Component</td>
<td>Page Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_LFOSC</td>
<td>163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RGBA_DRV</td>
<td>164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_LEDDA_IP</td>
<td>166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_IO_OD</td>
<td>167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_I2C</td>
<td>170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_SPI</td>
<td>171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_MAC16</td>
<td>172</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_SPRAM256KA</td>
<td>173</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_IO_I3C</td>
<td>176</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device Configuration Primitives</td>
<td>180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_WARMBOOT</td>
<td>180</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Register Primitives

SB_DFF
D Flip-Flop

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

\[
\begin{array}{ccc}
\text{D} & \text{SB_DFF} & \text{Q} \\
\text{C} & & \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\text{X} & \text{X} & 0 \\
\end{array}
\]

Key

\begin{itemize}
  \item \text{Rising Edge}
  \item 1 \text{ High logic level}
  \item 0 \text{ Low logic level}
  \item \text{X Don’t care}
  \item \text{? Unknown}
\end{itemize}

HDL use

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

Verilog Instantiation

```verilog
// SB_DFF - D Flip-Flop.
SB_DFF SB_DFF_inst (  // Registered Output
  .Q(Q),
  .C(C),  // Clock
  .D(D),  // Data
);
```

// End of SB_DFF instantiation

VHDL Instantiation

```vhdl
-- 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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

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 will result in a non-optimal implementation, since an extra LUT will be 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**

```verilog
// 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
**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.

```
D Q
--- ---
SB_DFFSR
--- ---
C R
```

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

Key

- \( \uparrow \) 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.

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

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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

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**

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

-- End of SB_DFFSS instantiation
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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

```verilog
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
**SB_DFFESR**

D Flip-Flop with Clock Enable and Synchronous 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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R    E    D    C    Q</td>
<td></td>
</tr>
<tr>
<td>---  ---  ---  ---  ---</td>
<td></td>
</tr>
<tr>
<td>1    1    X    X    0</td>
<td></td>
</tr>
<tr>
<td>X    0    X    X    Previous Q</td>
<td></td>
</tr>
<tr>
<td>0    1    0    X    0</td>
<td></td>
</tr>
<tr>
<td>0    1    1    X    1</td>
<td></td>
</tr>
<tr>
<td>Power on State</td>
<td>X  X  X  0</td>
</tr>
</tbody>
</table>

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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_DFFESR - D Flip-Flop, Reset is synchronous with rising clock edge
// Clock Enable.
SB_DFFESR SB_DFFESR_inst (  
  .Q(Q),                      // Registered Output
  .C(C),                      // Clock
  .E(E),                      // Clock Enable
  .D(D),                      // Data
  .R(R)                       // Synchronous Reset
);
// End of SB_DFFESR instantiation

VHDL Instantiation

-- SB_DFFESR - D Flip-Flop, Reset is synchronous with rising clock edge
-- Clock Enable.
SB_DFFESR_inst: SB_DFFESR  
  port map (               
    Q => Q,                -- Registered Output
    C => C,                -- Clock
    E => E,                -- Clock Enable
    D => D,                -- Data
    R => R                 -- Synchronous Reset
  );
-- End of SB_DFFESR instantiation
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 input is active high, overrides all other inputs and asynchronously resets the Q output.

```
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>
```

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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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
**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.

![Diagram of SB_DFFESS](image)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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.

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

Key

- Rising Edge
- 1 High logic level
- 0 Low logic level
- X Don’t care
- ? Unknown
// End of SB_DFFESS instantiation

**VHDL Instantiation**

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

```vhdl
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
```
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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**

```verilog
// SB_DFFES - D Flip-Flop, Set is asynchronous on rising clock edge with Clock Enable.
SB_DFFES SB_DFFES_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 asynchronous on rising clock edge with Clock Enable.

SB_DFFES_inst : SB_DFFES
  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
**SB_DFFN**

**D Flip-Flop – Negative Edge Clock**

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

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| Power on State | X | X | 0 |

**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**

```verilog
// 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
**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.

![Diagram of SB_DFFNE D Flip-Flop](image)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E</strong></td>
<td><strong>D</strong></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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**

```verilog
// 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
**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.

```
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>
```

**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 clock edge
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

```vhdl
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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

Key

- \(\uparrow\) 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
**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.

![SB_DFFNSS Diagram]

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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
**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.

### Inputs

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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**

```verilog
// 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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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**

```verilog
// 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
**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.

```

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>
```

### 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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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_DFFNESS instantiation

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
**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.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
</tr>
</tbody>
</table>

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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_DFFNES instantiation

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
Combinational Logic Primitives

**SB_LUT4**

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

```
+---+---+---+---+---+
| I0| I1| I2| I3| 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] |
```

**Initialization values**

LUT state initialization parameter LUT_INIT = 16'hxxxx;

**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  
  );

// LUT state initialization parameter, 16 bits.

defparam SB_LUT4_inst.LUT_INIT=16'hxxxx;

// 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 => 0  
    );
**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**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I0</strong></td>
<td><strong>I1</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**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 (  
 C0 => CO,  
 CI => CI,  
 I0 => I0,  
 I1 => I1  
);
Block RAM Primitives

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

ICE40 Block RAM

Each ICE40 device includes multiple high-speed synchronous RAM blocks, each 4Kbit 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 256x16, 512x8, 1024x4 or 2048x2. 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 4Kbits in size. The RAM blocks can be directly instantiated in the top module and taken through iCube2 flow.

<table>
<thead>
<tr>
<th>Block RAM Configuration</th>
<th>Block RAM Size</th>
<th>WADDR Port Size (Bits)</th>
<th>WDATA Port Size (Bits)</th>
<th>RADDR Port Size (Bits)</th>
<th>RDATA Port Size (Bits)</th>
<th>MASK Port Size (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>256x16 (4K)</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td></td>
</tr>
<tr>
<td>SB_RAM256x16NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM256x16NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM512x8</td>
<td>512x8 (4K)</td>
<td>9 [8:0]</td>
<td>8 [7:0]</td>
<td>8 [8:0]</td>
<td></td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM512x8NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM512x8NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM512x8NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM1024x4NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM1024x4NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM1024x4NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM2048x2</td>
<td>2048x2 (4K)</td>
<td>11 [10:0]</td>
<td>2 [1:0]</td>
<td>10 [9:0]</td>
<td>2 [1:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM2048x2NR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM2048x2NW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB_RAM2048x2NRNW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 256x16 RAM block configuration.

<table>
<thead>
<tr>
<th>RAM Primitive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>Posedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM4256x16NR</td>
<td>Negedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NW</td>
<td>Posedge Read clock, Negedge Write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td>Negedge Read clock, Negedge Write clock</td>
</tr>
</tbody>
</table>
The following modules are the complete list of SB_RAM256x16 based primitives

**SB_RAM256x16**

```
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_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
);

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:

```vhdl
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
);  
```

**SB_RAM256x16NW**

---

**SB_RAM256x16NW** // Negative edged Write Clock – i.e. WCLKN
(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
Verilog Instantiation:

SB_RAM256x16NW  ram256x16nw_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 ram256x16nw_inst.INIT_0 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_1 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_2 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_3 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_4 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_5 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_6 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_7 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_8 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_9 =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_A =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_B =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_C =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_D =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_E =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram256x16nw_inst.INIT_F =  
    256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

ram256x16nw_inst : SB_RAM256x16NW  
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"  
);
IN\_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_A => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_B => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_C => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_D => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_E => X"0000000000000000000000000000000000000000000000000000000000000000",
IN\_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, 
  MASK => MASK\_c,
) ;

SB\_RAM256x16NRNW

SB\_RAM256x16NRNW // Negative edged Read and Write – i.e. RCLKN WRCLKLN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB\_RAM256x16NRNW ram256x16nrnw\_inst (  
  .RD\_DATA(RDATA\_c[15:0]),  
  .RADDR(RADDR\_c[7:0]),  
  .RCLK(RCLK\_c),  
  .R\_CLKE(RCLKE\_c),  
  .RE(RE\_c),  
  .WA\_ADDR(WADDR\_c[7:0]),  
  .WA\_CLKN(WCLKN\_c),  
  .WA\_CLKE(WCLKE\_c),  
  .WA\_DATA(WDATA\_c[15:0]),  
  .WE(WE\_c),  
  .M\_ASK(MASK\_c[15:0])
) ;
defparam ram256x16nrnw\_inst.INIT\_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw\_inst.INIT\_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
```
defparam \texttt{ram256x16nrmw\_inst.INIT\_2} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_3} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_4} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_5} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_6} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_7} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_8} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_9} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_A} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_B} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_C} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_D} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_E} = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam \texttt{ram256x16nrmw\_inst.INIT\_F} = 256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

\texttt{ram256x16nrmw\_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");
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  
);
The following modules are the complete list of SB_RAM512x8 based primitives

**SB_RAM512x8**

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

Verilog Instantiation:

```verilog
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;
```

ICE Technology Library
Lattice Semiconductor Corporation Confidential
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,
WADDR => WADDR_c,
WCLK=> WCLK_c,
WCLKE => WCLKE_c,
WDATA => WDATA_c,
WE => WE_c
);

**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_RAM512x8NR ram512x8nr_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:

```vhdl
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,
    RCLK  => RCLK_c,
    RCLKE => RCLKE_c,
    RE    => RE_c,
    WADDR => WADDR_c,
    WCLK  => WCLK_c,
    WCLKE => WCLKE_c,
    WDATA => WDATA_c,
    WE    => WE_c
    );
"
)

SB_RAM512x8NW

SB_RAM512x8NW // Negative edged Write Clock – i.e. WCLKN
(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLK, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```verilog
SB_RAM512x8NW ram512x8nw_inst ("
    .RDAT(A(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"
);
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);

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 (  .RDATARDATA(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:**

```vhdl
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
);
The following modules are the complete list of SB_RAM1024x4 based primitives

**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:**

```vhdl
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
);

// 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:

```vhdl
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,
  RCLK => RCLK_c,
  RCLKE => RCLKE_c,
  RE => RE_c,
  WADDR => WADDR_c,
  WCLK => WCLK_c,
  WCLKE => WCLKE_c,
  WDATA => WDATA_c,
  WE => WE_c
);
```

**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]),
  .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:

```
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"
  )
```

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
);

**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(WCLKN_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
);
The following modules are the complete list of SB_RAM2048x2 based primitives

Verilog Instantiation:

```verilog
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"
)
     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,
SB_RAM2048x2NR

SB_RAM2048x2NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM2048x2NR ram2048x2nr_inst (
    .RDATA(RDATA_c[1:0]),
    .RADDR(RADDR_c[10:0]),
    .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:
am2048x2nr_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,    
    RCLK => RCLK_c,    
    RCLKE => RCLKE_c,    
    RE => RE_c,    
    WADDR => WADDR_c,    
    WCLK => WCLK_c,    
    WCLKE => WCLKE_c,    
    WDATA => WDATA_c,    
    WE => WE_c    
);

SB_RAM2048x2NW

SB_RAM2048x2NW // Negative edged Write Clock – i.e. WCLKN
(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM2048x2NW     ram2048x2nw_inst (    
    .RDAT(A(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"
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  
    );

\textbf{SB\_RAM2048x2NRNW}

\textbf{SB\_RAM2048x2NRNW} // Negative edged Read and Write – i.e. RCLKN WRCKLN 
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB\_RAM2048x2NRNW ram2048x2nrrnw\_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 ram2048x2nrrnw\_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2nrrnw\_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
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  
);
**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 4K bits 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.

**SB_RAM40_4K Naming Convention Rules**

<table>
<thead>
<tr>
<th>RAM Primitive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM40_4K</td>
<td>Posedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM40_4KNR</td>
<td>Negedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM40_4KNW</td>
<td>Posedge Read clock, Negedge Write clock</td>
</tr>
<tr>
<td>SB_RAM40_4KNRNW</td>
<td>Negedge Read clock, Negedge Write clock</td>
</tr>
</tbody>
</table>

The following table lists the signals for both ports.

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDATA[15:0]</td>
<td>Input</td>
<td>Write Data input</td>
</tr>
<tr>
<td>MASK[15:0]</td>
<td>Input</td>
<td>Bit-line Write Enable input, active low. Applicable only when WRITE_MODE parameter is set to 0.</td>
</tr>
<tr>
<td>WADDR[7:0]</td>
<td>Input</td>
<td>Write Address input. Selects up to 256 possible locations</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>Write Enable input, active high</td>
</tr>
<tr>
<td>WCLK</td>
<td>Input</td>
<td>Write Clock input, rising-edge active</td>
</tr>
<tr>
<td>WCLKE</td>
<td>Input</td>
<td>Write Clock Enable input</td>
</tr>
<tr>
<td>RDATA[15:0]</td>
<td>Output</td>
<td>Read Data output</td>
</tr>
<tr>
<td>RADDR[7:0]</td>
<td>Input</td>
<td>Read Address input. Selects one of 256 possible locations</td>
</tr>
<tr>
<td>RE</td>
<td>Input</td>
<td>Read Enable input, active high</td>
</tr>
<tr>
<td>RCLK</td>
<td>Input</td>
<td>Read Clock input, rising-edge active</td>
</tr>
<tr>
<td>RCLKE</td>
<td>Input</td>
<td>Read Clock Enable input</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Parameter Value</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>INIT_0, ..., INIT_F</td>
<td>RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16x256=4096 total bits)</td>
<td>INIT_0 to INIT_F</td>
</tr>
<tr>
<td>WRITE_MODE</td>
<td>Sets the RAM block write port configuration</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>READ_MODE</td>
<td>Sets the RAM block read port configuration</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

**SB_RAM40_4K**

Verilog Instantiation:

```verbatim
// 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

```verbatim
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
);
```

---

ICE Technology Library
Lattice Semiconductor Corporation Confidential
SB_RAM40_4KNR

Verilog Instantiation:

// Physical RAM Instance without Pre Initialization

SB_RAM40_4KNR ram40_4knrst_physical (  
    .RDATA(RDATA),  
    .RADDR(RADDR),  
    .WADDR(WADDR),  
    .MASK(MASK),  
    .RCLKE(RCLKE),  
    .RCLKN(RCLKN),  
    .RE(RE),  
    .WCLKE(WCLKE),  
    .WCLK(WCLK),  
    .WE(WE)  
);

defparam ram40_4knrst_physical.READ_MODE=0;
defparam ram40_4knrst_physical.WRITE_MODE=0;

VHDL Instantiation:

-- Physical RAM Instance without Pre Initialization

ram40_4knrst_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  
);
**SB_RAM40_4KNW**

Verilog Instantiation:

```verilog
// 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:

```vhdl
-- 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  
);
```
**SB_RAM40_4KNRNW**

Verilog Instantiation:

```verilog
// 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:

```vhdl
-- 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  
);
```
IO Primitives

**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.

![Logic Diagram](image)

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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 via PIN_TYPE [1:0] and PIN_TYPE [5:2] parameter settings respectively. Specific IO 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 IO Standards such as SSTL and LVCMOS are not defined by these tables.

### Input Pin Function Table

<table>
<thead>
<tr>
<th>#</th>
<th>Pin Function Mnemonic</th>
<th>PIN_TYPE[1:0]</th>
<th>Functional Description of Package Pin Input Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIN_INPUT</td>
<td>0 1</td>
<td>Simple input pin (D_IN_0)</td>
</tr>
<tr>
<td>2</td>
<td>PIN_INPUT_LATCH</td>
<td>1 1</td>
<td>Disables internal data changes on the physical input pin by latching the value.</td>
</tr>
<tr>
<td>3</td>
<td>PIN_INPUT_REGISTERED</td>
<td>0 0</td>
<td>Input data is registered in input cell</td>
</tr>
<tr>
<td>4</td>
<td>PIN_INPUT_REGISTERED_LATCH</td>
<td>1 0</td>
<td>Disables internal data changes on the physical input pin by latching the value on the input register</td>
</tr>
<tr>
<td>5</td>
<td>PIN_INPUT_DDR</td>
<td>0 0</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Output Pin Function Table

<table>
<thead>
<tr>
<th>#</th>
<th>Pin Function Mnemonic</th>
<th>PIN_TYPE[5:2]</th>
<th>Functional Description of Package Pin Output Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIN_NO_OUTPUT</td>
<td>0 0 0 0</td>
<td>Disables the output function</td>
</tr>
<tr>
<td>2</td>
<td>PIN_OUTPUT</td>
<td>0 1 1 0</td>
<td>Simple output pin, (no enable)</td>
</tr>
<tr>
<td>3</td>
<td>PIN_OUTPUT_TRISTATE</td>
<td>1 0 1 0</td>
<td>The output pin may be tristated using the enable</td>
</tr>
<tr>
<td>4</td>
<td>PIN_OUTPUT_ENABLE_REGISTERED</td>
<td>1 1 1 0</td>
<td>The output pin may be tristated using a registered enable signal</td>
</tr>
<tr>
<td>5</td>
<td>PIN_OUTPUT_REGISTERED</td>
<td>0 1 0 1</td>
<td>Output registered, (no enable)</td>
</tr>
<tr>
<td>6</td>
<td>PIN_OUTPUT_REGISTERED_ENABLE</td>
<td>1 0 0 1</td>
<td>Output registered with enable (enable is not registered)</td>
</tr>
<tr>
<td>7</td>
<td>PIN_OUTPUT_REGISTERED_ENABLE_REGISTERED</td>
<td>1 1 0 1</td>
<td>Output registered and enable registered</td>
</tr>
<tr>
<td>8</td>
<td>PIN_OUTPUT_DDR</td>
<td>0 1 0 0</td>
<td>Output 'DDR' data is clocked out on rising and falling clock edges</td>
</tr>
<tr>
<td>9</td>
<td>PIN_OUTPUT_DDR_ENABLE</td>
<td>1 0 0 0</td>
<td>Output data is clocked out on rising and falling clock edges</td>
</tr>
<tr>
<td>10</td>
<td>PIN_OUTPUT_DDR_ENABLE_REGISTERED</td>
<td>1 1 0 0</td>
<td>Output 'DDR' data with registered enable signal</td>
</tr>
<tr>
<td>11</td>
<td>PIN_OUTPUT_REGISTERED_INVERTED</td>
<td>0 1 1 1</td>
<td>Output registered signal is inverted</td>
</tr>
<tr>
<td>12</td>
<td>PIN_OUTPUT_REGISTERED_ENABLE_INVERTED</td>
<td>1 0 1 1</td>
<td>Output signal is registered and inverted, (no enable function)</td>
</tr>
<tr>
<td>13</td>
<td>PIN_OUTPUT_REGISTERED_ENABLE_REGISTERED_INVERTED</td>
<td>1 1 1 1</td>
<td>Output signal is registered and inverted, the enable/tristate control is registered.</td>
</tr>
</tbody>
</table>
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};
```

**DDR IO Configuration**

The following setting configures the SB_IO into a DDR IO.

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

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

This creates a DDR IO 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.

**High Drive SB_IO**

IO’s in iCE40/iCE40LM device can be configured with different drive strengths to increase the IO 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 IO should be configured as output-only registered IO.

**Synthesis Attribute Syntax:**

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

**Drive Value:**

<table>
<thead>
<tr>
<th>Drive Strength Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Default drive strength. No replication of SB_IO.</td>
</tr>
<tr>
<td>x2</td>
<td>Increase default drive strength by 2. SB_IO replicated once.</td>
</tr>
<tr>
<td>x3</td>
<td>Increase default drive strength by 3. SB_IO replicated twice.</td>
</tr>
</tbody>
</table>

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.
Pull Up Resistor Configuration.

For iCE40UL device, user can configure the internal pull up resistor strength of an IO to a predefined resistor value via attribute settings. By default, all the IO’s 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:

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

<table>
<thead>
<tr>
<th>Resistor Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;3P3K&quot;</td>
<td>Pull up resistor level is 3.3K.</td>
</tr>
<tr>
<td>&quot;6P8K&quot;</td>
<td>Pull up resistor level is 6.8K.</td>
</tr>
<tr>
<td>&quot;10K&quot;</td>
<td>Pull up resistor level is 10K.</td>
</tr>
<tr>
<td>&quot;100K&quot;</td>
<td>Pull up resistor level is 100K. (Default)</td>
</tr>
</tbody>
</table>

Note: PULLUP_RESISTOR attribute is supported only for iCE40UL devices.

Verilog Instantiation

```verbatim
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.
```
Global Buffer Primitives

SB_GB_IO

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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user’s 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
.OUTCLK (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 that 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.

**SB_GB Primitive**

![](global_buffer_output.png)

**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)
);

ICE Technology Library
Lattice Semiconductor Corporation Confidential
PLL Primitives

The Phase Lock Loop (PLL) function is offered as a feature in certain iCE device packages.

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.

iCE40 PLL Primitives

There are 5 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.

SB_PLL40_CORE

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

![SB_PLL40_CORE Diagram]

**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 PLLOUTGLOBAL 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>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_FEEDBACK</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins</td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,...,15</td>
<td>The PLLOUTGLOBAL &amp; 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.</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,...,15</td>
<td>The PLLOUTGLOBALA &amp; PLLOUTCOREA signals are additionally delayed by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.</td>
</tr>
<tr>
<td>SHIFTREG_DIV_MODE</td>
<td>Selects shift register configuration</td>
<td>0,1</td>
<td>Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0→Divide by 4 1→Divide by 7</td>
</tr>
<tr>
<td>PLLOUT_SELECT</td>
<td>Selects the signal to be output at the PLLOUTCORE and PLLOUTGLOBAL ports</td>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency will be output without any phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency will be divided by 2 and then output. No phase shift.</td>
</tr>
<tr>
<td>DIVR</td>
<td>REFERENCECLK divider</td>
<td>0,1,2,...,15</td>
<td>These parameters are used to</td>
</tr>
<tr>
<td>DIVF</td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td>control the output frequency, depending on the FEEDBACK_PATH setting.</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
**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 IO bank (IO Bank 2) or the top IO bank (IO Bank 0), and the source clock is not required inside the FPGA.

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 PLLOUTGLOBAL 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>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_FEEDBACK</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins</td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,...,15</td>
<td>The PLLGLOBAL &amp; 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.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_RELATIVE</td>
<td>Selects the mode for the Fine Delay Adjust block</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,...,15</td>
<td>The PLLGLOBALA &amp; PLLOUTCOREA signals are additionally delayed by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is &quot;FIXED&quot;.</td>
</tr>
</tbody>
</table>
| SHIFTREG_DIV_MODE      | Selects shift register configuration             | 0, 1              | Used when FEEDBACK_PATH is "PHASE_AND_DELAY".
|                        |                                                  |                  | 0→Divide by 4
<p>|                        |                                                  |                  | 1→Divide by 7 |
| PLLOUT_SELECT          | Selects the signal to be output at the PLLGLOBAL &amp; PLLOUTCORE ports | SHIFTREG_0deg     | 0° phase shift only if the setting of FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot; |
|                        |                                                  | SHIFTREG_90deg    | 90° phase shift only if the setting of FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot; and SHIFTREG_DIV_MODE=0 |
|                        |                                                  | GENCLK            | The internally generated PLL frequency will be output without any phase shift. |
|                        |                                                  | GENCLK_HALF       | The internally generated PLL frequency will be divided by 2 and then output. No phase shift. |
| DIVR                   | REFERENCECLK divider                            | 0,1,2,...,15      | These parameters are used to |</p>
<table>
<thead>
<tr>
<th>DIVF</th>
<th>Feedback divider</th>
<th>0,1,...,63</th>
<th>control the output frequency, depending on the FEEDBACK_PATH setting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
**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 IO bank (IO Bank 2) or the top IO bank (IO Bank 0), and in addition to the PLL output, the source clock is also required inside the FPGA.

![SB_PLL40_2_PAD diagram](image)

### 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 PLLOUTGLOBAL 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>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_FEEDBACK</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMIC_DELAY[3:0] pins</td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,…,15</td>
<td>The PLLOUTGLOBAL &amp; 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.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_RELATIVE</td>
<td>Selects the mode for the Fine Delay Adjust block</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMIC_DELAY[7:4] pins</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,…,15</td>
<td>The PLLOUTGLOBALA &amp; 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 &quot;FIXED&quot;.</td>
</tr>
<tr>
<td>SHIFTREG_DIV_MODE</td>
<td>Selects shift register configuration</td>
<td>0,1</td>
<td>Used when FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot;. 0(\rightarrow)Divide by 4 1(\rightarrow)Divide by 7</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTB</td>
<td>Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports</td>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot; and SHIFTREG_DIV_MODE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency will be output to PortB. No phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency will be divided by 2 and then output to PORTB. No phase shift.</td>
</tr>
<tr>
<td>DIVR</td>
<td>REFERENCECLK divider</td>
<td>0,1,2,...,15</td>
<td>These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DIVF</td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTA</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTB</td>
<td>Enables the PLL power-down control</td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>

### SB_PLL40_2F_CORE

The SB_PLL40_2F_CORE primitive should be used when PLL is used to generate 2 different output frequencies, and the source clock of the PLL is driven by FPGA routing i.e. when the PLL source clock originates on the FPGA.

![SB_PLL40_2F_CORE Diagram](image)

#### 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.

**RESETP**: 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>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_FEEDBACK</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins</td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,….15</td>
<td>The PLLOUTGLOBALA &amp; 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.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_RELATIVE</td>
<td>Selects the mode for the Fine Delay Adjust block</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,….15</td>
<td>The PLLOUTGLOBALA &amp; 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”.</td>
</tr>
<tr>
<td>SHIFTREG_DIV_MODE</td>
<td>Selects shift register configuration</td>
<td>0,1</td>
<td>Used when FEEDBACK_PATH is “PHASE_AND_DELAY”: 0→Divide by 4 1→Divide by 7</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTA</td>
<td>Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports</td>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency will be output to PortA. No phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency will be divided by 2 and then output to PORTA. No phase shift.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTB</td>
<td>Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is &quot;PHASE_AND_DELAY&quot; and SHIFTREG_DIV_MODE=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENCLK</td>
<td>The internally generated PLL frequency will be output to PortB. No phase shift.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency will be divided by 2 and then output to PORTB. No phase shift.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVR</td>
<td>REFERENCECLK divider</td>
<td>0,1,2,...,15</td>
<td>These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.</td>
</tr>
<tr>
<td>DIVF</td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTA</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTB</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
**SB_PLL40_2F_PAD**

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

**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 PLLOUTGLOBALA 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 PLLOUTGLOBALB 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>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_FEEBACK</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELY[3:0] pins</td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,….15</td>
<td>The PLLOUTGLOBALA &amp; PLLOUTCOREA signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEBACK is FIXED.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_RELATIVE</td>
<td>Selects the mode for the Fine Delay Adjust block</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELY[7:4] pins</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,….15</td>
<td>The PLLOUTGLOBALA &amp; 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”.</td>
</tr>
<tr>
<td>SHIFTREG_DIV_MODE</td>
<td>Selects shift register configuration</td>
<td>0, 1</td>
<td>Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0⇒Divide by 4 1⇒Divide by 7</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTA</td>
<td>Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports</td>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency will be output to PortA. No phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency will be divided by 2 and then output to PORTA. No phase shift.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------------------------------------------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTB</td>
<td>Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENCLK</td>
<td>The internally generated PLL frequency will be output to PortB. No phase shift.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency will be divided by 2 and then output to PORTB. No phase shift.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVR</td>
<td>REFERENCECLK divider</td>
<td>0,1,2,…,15</td>
<td>These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.</td>
</tr>
<tr>
<td>DIVF</td>
<td>Feedback divider</td>
<td>0,1,…,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,…,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,…,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTA</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTB</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
Hard Macro Primitives

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

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.

![SB_HSOSC Diagram]

**Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENACLKM</td>
<td>Input</td>
<td>Enable High Speed Strobe Generator. Active High.</td>
</tr>
<tr>
<td>CLKM</td>
<td>Output</td>
<td>Strobe Generator Output (12Mhz).</td>
</tr>
</tbody>
</table>

Clock Network Selection

By default the strobe generator use one of the dedicated clock networks in the device to drive the elements. The user 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**

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

![SB_LSOSC Diagram]

**Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENACLKK</td>
<td>Input</td>
<td>Enable Low Power Strobe Generator. Active High.</td>
</tr>
<tr>
<td>CLKK</td>
<td>Output</td>
<td>Strobe Generator Output (10Khz).</td>
</tr>
</tbody>
</table>

**Clock Network Selection**

By default the strobe generator use one of the dedicated clock networks in the device to drive the elements. The user 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] */;
```

**SB_I2C**

The I2C hard IP provides industry standard two pin communication interface that conforms to V2.1 of the I2C bus specification. It could be configured as either master or slave port. In master mode, it support configurable data transfer rate and perform arbitration detection to allow it to operate in multi-master systems. It supports both 7 bits and 10 bits addressing in slave mode with configurable slave address and clock stretching in both master and slave 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.
## Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCLKI</td>
<td>Input</td>
<td>System Clock input.</td>
</tr>
<tr>
<td>SBRWI</td>
<td>Input</td>
<td>System Read/Write Input.</td>
</tr>
<tr>
<td>SBSTBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>SBADRI0</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 0.</td>
</tr>
<tr>
<td>SBADRI1</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 1.</td>
</tr>
<tr>
<td>SBADRI2</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 2.</td>
</tr>
<tr>
<td>SBADRI3</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 3.</td>
</tr>
<tr>
<td>SBADRI5</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 5.</td>
</tr>
<tr>
<td>SBADRI6</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 6.</td>
</tr>
<tr>
<td>SBADRI7</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 7.</td>
</tr>
<tr>
<td>SBDATI0</td>
<td>Input</td>
<td>System Data Input. Bit 0.</td>
</tr>
<tr>
<td>SBDATI1</td>
<td>Input</td>
<td>System Data Input. Bit 1.</td>
</tr>
<tr>
<td>SBDATI2</td>
<td>Input</td>
<td>System Data Input. Bit 2.</td>
</tr>
<tr>
<td>SBDATI3</td>
<td>Input</td>
<td>System Data Input. Bit 3.</td>
</tr>
<tr>
<td>SBDATI5</td>
<td>Input</td>
<td>System Data Input. Bit 5.</td>
</tr>
<tr>
<td>SBDATI7</td>
<td>Input</td>
<td>System Data Input. Bit 7.</td>
</tr>
<tr>
<td>SBDATO0</td>
<td>Output</td>
<td>System Data Output. Bit 0.</td>
</tr>
<tr>
<td>SBDATO1</td>
<td>Output</td>
<td>System Data Output. Bit 1.</td>
</tr>
<tr>
<td>SBDATO2</td>
<td>Output</td>
<td>System Data Output. Bit 2.</td>
</tr>
<tr>
<td>SBDATO3</td>
<td>Output</td>
<td>System Data Output. Bit 3.</td>
</tr>
<tr>
<td>SBDATO5</td>
<td>Output</td>
<td>System Data Output. Bit 5.</td>
</tr>
<tr>
<td>SBDATO7</td>
<td>Output</td>
<td>System Data Output. Bit 7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
<tr>
<td>SBACKO</td>
<td>Output</td>
<td>System Acknowledgement.</td>
</tr>
<tr>
<td>I2CIRQ</td>
<td>Output</td>
<td>I2C Interrupt output.</td>
</tr>
<tr>
<td>I2CWKUP</td>
<td>Output</td>
<td>I2C Wake Up from Standby signal.</td>
</tr>
<tr>
<td>SCLI</td>
<td>Input</td>
<td>Serial Clock Input.</td>
</tr>
<tr>
<td>SCLO</td>
<td>Output</td>
<td>Serial Clock Output</td>
</tr>
<tr>
<td>SCLOE</td>
<td>Output</td>
<td>Serial Clock Output Enable. Active High.</td>
</tr>
<tr>
<td>SDAI</td>
<td>Input</td>
<td>Serial Data Input</td>
</tr>
<tr>
<td>SDAO</td>
<td>Output</td>
<td>Serial Data Output</td>
</tr>
<tr>
<td>SDAOE</td>
<td>Output</td>
<td>Serial Data Output Enable. Active High.</td>
</tr>
</tbody>
</table>

**Parameters**

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

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

**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(sbclk0),  
.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(sda0),  
.SBDAO(sbdao),  
.SBDAOE(sbdaoe)  
);/* synthesis I2C_CLK_DIVIDER= 1 */;

defparam i2cInst0.I2C_SLAVE_INIT_ADDR = "0b1111100001";
defparam i2cInst0.BUS_ADDR74 = "0b00001";

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 Master or Slave SPI port with separate Chip Select Pin. In master 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 etc. for easy communicate with system host.

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

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCLKI</td>
<td>Input</td>
<td>System Clock input.</td>
</tr>
<tr>
<td>SBRWNI</td>
<td>Input</td>
<td>System Read/Write Input.</td>
</tr>
<tr>
<td>SBSTBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>SBADRI0</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 0.</td>
</tr>
<tr>
<td>SBADRI1</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 1.</td>
</tr>
<tr>
<td>SBADRI2</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 2.</td>
</tr>
<tr>
<td>SBADRI3</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 3.</td>
</tr>
<tr>
<td>SBADRI5</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 5.</td>
</tr>
<tr>
<td>SBADRI6</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 6.</td>
</tr>
<tr>
<td>SBADRI7</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 7.</td>
</tr>
<tr>
<td>SBDATI0</td>
<td>Input</td>
<td>System Data Input. Bit 0.</td>
</tr>
<tr>
<td>SBDATI1</td>
<td>Input</td>
<td>System Data input. Bit 1.</td>
</tr>
<tr>
<td>SBDATI2</td>
<td>Input</td>
<td>System Data input. Bit 2.</td>
</tr>
<tr>
<td>SBDATI3</td>
<td>Input</td>
<td>System Data input. Bit 3.</td>
</tr>
<tr>
<td>SBDATI5</td>
<td>Input</td>
<td>System Data input. Bit 5.</td>
</tr>
<tr>
<td>SBDATI7</td>
<td>Input</td>
<td>System Data input. Bit 7.</td>
</tr>
<tr>
<td>SBDATO0</td>
<td>Input</td>
<td>System Data Output. Bit 0.</td>
</tr>
<tr>
<td>SBDATO1</td>
<td>Input</td>
<td>System Data Output. Bit 1.</td>
</tr>
<tr>
<td>SBDATO2</td>
<td>Input</td>
<td>System Data Output. Bit 2.</td>
</tr>
<tr>
<td>SBDATO3</td>
<td>Input</td>
<td>System Data Output. Bit 3.</td>
</tr>
<tr>
<td>SBDATO5</td>
<td>Input</td>
<td>System Data Output. Bit 5.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SBDATO7</td>
<td>Input</td>
<td>System Data Output. Bit 7.</td>
</tr>
<tr>
<td>SBACKO</td>
<td>Output</td>
<td>System Acknowledgement</td>
</tr>
<tr>
<td>SPIIRQ</td>
<td>Output</td>
<td>SPI Interrupt output.</td>
</tr>
<tr>
<td>SPIWKUP</td>
<td>Output</td>
<td>SPI Wake Up from Standby signal.</td>
</tr>
<tr>
<td>MI</td>
<td>Input</td>
<td>Master Input from PAD</td>
</tr>
<tr>
<td>SO</td>
<td>Output</td>
<td>Slave Output to PAD</td>
</tr>
<tr>
<td>SOE</td>
<td>Output</td>
<td>Slave Output Enable to PAD. Active High.</td>
</tr>
<tr>
<td>SI</td>
<td>Input</td>
<td>Slave Input from PAD</td>
</tr>
<tr>
<td>MO</td>
<td>Output</td>
<td>Master Output to PAD</td>
</tr>
<tr>
<td>MOE</td>
<td>Output</td>
<td>Master Output Enable to PAD. Active High.</td>
</tr>
<tr>
<td>SCKI</td>
<td>Input</td>
<td>Slave Clock Input From PAD</td>
</tr>
<tr>
<td>SCKO</td>
<td>Output</td>
<td>Slave Clock Output to PAD</td>
</tr>
<tr>
<td>SCKOE</td>
<td>Output</td>
<td>Slave Clock Output Enable to PAD. Active High.</td>
</tr>
<tr>
<td>SCSNI</td>
<td>Input</td>
<td>Slave Chip Select Input From PAD</td>
</tr>
<tr>
<td>MCSNO0</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 0.</td>
</tr>
<tr>
<td>MCSNO1</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 1.</td>
</tr>
<tr>
<td>MCSNO2</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 2.</td>
</tr>
<tr>
<td>MCSNO3</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 3.</td>
</tr>
<tr>
<td>MCSNOE0</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 0.</td>
</tr>
<tr>
<td>MCSNOE1</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 1.</td>
</tr>
<tr>
<td>MCSNOE2</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 2.</td>
</tr>
<tr>
<td>MCSNOE3</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 3.</td>
</tr>
</tbody>
</table>

### Parameters

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

<table>
<thead>
<tr>
<th>I2C Location</th>
<th>Parameters</th>
<th>Parameter Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Left Corner</td>
<td>BUS_ADDR74</td>
<td>0b0000</td>
<td>Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.</td>
</tr>
<tr>
<td>Lower Right Corner</td>
<td>BUS_ADDR74</td>
<td>0b0001</td>
<td>Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.</td>
</tr>
</tbody>
</table>

### 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.

```c
/* synthesis SPI_CLK_DIVIDER=[Divide Range] */

Divide Range: 0, 1, 2, 3…63. Default is 0.
```
Verilog Instantiation

SB_SPI spiInst0 (  
    .SBCLKI(sbc1ki),  
    .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(sck1),  
    .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";
**iCE5LP (iCE40 Ultra) 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

**SB_HFOSC**

SB_HFOSC primitive generates 48MHz 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.

### Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHFPU</td>
<td>Input</td>
<td>Power up the HFOSC circuit. After power up, oscillator output will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>CLKHFEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100us power up period. Active High.</td>
</tr>
<tr>
<td>CLKHF</td>
<td>Output</td>
<td>HF Oscillator output.</td>
</tr>
</tbody>
</table>

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHF_DIV</td>
<td>&quot;0b00&quot;</td>
<td>Sets 48MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b01&quot;</td>
<td>Sets 24MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b10&quot;</td>
<td>Sets 12MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b11&quot;</td>
<td>Sets 6MHz HFOSC output</td>
</tr>
</tbody>
</table>

### 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 uses one of the dedicated clock networks in the device to drive the elements. The user may configure the oscillator to use the fabric routes instead of the global clock network using the synthesis attribute.

Synthesis Attributes

```verbatim
/* 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";
```

**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.

![SB_LFOSC Ports Diagram]

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKLFPU</td>
<td>Input</td>
<td>Power up the LFOSC circuit. After power up, oscillator output will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>CLKLFEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100us power up period. Active High.</td>
</tr>
<tr>
<td>CLKLF</td>
<td>Output</td>
<td>LF Oscillator output.</td>
</tr>
</tbody>
</table>

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 uses one of the dedicated clock networks in the device to drive the elements. The user may configure the oscillator to use the fabric routes instead of the global clock network using the synthesis attribute.

Synthesis Attributes

```verilog
/* synthesis ROUTE_THROUGH_FABRIC = <value> */
```

Value:
0: Use dedicated clock network. Default option.
1: Use fabric routes.

Verilog Instantiation

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

**SB_LED_DRV_CUR**

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

### Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>Input</td>
<td>Enable to generate constant current source for SB_RGB_DRV and SB_IR_DRV primitives. After Enable, reference current value will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>LEDPU</td>
<td>Output</td>
<td>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.</td>
</tr>
</tbody>
</table>

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)
);
```

**SB_RGB_DRV**

SB_RGB_DRV primitive contains 3 dedicated open drain I/O pins for RGB LED outputs. Each of the RGB LED output isbonded out together with an SB_IO_OD primitive to the package pin. User 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.

![SB_RGB_DRV Diagram](image)

### Ports

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

### 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" = 0mA.  // Set this value to use the associated SB_IO_OD instance at RGB LED location.
"0b000001" = 4mA.
"0b000011" = 8mA.
"0b000111" = 12mA.
"0b011111" = 16mA.
"0b111111" = 20mA.
"0b111111" = 24mA.

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";

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. User 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.
### Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRLEDEN</td>
<td>Input</td>
<td>Enable the SB_IR_DRV primitive. Active High.</td>
</tr>
<tr>
<td>IRPU</td>
<td>Input</td>
<td>Power Up Signal. Connect to LEDPU port of SB_LED_DRV_CUR primitive.</td>
</tr>
<tr>
<td>IRPWM</td>
<td>Input</td>
<td>PWM Input data to drive IRLED pin.</td>
</tr>
<tr>
<td>IRLED</td>
<td>Output</td>
<td>IR LED output.</td>
</tr>
</tbody>
</table>

### 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 = "0b000000000000"

#### Parameter Values:

"0b000000000000" = 0mA  // Set this value to use the associated SB_IO_OD instance at IR LED location.
"0b000000000001" = 50mA
"0b000000000011" = 100mA
"0b000000000111" = 150mA
"0b000000011111" = 200mA
"0b000001111111" = 250mA
"0b000111111111" = 300mA
"0b001111111111" = 350mA
"0b011111111111" = 400mA
"0b111111111111" = 450mA
"0b111111111111" = 500mA

### Verilog Instantiation

```verilog
SB_IR_DRV   IRDRVinst (  
  .IRLEDEN(ENABLE_IRLED),  
  .IRPWM(IR_INPUT),      
  .IRPU(led_power_up),    
  .IRLED(IR_LED)       
);                        
defparam IRDRVinst.IR_CURRENT = "0b1111111111";
```
**SB_RGB_IP**

SB_RGB_IP primitive generates the 3 RGB PWM outputs, to be connected to the LED drivers.

<table>
<thead>
<tr>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal Name</strong></td>
</tr>
<tr>
<td>CLK</td>
</tr>
<tr>
<td>RST</td>
</tr>
<tr>
<td>PARAMSOK</td>
</tr>
<tr>
<td>RGBCOLOR[3:0]</td>
</tr>
<tr>
<td>BRIGHTNESS[3:0]</td>
</tr>
<tr>
<td>BREATHRAMP[3:0]</td>
</tr>
<tr>
<td>BLINKRATE[3:0]</td>
</tr>
<tr>
<td>REDPWM</td>
</tr>
<tr>
<td>GREENPWM</td>
</tr>
<tr>
<td>BLUEPWM</td>
</tr>
</tbody>
</table>

**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)
);
```
**SB_IO_OD**

The SB_IO_OD is the open drain IO primitive. When the Tristate output is enabled, the IO 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.

---

**Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGEPIN</td>
<td>Bidirectional</td>
<td>Bidirectional IO pin.</td>
</tr>
<tr>
<td>LATCHINPUTVALUE</td>
<td>Input</td>
<td>Latches/Holds the input data.</td>
</tr>
<tr>
<td>CLOCKENABLE</td>
<td>Input</td>
<td>Clock enable signal.</td>
</tr>
<tr>
<td>INPUTCLK</td>
<td>Input</td>
<td>Clock for the input registers.</td>
</tr>
<tr>
<td>OUTPUTCLK</td>
<td>Input</td>
<td>Clock for the output registers.</td>
</tr>
<tr>
<td>OUTPUTENABLE</td>
<td>Input</td>
<td>Enable Tristate output. Active high.</td>
</tr>
<tr>
<td>DOUT0</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DOUT1</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DIN0</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
<tr>
<td>DIN1</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
</tbody>
</table>
**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 will result in a non-optimal implementation, since extra LUT will be used to generate the Logic “1”. If the user's 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**

```verilog
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'b0000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
```
**SB_I2C**

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

### Ports

The port interface is similar as ICE40LM SB_I2C primitive. Refer Page 114.

### Parameters

The parameters are same as ICE40LM SB_I2C primitive.

### 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 50ns additional delay to the SDAI signal.

```
/* synthesis SDA_INPUT_DELAYED= value */
```

Value:
0: No delay.
1: Add 50ns delay. (Default value).

**SDA_OUTPUT_DELAYED**

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

```c
/* synthesis SDA_OUTPUT_DELAYED=value */
```

Value:
0: No delay (Default value).
1: Add 50ns delay.

**SB_SPI**

iCE5LP device supports two SPI hard IP primitives, located at lower left corner and lower right corner of the chip. The port interface of SB_SPI is similar as ICE40LM SB_SPI primitive. Refer Page 117.

![SB_SPI Diagram]

**Ports**

The port interface is similar as iCE40LM SB_SPI primitive. Refer Page 117.

**Parameters**

The parameters are same as ICE40LM SB_SPI primitive.

**Synthesis Attribute**

The synthesis attribute is same as ICE40LM SB_SPI primitive.
**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 Page 138 for the list of supported configurations.
Port A (Generated Clock)

### Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Input</td>
<td>Clock input. Applies to all clocked elements in the MAC16 Block.</td>
<td>1'b1</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>Clock Enable Input. Active High.</td>
<td>16'b0</td>
</tr>
<tr>
<td>C[15:0]</td>
<td>Input</td>
<td>Input to the C Register / Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>A[15:0]</td>
<td>Input</td>
<td>Input to the A Register / Direct input to the multiplier blocks / Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>B[15:0]</td>
<td>Input</td>
<td>Input to the B Register / Direct input to the multiplier blocks / Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>Input</td>
<td>Input to the D Register / Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>AHOLD</td>
<td>Input</td>
<td>Hold A registers Data .Controls data flow into the input register A. Active High.</td>
<td>1'b0</td>
</tr>
</tbody>
</table>

0 Update (load) register at next active clock edge.
<table>
<thead>
<tr>
<th>Input</th>
<th>Function Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_HOLD</td>
<td>Hold B registers Data. Controls data flow into the B input register. Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>C_HOLD</td>
<td>Hold C registers Data. Controls data flow into the A input register. Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>D_HOLD</td>
<td>Hold D registers Data. Controls data flow into the A input register. Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>IRSTTOP</td>
<td>Reset input to the A and C input registers, and the pipeline registers in the upper half of the multiplier block. Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>IRSTBOT</td>
<td>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.</td>
<td>1'b0</td>
</tr>
<tr>
<td>ORSTTOP</td>
<td>Reset the high-order bits of the accumulator register ([31:16]). Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>ORSTBOT</td>
<td>Reset the low-order accumulator register bits ([15:0]). Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>OLOADTOP</td>
<td>High-order Accumulator Register Accumulate/Load. Controls whether the accumulator register accepts the output of the adder/subtractor or whether the register is loaded with the value from Input C (or Register C, if configured).</td>
<td>1'b0</td>
</tr>
<tr>
<td>OLOADBOT</td>
<td>Low-order Accumulator Register Accumulate/Load. Controls whether the low-order accumulator register bits (15:0) accepts the output of the adder/subtractor or whether the register is loaded with the value from Input D (or Register D, if configured).</td>
<td>1'b0</td>
</tr>
<tr>
<td>ADDSUBTOP</td>
<td>High-order Add/Subtract. Controls whether the adder/subtractor adds or subtracts.</td>
<td>1'b0: Add</td>
</tr>
<tr>
<td>ADDSUBBOT</td>
<td>Low-order Add/Subtract. Controls whether the adder/subtractor adds or subtracts.</td>
<td>1'b0: Add</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Configuration bits</td>
<td>Parameter Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>NEG_TRIGGER</td>
<td></td>
<td>Controls input clock polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_REG</td>
<td>C0</td>
<td>Input C register Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_REG</td>
<td>C1</td>
<td>Input A register Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B_REG</td>
<td>C2</td>
<td>Input B register Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D_REG</td>
<td>C3</td>
<td>Input D register Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOP_8x8_MULT_REG</td>
<td>C4</td>
<td>Top 8x8 multiplier output register control (point F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOT_8x8_MULT_REG</td>
<td>C5</td>
<td>Bottom 8x8 multiplier</td>
</tr>
</tbody>
</table>

**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.
output register control. (point G) | output is not registered.
--- | ---
P pipelines | 

**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.

**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
<table>
<thead>
<tr>
<th>Function</th>
<th>Code</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOTADDSUB_LOWERINPUT</td>
<td>C18,C17</td>
<td>Selects Input Z for the lower adder/subtractor</td>
<td>00: 16-bit input from B input or associated pipeline register&lt;br&gt;01: 16-bit output from lower 8x8 multiplier, G&lt;br&gt;10: Lower 16-bit output from 16x16 multiplier, H&lt;br&gt;11: Sign extension input SIGNEXTIN. Duplicate 16 bits.</td>
</tr>
<tr>
<td>BOTADDSUB_UPPERINPUT</td>
<td>C19</td>
<td>Selects Input Y for the lower adder/subtractor</td>
<td>0: 16-bit feedback from lower accumulator register, S&lt;br&gt;1: 16-bit input from D input or associated input register.</td>
</tr>
<tr>
<td>BOTADDSUB_CARRYSELECT</td>
<td>C21,C20</td>
<td>Carry/borrow input select to lower adder/subtractor</td>
<td>00: Constant 0&lt;br&gt;01: Constant 1&lt;br&gt;10: Cascade Carry/borrow input from ACCUMCI&lt;br&gt;11: Carry/borrow input from CI</td>
</tr>
<tr>
<td>MODE_8x8</td>
<td>C22</td>
<td>Selects 8x8 Multiplier mode and 8x8 Low-Power Multiplier Blocking Option</td>
<td>0: No effect&lt;br&gt;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.</td>
</tr>
<tr>
<td>A_SIGNED</td>
<td>C23</td>
<td>Indicates whether multiplier input A is signed or unsigned. Applies regardless if input A is 16- or 32-bits wide.</td>
<td>0: Multiplier input A is unsigned.&lt;br&gt;1: Multiplier input A is signed.</td>
</tr>
<tr>
<td>B_SIGNED</td>
<td>C24</td>
<td>Indicates whether multiplier input B is signed or unsigned. Applies regardless if input B is 16- or 32-bits wide.</td>
<td>0: Multiplier input B is unsigned.&lt;br&gt;1: Multiplier input B is signed.</td>
</tr>
</tbody>
</table>
SB_MAC16 Configurations

The following SB_MAC16 functional blocks are supported in iCEcube2.

1. Multiplier.
2. Multiply and Accumulate (MAC).
3. Accumulator (ACC).
4. Add/Subtract (ADD/SUB).
5. 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.

### Table 1: Multiplier Configurations

<table>
<thead>
<tr>
<th>SB_MAC16 Function/Parameter Settings</th>
<th>B_SIGNED</th>
<th>A_SIGNED</th>
<th>MODE</th>
<th>••••</th>
<th>BOTADDSUB_CARRYSELECT</th>
<th>BOTADDSUB_UPPERINPUT</th>
<th>BOTADDSUB_LOWERINPUT</th>
<th>BOTOUTPUT_SELECT</th>
<th>TOPADDSUB_CARRYSELECT</th>
<th>TOPADDSUB_UPPERINPUT</th>
<th>TOPADDSUB_LOWERINPUT</th>
<th>TOPOUTPUT_SELECT</th>
<th>PIPELINE_16x16_MULTI_REG2</th>
<th>PIPELINE_16x16_MULTI_REG1</th>
<th>BOT_8x8_MULTI_REG</th>
<th>TOP_8x8_MULTI_REG</th>
<th>D_REG</th>
<th>B_REG</th>
<th>A_REG</th>
<th>C_REG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>8x8 Multiplier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_all_pipeline_unsigned</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_all_pipeline_signed</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_bypass_unsigned</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_bypass_signed</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>16x16 Multiplier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_all_pipeline_unsigned</td>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 2: MAC Configurations

16 bit MAC: 8x8 input, 16 bit output MAC.
32 bit MAC: 16x16 input, 32 bit output MAC.

<table>
<thead>
<tr>
<th>SB_MAC16 Function/Parameter Settings</th>
<th>B_SIGNED</th>
<th>A_SIGNED</th>
<th>MODE_8x8</th>
<th>BOTADDSUB_CARRYSELECT</th>
<th>TOPADDSUB_CARRYSELECT</th>
<th>BOTADDSUB_UPPERINPUT</th>
<th>TOPADDSUB_UPPERINPUT</th>
<th>BOTADDSUB_LOWERINPUT</th>
<th>TOPADDSUB_LOWERINPUT</th>
<th>BOTOUTPUT_SELECT</th>
<th>TOPOUTPUT_SELECT</th>
<th>PIPELINE_16x16_MULT_REG1</th>
<th>PIPELINE_16x16_MULT_REG2</th>
<th>TOP_8x8_MULT_REG</th>
<th>D_REG</th>
<th>B_REG</th>
<th>A_REG</th>
<th>C_REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bit MAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_16_all_pipelined_unsigned</td>
<td>0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_16_intermediate_register_bypassed_unsigned</td>
<td>0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_16_bypassed_unsigned</td>
<td>0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 bit MAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_all_pipelined_unsigned</td>
<td>0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 3: ACCUMULATOR Configurations

16 bit ACCUMULATOR: 16 bit input, 16 bit output Accumulator.
32 bit ACCUMULATOR: 32 bit input, 32 bit output Accumulator.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mac_32_all_pipelined_cascaded_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_all_pipelined_cin_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_intermediate_register_bypassed_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_intermediate_register_bypassed_cascaded_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_intermediate_register_bypassed_cin_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_bypassed_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_bypassed_cascaded_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_bypassed_cin_unsigned</td>
<td>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</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_intermediate_register_bypassed_signed</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td>1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4: Add/Subtract Configurations

16 bit ADDSUB: 16 bit input, 16 bit output ADDSUB.
32 bit ADDSUB: 32 bit input, 32 bit output ADDSUB.

| 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 |
|--------------------------------------|----------|----------|----------|------------------------|-----------------------|-----------------------|-------------------|------------------------|-----------------------|-----------------------|-------------------|-----------|----------|-----------|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 16 bit ADDSUB                        |          |          |          |                        |                       |                       |                   |                        |                       |                       |                   |          |          |          |          |        |        |        |        |        |        |        |        |        |
| add_sub_16_all_pipelined_unsigned    | 0        | 1        | 0        | 0                      | 1                     | 0                     | 0                 | 0                      | 1                     | 0                     | 0                 | 0        | 1        | 1        | 1        | 1      | 1      | 1      | 1      | 1      | 1      | 1      | 1      | 1      |
Table 5: Multiply Add/Subtract Configurations

16 bit Mult-Add/Sub: 8x8 input, 16 bit output Mult-Add/Sub.
32 bit Mult-Add/Sub: 16x16 input, 32 bit output Mult-Add/Sub.
<table>
<thead>
<tr>
<th>Function</th>
<th>Data</th>
<th>Instruction 1</th>
<th>Instruction 2</th>
<th>Instruction 3</th>
<th>Instruction 4</th>
<th>Instruction 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>mult_add_sub_16_int</td>
<td>0</td>
<td>0 1 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_unsigned</td>
<td>0</td>
<td>0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 bit Mult-Add/Sub</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_all</td>
<td>0</td>
<td>0 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_pipelined_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_all</td>
<td>0</td>
<td>0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_pipelined_cascaded_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_all</td>
<td>0</td>
<td>0 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_pipelined_cin_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_int</td>
<td>0</td>
<td>0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_unsigned</td>
<td>0</td>
<td>0 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_int</td>
<td>0</td>
<td>0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_cascaded_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_int</td>
<td>0</td>
<td>0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_cin_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_by</td>
<td>0</td>
<td>0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>passed_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_by</td>
<td>0</td>
<td>0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>passed_cascaded_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_by</td>
<td>0</td>
<td>0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>passed_cin_unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_add_sub_32_by</td>
<td>0</td>
<td>1 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>passed_signed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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),
    .AHOLD(AHOLD_i),
    .BHOLD(BHOLD_i),
    .CHOLD(CHOLD_i),
    .DHOLD(DHOLD_i),
    .OHOLDTOP(HOLDTOP_i),
    .OHOLDBOT(HOLDBOT_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. BOUTPUT_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. TOUTPUT_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 ;
iCE40UL (iCE40 Ultra Lite) Hard Macros

This section describes the following dedicated hard macro primitives available in iCE40UL device.

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

**SB_HFOSC**

SB_HFOSC primitive generates 48MHz 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.

**Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHFPUL</td>
<td>Input</td>
<td>Power up the HFOSC circuit. After power up, oscillator output will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>CLKHFEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100us power up period. Active High.</td>
</tr>
<tr>
<td>CLKHF</td>
<td>Output</td>
<td>HF Oscillator output.</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Values.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHF_DIV</td>
<td>&quot;0b00&quot;</td>
<td>Sets 48MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b01&quot;</td>
<td>Sets 24MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b10&quot;</td>
<td>Sets 12MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b11&quot;</td>
<td>Sets 6MHz HFOSC output</td>
</tr>
</tbody>
</table>

**Default Signal Values**

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

- Input CLKHFEN: Logic "0"
- Input CLKHFPUL: Logic "0"
Clock Network Selection
By default the oscillator use one of the dedicated clock networks in the device to drive the elements. The user 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";

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.

Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKLFPU</td>
<td>Input</td>
<td>Power up the LFOSC circuit. After power up, oscillator output will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>CLKLFEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100us power up period. Active High.</td>
</tr>
<tr>
<td>CLKLF</td>
<td>Output</td>
<td>LF Oscillator output.</td>
</tr>
</tbody>
</table>

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 uses one of the dedicated clock networks in the device to drive the elements. The user 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] */;

**SB_RGBA_DRV**

SB_RGBA_DRV primitive is the RGB LED drive module which contains 3 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. User 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.

<table>
<thead>
<tr>
<th>Ports</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURREN</td>
<td>Input</td>
<td>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 100us to reach a stable reference current value.</td>
</tr>
<tr>
<td>RGBLEDEN</td>
<td>Input</td>
<td>Enable the SB_RGBA_DRV primitive. Active High.</td>
</tr>
<tr>
<td>RGB0PWM</td>
<td>Input</td>
<td>Input data to drive RGB0 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
</tr>
<tr>
<td>RGB1PWM</td>
<td>Input</td>
<td>Input data to drive RGB1 LED pin. This input is usually</td>
</tr>
<tr>
<td>RGB2PWM</td>
<td>Input</td>
<td>Input data to drive RGB2 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RGB0</td>
<td>Output</td>
<td>RGB0 LED output.</td>
</tr>
<tr>
<td>RGB1</td>
<td>Output</td>
<td>RGB1 LED output.</td>
</tr>
<tr>
<td>RGB2</td>
<td>Output</td>
<td>RGB2 LED output.</td>
</tr>
</tbody>
</table>

**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** = "0b0000000";
- **Parameter RGB1_CURRENT** = "0b0000000";
- **Parameter RGB2_CURRENT** = "0b0000000";
  - Parameter values:
    - "0b0000000" = 0mA. // Set this value to use the associated SB_IO_OD instance at RGB LED location.
    - "0b0000001" = 4mA for Full Mode; 2mA for Half Mode
    - "0b0000011" = 8mA for Full Mode; 4mA for Half Mode
    - "0b0000111" = 12mA for Full Mode; 6mA for Half Mode
    - "0b0001111" = 16mA for Full Mode; 8mA for Half Mode
    - "0b0111111" = 20mA for Full Mode; 10mA for Half Mode.
    - "0b1111111" = 24mA for Full Mode; 12mA for Half Mode.

**Verilog Instantiation**

```verilog
SB_RGBA_DRV RGB0 годVER (  .CURRENT(ENABLE_CURR),  .RGBLEDEN(ENABLE_RGBDRV),  .RGB0PWM(RGB0),  .RGB1PWM(RGB1),  .RGB2PWM(RGB2),  .RGB0(LED0),  .RGB1(LED1),  .RGB2(LED2)  );
```
**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. User 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.

![SB_IR400_DRV Diagram](image)

<table>
<thead>
<tr>
<th>Ports</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_IR400_DRV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Name</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>CURREN</td>
<td>Input</td>
<td>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 100us to reach a stable reference current value.</td>
</tr>
<tr>
<td>IRLEDEN</td>
<td>Input</td>
<td>Enable the SB_IR400_DRV primitive. Active High.</td>
</tr>
<tr>
<td>IRPWM</td>
<td>Input</td>
<td>PWM Input data to drive IRLED pin.</td>
</tr>
<tr>
<td>IRLED</td>
<td>Output</td>
<td>IR LED output.</td>
</tr>
</tbody>
</table>

**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 = "0b000000000";

Parameter Values:
- "0b000000000"; = 0mA. //This is the setting to tristate the IR output to allow it to be used as GPIO (SB_IO_OD)
- "0b000000001"; = 50mA for Full Mode; 25mA for Half Mode.
- "0b000000011"; = 100mA for Full Mode; 50mA for Half Mode.
- "0b000000111"; = 150mA for Full Mode; 75mA for Half Mode.
- "0b000001111"; = 200mA for Full Mode; 100mA for Half Mode.
- "0b000111111"; = 250mA for Full Mode; 125mA for Half Mode.
- "0b001111111"; = 300mA for Full Mode; 150mA for Half Mode.
- "0b111111111"; = 350mA for Full Mode; 175mA for Half Mode.
"0b11111111"; = 400mA for Full Mode; 200mA 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";

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. User 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.

<table>
<thead>
<tr>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SB_BARCODE_DRV Ports</strong></td>
</tr>
<tr>
<td><strong>Signal Name</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>CURREN</td>
</tr>
<tr>
<td>BARCODEN</td>
</tr>
<tr>
<td>BARCODEPWM</td>
</tr>
<tr>
<td>BARCODE</td>
</tr>
</tbody>
</table>

Default Signal Values

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

Input CURREN      : Logic "0"
Input BARCODEN    : 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" = 0mA.  //This is the setting to tristate the BARCODE output to allow it to be used
  // as GPIO (SB_IO_OD)
  "0b0001" = 16.6mA for Full Mode; 8.3mA for Half Mode,
  "0b0011" = 33.3mA for Full Mode; 16.6mA for Half Mode,
  "0b1001" = 66.6mA for Full Mode; 33.3mA for Half Mode,
  "0b1010" = 83.3mA for Full Mode; 41.6mA for Half Mode,
  "0b1111" = 50mA for Full Mode; 25mA for Half Mode

Verilog Instantiation
SB_BARCODE_DRV BARCODEDRVinst (  
  .CURRENT(ENABLE_CURRENT),  
  .BARCODEEN(ENABLE_BARCODEDRV),  
  .BARCODEPWM(BARCODE_PWMINPUT),  
  .BARCODE(BARCODEOUT)  
);

defparam BARCODEDRVinst.CURRENT_MODE = "0b0";
defparam BARCODEDRVinst.BARCODE_CURRENT = "0b1111";

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. User 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.

SB_IR500_DRV DRC Rule
This primitive cannot be instantiated along with SB_BARCODE_DRV or SB_IR400_DRV instance.
## Ports

### SB_IR500_DRV Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURREN</td>
<td>Input</td>
<td>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 100us to reach a stable reference current value.</td>
</tr>
<tr>
<td>IRLEDEN</td>
<td>Input</td>
<td>Enable the SB_IR400_DRV primitive. Active High.</td>
</tr>
<tr>
<td>IRPWM</td>
<td>Input</td>
<td>PWM Input data to drive IRLED pin.</td>
</tr>
<tr>
<td>IRLED1</td>
<td>Output</td>
<td>IR LED output 1.</td>
</tr>
<tr>
<td>IRLED2</td>
<td>Output</td>
<td>IR LED output 2.</td>
</tr>
</tbody>
</table>

### 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_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"; = 0mA. // This is the setting to tristate the BARCODE output to allow it to be used as GPIO (SB_IO_OD).
    - "0b000000000111"; = 50mA for Full Mode; 25mA for Half Mode.
    - "0b000000001111"; = 100mA for Full Mode; 50mA for Half Mode.
    - "0b000000011111"; = 150mA for Full Mode; 75mA for Half Mode.
    - "0b000000111111"; = 200mA for Full Mode; 100mA for Half Mode.
    - "0b000001111111"; = 250mA for Full Mode; 125mA for Half Mode.
    - "0b000011111111"; = 300mA for Full Mode; 150mA for Half Mode.
    - "0b000111111111"; = 350mA for Full Mode; 175mA for Half Mode.
    - "0b011111111111"; = 400mA for Full Mode; 200mA for Half Mode.
    - "0b111111111111"; = 450mA for Full Mode; 225mA for Half Mode.
    - "0b111111111111"; = 500mA for Full Mode; 250mA for Half Mode.

### Verilog Instantiation

```verilog
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";
```
**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.

### Ports

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

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

**Verilog Instantiation**

```verilog
SB_LEDDA_IP  PWMgen_inst ( 
  .LEDDCS(led_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)
);
```

**SB_IR_IP**

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

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKI</td>
<td>Input</td>
<td>Clock input for IR IP</td>
</tr>
<tr>
<td>CSI</td>
<td>Input</td>
<td>Select Signal. Active High to activate the IP. This usually connects to the output of the decoding logic from MSB of the address bus</td>
</tr>
<tr>
<td>DENI</td>
<td>Input</td>
<td>Data Enable. When asserted, indicates that the data and address on the IR Transceiver Control Bus are stabilized and ready to be captured.</td>
</tr>
<tr>
<td>WEI</td>
<td>Input</td>
<td>Data Write Enable. Asserted during WRITE and de-asserted during READ cycle.</td>
</tr>
<tr>
<td>ADRI3</td>
<td>Input</td>
<td>Control Register Address Bit 3</td>
</tr>
<tr>
<td>ADRI2</td>
<td>Input</td>
<td>Control Register Address Bit 2</td>
</tr>
<tr>
<td>ADRI1</td>
<td>Input</td>
<td>Control Register Address Bit 1</td>
</tr>
<tr>
<td>ADRI0</td>
<td>Input</td>
<td>Control Register Address Bit 0</td>
</tr>
<tr>
<td>WDATA7</td>
<td>Input</td>
<td>Write Data Input Bit 7</td>
</tr>
<tr>
<td>WDATA6</td>
<td>Input</td>
<td>Write Data Input Bit 6</td>
</tr>
<tr>
<td>WDATA5</td>
<td>Input</td>
<td>Write Data Input Bit 5</td>
</tr>
<tr>
<td>WDATA4</td>
<td>Input</td>
<td>Write Data Input Bit 4</td>
</tr>
<tr>
<td>WDATA3</td>
<td>Input</td>
<td>Write Data Input Bit 3</td>
</tr>
<tr>
<td>WDATA2</td>
<td>Input</td>
<td>Write Data Input Bit 2</td>
</tr>
<tr>
<td>WDATA1</td>
<td>Input</td>
<td>Write Data Input Bit 1</td>
</tr>
<tr>
<td>WDATA0</td>
<td>Input</td>
<td>Write Data Input Bit 0</td>
</tr>
<tr>
<td>RDATA7</td>
<td>Output</td>
<td>Read Data Output Bit 7</td>
</tr>
<tr>
<td>RDATA6</td>
<td>Output</td>
<td>Read Data Output Bit 6</td>
</tr>
<tr>
<td>RDATA5</td>
<td>Output</td>
<td>Read Data Output Bit 5</td>
</tr>
<tr>
<td>RDATA4</td>
<td>Output</td>
<td>Read Data Output Bit 4</td>
</tr>
<tr>
<td>RDATA3</td>
<td>Output</td>
<td>Read Data Output Bit 3</td>
</tr>
<tr>
<td>RDATA2</td>
<td>Output</td>
<td>Read Data Output Bit 2</td>
</tr>
<tr>
<td>RDATA1</td>
<td>Output</td>
<td>Read Data Output Bit 1</td>
</tr>
<tr>
<td>RDATA0</td>
<td>Output</td>
<td>Read Data Output Bit 0</td>
</tr>
<tr>
<td>EXE</td>
<td>Input</td>
<td>Execute. when asserted, starts the IR Transceiver Hard IP to transmit or receive IR data</td>
</tr>
<tr>
<td>LEARN</td>
<td>Input</td>
<td>Learning Mode control. When asserted the IR Transceiver is in learning mode. The IR Transceiver will receive data instead of transmit data.</td>
</tr>
<tr>
<td>BUSY</td>
<td>Output</td>
<td>Busy status output</td>
</tr>
<tr>
<td>DRDY</td>
<td>Output</td>
<td>Data Buffer Ready status output</td>
</tr>
<tr>
<td>ERR</td>
<td>Output</td>
<td>Data Error status</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>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.</td>
</tr>
<tr>
<td>IRIN</td>
<td>Input</td>
<td>Modulated ON/OFF pulse from IR sensor.</td>
</tr>
<tr>
<td>IROUT</td>
<td>Output</td>
<td>Modulated ON/OFF pulse for IR Transmit.</td>
</tr>
</tbody>
</table>

### 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(busi_o),  
  .DRDY(drdy_o),  
  .ERR(err_o),  
  .RST(rst_i),  
  .IRIN(irin_i),  
  .IROUT(irpulse_w)
);
**SB_IO_OD**

The SB_IO_OD is the open drain IO primitive. When the Tristate output is enabled, the IO 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.

---

**Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGEPIN</td>
<td>Bidirectional</td>
<td>Bidirectional IO pin.</td>
</tr>
<tr>
<td>LATCHINPUTVALUE</td>
<td>Input</td>
<td>Latches/Holds the input data.</td>
</tr>
<tr>
<td>CLOCKENABLE</td>
<td>Input</td>
<td>Clock enable signal.</td>
</tr>
<tr>
<td>INPUTCLK</td>
<td>Input</td>
<td>Clock for the input registers.</td>
</tr>
<tr>
<td>OUTPUTCLK</td>
<td>Input</td>
<td>Clock for the output registers.</td>
</tr>
<tr>
<td>OUTPUTENABLE</td>
<td>Input</td>
<td>Enable Tristate output. Active high.</td>
</tr>
<tr>
<td>DOUT0</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DOUT1</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DIN0</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
<tr>
<td>DIN1</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
</tbody>
</table>
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 will result in a non-optimal implementation, since extra LUT will be used to generate the Logic “1”. If the user’s 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),
 .LATCHINPUTVALUE (latchinputvalue),
 .CLOCKENABLE (clockenable),
 .INPUTCLK (inputclk),
 .OUTPUTCLK (outputclk),
 .OUTPUTENABLE (outputenable),
 .DOUT0 (dout0),
 .DOUT1 (dout1),
 .DIN0 (din0),
 .DIN1 (din1)
);
defparam OpenDrainInst0.PIN_TYPE = 6'b000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
**SB_I2C_FIFO**

ICE40UL device supports two I2C hard IP primitives, located at bottom left corner and bottom right corner of the chip.

### Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKI</td>
<td>Input</td>
<td>System Clock input</td>
</tr>
<tr>
<td>CSI</td>
<td>Input</td>
<td>Select Signal. Active High to activate the IP. This usually connects to the output of the decoding logic from MSB of the address bus</td>
</tr>
<tr>
<td>STBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>WEI</td>
<td>Input</td>
<td>Write Enable Signal</td>
</tr>
<tr>
<td>ADRI3</td>
<td>Input</td>
<td>Control Register Address Bit 3</td>
</tr>
<tr>
<td>ADRI2</td>
<td>Input</td>
<td>Control Register Address Bit 2</td>
</tr>
<tr>
<td>ADRI1</td>
<td>Input</td>
<td>Control Register Address Bit 1</td>
</tr>
<tr>
<td>ADRI0</td>
<td>Input</td>
<td>Control Register Address Bit 0</td>
</tr>
<tr>
<td>DAT19</td>
<td>Input</td>
<td>Data Input Bit 9</td>
</tr>
<tr>
<td>DAT18</td>
<td>Input</td>
<td>Data Input Bit 8</td>
</tr>
<tr>
<td>DAT17</td>
<td>Input</td>
<td>Data Input Bit 7</td>
</tr>
<tr>
<td>DAT16</td>
<td>Input</td>
<td>Data Input Bit 6</td>
</tr>
<tr>
<td>DAT15</td>
<td>Input</td>
<td>Data Input Bit 5</td>
</tr>
<tr>
<td>DAT14</td>
<td>Input</td>
<td>Data Input Bit 4</td>
</tr>
<tr>
<td>DAT13</td>
<td>Input</td>
<td>Data Input Bit 3</td>
</tr>
<tr>
<td>DAT12</td>
<td>Input</td>
<td>Data Input Bit 2</td>
</tr>
<tr>
<td>DAT11</td>
<td>Input</td>
<td>Data Input Bit 1</td>
</tr>
<tr>
<td>DAT10</td>
<td>Input</td>
<td>Data Input Bit 0</td>
</tr>
<tr>
<td>DAT09</td>
<td>Output</td>
<td>Data Output Bit 9</td>
</tr>
<tr>
<td>DAT08</td>
<td>Output</td>
<td>Data Output Bit 8</td>
</tr>
<tr>
<td>DAT07</td>
<td>Output</td>
<td>Data Output Bit 7</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATO6</td>
<td>Output</td>
<td>Data Output Bit 6</td>
</tr>
<tr>
<td>DATO5</td>
<td>Output</td>
<td>Data Output Bit 5</td>
</tr>
<tr>
<td>DATO4</td>
<td>Output</td>
<td>Data Output Bit 4</td>
</tr>
<tr>
<td>DATO3</td>
<td>Output</td>
<td>Data Output Bit 3</td>
</tr>
<tr>
<td>DATO2</td>
<td>Output</td>
<td>Data Output Bit 2</td>
</tr>
<tr>
<td>DATO1</td>
<td>Output</td>
<td>Data Output Bit 1</td>
</tr>
<tr>
<td>DATO0</td>
<td>Output</td>
<td>Data Output Bit 0</td>
</tr>
<tr>
<td>ACKO</td>
<td>Output</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>I2CIRQ</td>
<td>Output</td>
<td>I2C Interrupt output</td>
</tr>
<tr>
<td>I2CWKUP</td>
<td>Output</td>
<td>I2C Wake up from Standby signal</td>
</tr>
<tr>
<td>SRWO</td>
<td>Output</td>
<td>Slave RW “1” master receiving / Slave transmitting “0” master transmitting / Slave receiving</td>
</tr>
<tr>
<td>SCLI</td>
<td>Input</td>
<td>Serial Clock Input</td>
</tr>
<tr>
<td>SCLO</td>
<td>Output</td>
<td>Serial Clock Output</td>
</tr>
<tr>
<td>SCLOE</td>
<td>Output</td>
<td>Serial Clock Output Enable. Active High</td>
</tr>
<tr>
<td>SDAI</td>
<td>Input</td>
<td>Serial Data Input</td>
</tr>
<tr>
<td>SDAO</td>
<td>Output</td>
<td>Serial Data Output</td>
</tr>
<tr>
<td>SDAOE</td>
<td>Output</td>
<td>Serial Data Output Enable. Active High</td>
</tr>
<tr>
<td>FIFOST</td>
<td>Input</td>
<td>RESET for the FIFO logic</td>
</tr>
<tr>
<td>TXFIFOAEMPTY</td>
<td>Output</td>
<td>TX FIFO Status Signal</td>
</tr>
<tr>
<td>TXFIFOEMPTY</td>
<td>Output</td>
<td>TX FIFO Status Signal</td>
</tr>
<tr>
<td>TXFIFOFULL</td>
<td>Output</td>
<td>TX FIFO Status Signal</td>
</tr>
<tr>
<td>RXFIFOAFULL</td>
<td>Output</td>
<td>RX FIFO Status Signal</td>
</tr>
<tr>
<td>RXFIFOEMPTY</td>
<td>Output</td>
<td>RX FIFO Status Signal</td>
</tr>
<tr>
<td>MRDCMPL</td>
<td>Output</td>
<td>Master Read Complete (only valid for Master Read Mode)</td>
</tr>
</tbody>
</table>

Note : Only available if I2C_FIFO_ENB = ENABLED.

Parameters

<table>
<thead>
<tr>
<th>I2C Location</th>
<th>Parameters</th>
<th>Parameter Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Side Corner</td>
<td>I2C_SLAVE_ADDR</td>
<td>0b1111100001</td>
<td>Upper Bits &lt;9:2&gt; can be changed through control registers. Lower bits &lt;1:0&gt; are fixed.</td>
</tr>
<tr>
<td>Right Side</td>
<td>I2C_SLAVE_ADDR</td>
<td>0b1111100010</td>
<td>Upper Bits &lt;9:2&gt; can be changed through control registers. Lower bits &lt;1:0&gt; are fixed.</td>
</tr>
</tbody>
</table>

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 50ns additional delay to the SDAI signal.

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

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

/* synthesis SDA_OUTPUT_DELAYED= value */

Value:
0: No delay (Default value).
1: Add 50ns delay.

I2C_FIFO_ENB

"I2C_FIFO_ENB" attribute is used to enable or disable FIFO

/* synthesis I2C_FIFO_ENB= [value] */

Value:
ENABLED : FIFO mode will be enabled.
DISABLED : FIFO mode will be disabled.
iCE40UP (iCE40 Ultra Plus) Hard Macros

This section describes the following dedicated hard macro primitives available in iCE40UP device.

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

**SB_HFOSC**

SB_HFOSC primitive generates 48MHz 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.

**Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHFPU</td>
<td>Input</td>
<td>Power up the HFOSC circuit. After power up, oscillator output will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>CLKHFEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100us power up period. Active High.</td>
</tr>
<tr>
<td>CLKHF</td>
<td>Output</td>
<td>HF Oscillator output.</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHF_DIV</td>
<td>&quot;0b00&quot;</td>
<td>Sets 48MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b01&quot;</td>
<td>Sets 24MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b10&quot;</td>
<td>Sets 12MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b11&quot;</td>
<td>Sets 6MHz HFOSC output</td>
</tr>
</tbody>
</table>

**Default Signal Values**

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

Input CLKHFEN: Logic "0"
Input CLKHFPNU: Logic "0"
Clock Network Selection
By default the oscillator use one of the dedicated clock networks in the device to drive the elements. The user 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";

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

Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKLFPU</td>
<td>Input</td>
<td>Power up the LFOSC circuit. After power up, oscillator output will be stable after 100us. Active High.</td>
</tr>
<tr>
<td>CLKLFEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100us power up period. Active High.</td>
</tr>
<tr>
<td>CLKLF</td>
<td>Output</td>
<td>LF Oscillator output.</td>
</tr>
</tbody>
</table>

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. The user 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] */;

SB_RGBA_DRV
SB_RGBA_DRV primitive is the RGB LED drive module which contains 3 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. User can either use SB_RGBA_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.

Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURREN</td>
<td>Input</td>
<td>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 100us to reach a stable reference current value.</td>
</tr>
<tr>
<td>RGBLEDEN</td>
<td>Input</td>
<td>Enable the SB_RGBA_DRV primitive. Active High.</td>
</tr>
<tr>
<td>RGB0PWM</td>
<td>Input</td>
<td>Input data to drive RGB0 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
</tr>
<tr>
<td>RGB1PWM</td>
<td>Input</td>
<td>Input data to drive RGB1 LED pin. This input is usually</td>
</tr>
<tr>
<td>LED Input/Output</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>RGB2PWM</td>
<td>Input data to drive RGB2 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
<td></td>
</tr>
<tr>
<td>RGB0</td>
<td>Output RGB0 LED output.</td>
<td></td>
</tr>
<tr>
<td>RGB1</td>
<td>Output RGB1 LED output.</td>
<td></td>
</tr>
<tr>
<td>RGB2</td>
<td>Output RGB2 LED output.</td>
<td></td>
</tr>
</tbody>
</table>

### 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 = "0b000001"**
- **Parameter RGB1_CURRENT = "0b000001"**
- **Parameter RGB2_CURRENT = "0b000001"**
  - Parameter values:
    - "0b000000" = 0mA. // Set this value to use the associated SB_IO_OD instance at RGB LED location.
    - "0b000001" = 4mA for Full Mode; 2mA for Half Mode
    - "0b000011" = 8mA for Full Mode; 4mA for Half Mode
    - "0b000111" = 12mA for Full Mode; 6mA for Half Mode
    - "0b001111" = 16mA for Full Mode; 8mA for Half Mode
    - "0b011111" = 20mA for Full Mode; 10mA for Half Mode
    - "0b111111" = 24mA for Full Mode; 12mA for Half Mode.

### Verilog Instantiation

```verilog
SB_RGBA_DRV RGBA_DRIVER (  
  .CURRENT(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";
```
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.

Ports

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

**Default Signal Values**

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

**Verilog Instantiation**

```verilog
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)
);
```

**SB_IO_OD**

The SB_IO_OD is the open drain IO primitive. When the Tristate output is enabled, the IO 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.
Ports

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGEPIN</td>
<td>Bidirectional</td>
<td>Bidirectional IO pin.</td>
</tr>
<tr>
<td>LATCHINPUTVALUE</td>
<td>Input</td>
<td>Latches/Holds the input data.</td>
</tr>
<tr>
<td>CLOCKENABLE</td>
<td>Input</td>
<td>Clock enable signal.</td>
</tr>
<tr>
<td>INPUTCLK</td>
<td>Input</td>
<td>Clock for the input registers.</td>
</tr>
<tr>
<td>OUTPUTCLK</td>
<td>Input</td>
<td>Clock for the output registers.</td>
</tr>
<tr>
<td>OUTPUTENABLE</td>
<td>Input</td>
<td>Enable Tristate output. Active high.</td>
</tr>
<tr>
<td>DOUT0</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DOUT1</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DIN0</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
<tr>
<td>DIN1</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
</tbody>
</table>

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 will result in a non-optimal implementation, since extra LUT will be used to generate the Logic “1”. If the user’s 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**

```verilog
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;
```
**SB_I2C**

iCE40UP device supports two I2C hard IP primitives, located at upper left corner and upper right corner of the chip.

![SB_I2C Diagram]

**Ports**

The port interface is similar as iCE40LM/iCE5LP SB_I2C primitive. Refer Page 114.

**Parameters**

The parameters are same as ICE40LM/iCE5LP SB_I2C primitive.

**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.

```plaintext
/* 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 50ns additional delay to the SDAI signal.

```plaintext
/* synthesis SDA_INPUT_DELAYED= value */

Value:
```

*ICE Technology Library*

*Lattice Semiconductor Corporation Confidential*
0: No delay.
1: Add 50ns delay. (Default value).

**SDA_OUTPUT_DELAYED**

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

```c
/* synthesis SDA_OUTPUT_DELAYED= value */
```

Value:
0: No delay (Default value).
1: Add 50ns delay.

**SCL_INPUT_FILTERED**

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

```c
/* synthesis SCL_INPUT_FILTERED= value */
```

Value:
0: No Filter delay (Default value).
1: Add 50ns glitch filter delay.

**SB_SPI**

iCE40UP device supports two SPI hard IP primitives, located at lower left corner and lower right right corner of the chip. The port interface of SB_SPI is similar as ICE40LM/iCE5LP SB_SPI primitive. Refer Page 117.
**Ports**

The port interface is similar as iCE40LM/iCE5LP SB_SPI primitive. Refer Page 117.

**Parameters**

The parameters are same as ICE40LM/iCE5LP SB_SPI primitive.

**Synthesis Attribute**

The synthesis attribute is same as ICE40LM/iCE5LP SB_SPI primitive.

**SB_MAC16**

The SB_MAC16 primitive is the dedicated configurable DSP block available in iCE5LP/iCE40UP 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 Page 133 for SB_MAC16 functional model, port details and the list of supported configurations.
**SB_SPRAM256KA**

iCE40UP devices offer four embedded memory blocks of Single Port RAM. Each of these blocks can be configured in 16K x 16 mode. These RAM blocks can be cascaded using logic implemented in the fabric to create larger memories.

### Ports

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Port Width</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>[13:0]</td>
<td>14b'00000000 00000000</td>
<td>This Address input port is used to address the location to be written during the write cycle and read during the read cycle.</td>
</tr>
<tr>
<td>DATAIN</td>
<td>[15:0]</td>
<td>16b'0000000 00000000</td>
<td>The data input bus used to write the data into the memory location specified by address input port during the write cycle.</td>
</tr>
</tbody>
</table>
The Bit Write feature where selective write to individual I/O's can be done using the Maskable Write Enable signals. Each bit masks a nibble of DATAIN.

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.

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.

This is the external clock for the memory.

When this pin is active then memory goes into low leakage mode, there is no change in the output state.

This pin shut down power to periphery and maintain memory contents. The outputs of the memory are pulled low.

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.

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)
**SB_IO_I3C**

ICE40UP devices contain 2 special purpose IO 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.

---

**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 will result in a non-optimal implementation, since an extra LUT will be used to generate the Logic ‘1’. If the user's 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 will result extra LUTs will be 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 via PIN_TYPE [1:0] and PIN_TYPE [5:2] parameter settings respectively. Specific IO 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 IO Standards such as LVCMOS etc is not defined by these tables.

### Input Pin Function Table

<table>
<thead>
<tr>
<th>#</th>
<th>Pin Function Mnemonic</th>
<th>PIN_TYPE[1:0]</th>
<th>Functional Description of Package Pin Input Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIN_INPUT</td>
<td>0 1</td>
<td>Simple input pin (D_IN_0)</td>
</tr>
<tr>
<td>2</td>
<td>PIN_INPUT_LATCH</td>
<td>1 1</td>
<td>Enables internal data changes on the physical input pin by latching the value.</td>
</tr>
<tr>
<td>3</td>
<td>PIN_INPUT_REGISTERED</td>
<td>0 0</td>
<td>Input data is registered in input cell</td>
</tr>
<tr>
<td>4</td>
<td>PIN_INPUT_REGISTERED _LATCH</td>
<td>1 0</td>
<td>Enables internal data changes on the physical input pin by latching the value on the input register</td>
</tr>
<tr>
<td>5</td>
<td>PIN_INPUT_DDR</td>
<td>0 0</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Output Pin Function Table

<table>
<thead>
<tr>
<th>#</th>
<th>Pin Function Mnemonic</th>
<th>PIN_TYPE[5:2]</th>
<th>Functional Description of Package Pin Output Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIN_NO_OUTPUT</td>
<td>0 0 0 0</td>
<td>Disables the output function</td>
</tr>
<tr>
<td>2</td>
<td>PIN_OUTPUT</td>
<td>0 1 1 0</td>
<td>Simple output pin, (no enable)</td>
</tr>
<tr>
<td>3</td>
<td>PIN_OUTPUT_TRISTATE</td>
<td>1 0 1 0</td>
<td>The output pin may be tristated using the enable</td>
</tr>
<tr>
<td>4</td>
<td>PIN_OUTPUT_ENABLE_REGISTERED</td>
<td>1 1 1 0</td>
<td>The output pin may be tristated using a registered enable signal</td>
</tr>
<tr>
<td>5</td>
<td>PIN_OUTPUT_REGISTERED</td>
<td>0 1 0 1</td>
<td>Output registered, (no enable)</td>
</tr>
<tr>
<td>6</td>
<td>PIN_OUTPUTREGISTERED_ENABLE</td>
<td>1 0 0 1</td>
<td>Output registered with enable (enable is not registered)</td>
</tr>
<tr>
<td>7</td>
<td>PIN_OUTPUTREGISTERED_ENABLE_REGISTERED</td>
<td>1 1 0 1</td>
<td>Output registered and enable registered</td>
</tr>
<tr>
<td>8</td>
<td>PIN_OUTPUT_DDR</td>
<td>0 1 0 0</td>
<td>Output 'DDR' data is clocked out on rising and falling clock edges</td>
</tr>
<tr>
<td>9</td>
<td>PIN_OUTPUT_DDR_ENABLE</td>
<td>1 0 0 0</td>
<td>Output data is clocked out on rising and falling clock edges</td>
</tr>
<tr>
<td>10</td>
<td>PIN_OUTPUT_DDR_ENABLE_REGISTERED</td>
<td>1 1 0 0</td>
<td>Output 'DDR' data with registered enable signal</td>
</tr>
<tr>
<td>11</td>
<td>PIN_OUTPUT_REGISTERED_INVERTED</td>
<td>0 1 1 1</td>
<td>Output registered signal is inverted</td>
</tr>
<tr>
<td>12</td>
<td>PIN_OUTPUT_REGISTERED_ENABLE_INVERTED</td>
<td>1 0 1 1</td>
<td>Output signal is registered and inverted, (no enable function)</td>
</tr>
<tr>
<td>13</td>
<td>PIN_OUTPUT_REGISTERED_ENABLE_INVERTED</td>
<td>1 1 1 1</td>
<td>Output signal is registered and inverted, the enable/tristate control is registered.</td>
</tr>
</tbody>
</table>
Syntex 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.

The user can configure the internal pull up resistor strength of an IO to a predefined resistor value via 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 via PULLUP_RESISTOR attribute. The PULLUP_RESISTOR value is effective only when PULLUP parameter is set to 1.

Both the PULLUP & 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 the user to control the pull up resistor network dynamically.

Synthesis Attribute Syntax:

    /* synthesis PULLUP_RESISTOR = "3P3K" */

**Resistor Value:**

<table>
<thead>
<tr>
<th>Resistor Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;3P3K&quot;</td>
<td>Pull up resistor level is 3.3K.</td>
</tr>
<tr>
<td>&quot;6P8K&quot;</td>
<td>Pull up resistor level is 6.8K.</td>
</tr>
<tr>
<td>&quot;10K&quot;</td>
<td>Pull up resistor level is 10K.</td>
</tr>
</tbody>
</table>

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

Verilog Instantiation

    SB_IO_I3C_I3CIO_INST
    (                         // User's Pin signal name
        .PACKAGE_PIN (package_Pin), // Latches/holds the Input value
        .LATCH_INPUT_VALUE (latch_input_value), // Clock Enable common to input and output clock
        .CLOCK_ENABLE (clock_enable),
        .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";
Device Configuration Primitives

**SB_WARMBOOT**

iCE FPGA devices permit the user to load a different configuration image during regular operation. Through the use of the Warm Boot Primitive, the user can load one of 4 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.*

![SB_WARMBOOT Diagram](image)

The selection of one of these 4 images is accomplished through 2 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 i.e. every time the user wishes 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 4 images. These addresses should be specified in the iCEcube2 software as per the Warm Boot Application Note.

**Verilog Instantiation**

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