LATTICE ICE™ Technology Library

Version 2.6
April 12, 2014
<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. Modified</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>
</tbody>
</table>
# Table of Contents

Register Primitives .................................................................................................................. 6

- SB_DFF ............................................................................................................................. 6
- SB_DFFE ........................................................................................................................... 8
- SB_DFFSR ........................................................................................................................ 10
- SB_DFFR ........................................................................................................................... 12
- SB_DFFSS .......................................................................................................................... 14
- SB_DFFS ............................................................................................................................ 16
- SB_DFFESR ...................................................................................................................... 18
- SB_DFFER ......................................................................................................................... 20
- SB_DFFESS ....................................................................................................................... 22
- SB_DFFES ......................................................................................................................... 24
- SB_DFFN ........................................................................................................................... 26
- SB_DFFNE ......................................................................................................................... 28
- SB_DFFNSR ...................................................................................................................... 30
- SB_DFFNR ........................................................................................................................ 32
- SB_DFFNSS ...................................................................................................................... 34
- SB_DFFNS ........................................................................................................................ 36
- SB_DFFNESR ................................................................................................................... 38
- SB_DFFNER ....................................................................................................................... 40
- SB_DFFNESS ................................................................................................................... 42
- SB_DFFNES ...................................................................................................................... 44

Combinational Logic Primitives ............................................................................................ 46

- SB_LUT4 ............................................................................................................................ 46
- SB_CARRY .......................................................................................................................... 48

Block RAM Primitives .......................................................................................................... 50

iCE65 Block RAM .................................................................................................................... 50

- SB_RAM4K ......................................................................................................................... 55
- SB_RAM4KNR ..................................................................................................................... 55
- SB_RAM4KNW .................................................................................................................... 55
- SB_RAM4KNRW .................................................................................................................. 55

iCE40 Block RAM .................................................................................................................... 56

- SB_RAM256x16 ................................................................................................................... 57
- SB_RAM256x16NR ............................................................................................................. 59
- SB_RAM256x16NW ........................................................................................................... 60
- SB_RAM256x16NRRNW .................................................................................................... 62
- SB_RAM512x8 ................................................................................................................... 65
- SB_RAM512x8NR .............................................................................................................. 67
- SB_RAM512x8NW ............................................................................................................. 68
- SB_RAM512x8NRRNW ..................................................................................................... 70
- SB_RAM1024x4 .................................................................................................................. 73
- SB_RAM1024x4NR .......................................................................................................... 75
- SB_RAM1024x4NW .......................................................................................................... 76
<table>
<thead>
<tr>
<th>Primitive</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM1024x4NRNW</td>
<td>78</td>
</tr>
<tr>
<td>SB_RAM2048x2</td>
<td>81</td>
</tr>
<tr>
<td>SB_RAM2048x2NR</td>
<td>83</td>
</tr>
<tr>
<td>SB_RAM2048x2NW</td>
<td>84</td>
</tr>
<tr>
<td>SB_RAM2048x2N RNW</td>
<td>86</td>
</tr>
<tr>
<td>SB_RAM40_4K</td>
<td>88</td>
</tr>
<tr>
<td>IO Primitives</td>
<td>93</td>
</tr>
<tr>
<td>SB_IO</td>
<td>93</td>
</tr>
<tr>
<td>Global Buffer Primitives</td>
<td>97</td>
</tr>
<tr>
<td>SB_GB_IO</td>
<td>97</td>
</tr>
<tr>
<td>SB_GB Primitive</td>
<td>98</td>
</tr>
<tr>
<td>PLL Primitives</td>
<td>99</td>
</tr>
<tr>
<td>iCE65 PLL Primitives</td>
<td>99</td>
</tr>
<tr>
<td>SB_PLL_CORE</td>
<td>99</td>
</tr>
<tr>
<td>SB_PLL_PAD</td>
<td>102</td>
</tr>
<tr>
<td>SB_PLL_2_PAD</td>
<td>104</td>
</tr>
<tr>
<td>iCE40 PLL Primitives</td>
<td>107</td>
</tr>
<tr>
<td>SB_PLL40_CORE</td>
<td>107</td>
</tr>
<tr>
<td>SB_PLL40_PAD</td>
<td>111</td>
</tr>
<tr>
<td>SB_PLL40_2_PAD</td>
<td>115</td>
</tr>
<tr>
<td>SB_PLL40_2F_CORE</td>
<td>118</td>
</tr>
<tr>
<td>SB_PLL40_2F_PAD</td>
<td>122</td>
</tr>
<tr>
<td>Hard Macro Primitives</td>
<td>126</td>
</tr>
<tr>
<td>iCE40LM Hard Macros</td>
<td>126</td>
</tr>
<tr>
<td>SB_HSOSC (For HSSG)</td>
<td>126</td>
</tr>
<tr>
<td>SB_LSOSC (For LPSG)</td>
<td>127</td>
</tr>
<tr>
<td>SB_I2C</td>
<td>127</td>
</tr>
<tr>
<td>SB_SPI</td>
<td>130</td>
</tr>
<tr>
<td>Device Configuration Primitives</td>
<td>134</td>
</tr>
<tr>
<td>SB_WARMBOOT</td>
<td>134</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.

\[
\begin{array}{c}
D & \text{SB_DFF} & Q \\
\end{array}
\]

<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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Power on State**

Key

- Rising Edge
- 1 High logic level
- 0 Low logic level
- X Don't care
- ? Unknown

**HDL use**

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

**Verilog Instantiation**

// SB_DFF - D Flip-Flop.

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

// End of SB_DFF instantiation
VHDL Instantiation

-- SB_DFF - D Flip-Flop.

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

-- End of SB_DFF instantiation
**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.

### Inputs vs. Output

<table>
<thead>
<tr>
<th>E</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Previous Q</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
<td>X</td>
<td>0</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
);```
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 | C | Q |
+---+---+---+---+
| 1 | X | X | 0 |
| X | X | 0 | No Change |
| 0 | 0 | X | 0 |
| 0 | 1 | X | 1 |
| X | 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’
Input R: Logic ‘0’

**Verilog Instantiation**
```
// SB_DFFSR - D Flip-Flop, Reset is synchronous with the rising clock edge
SB_DFFSR SB_DFFSR_inst (                      // Registered Output
   .Q(Q),                                      // Clock
   .C(C),                                      // Data
   .D(D),                                      // Synchronous Reset
   .R(R)
);
```

// 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.

---

### Inputs

<table>
<thead>
<tr>
<th>R</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

### Power On State

<table>
<thead>
<tr>
<th>R</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>0</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_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>

<table>
<thead>
<tr>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>?</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**

```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

*ICE Technology Library*
*Lattice Semiconductor Corporation Confidential*
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>
</tbody>
</table>

Power on State: X X 0

Key

- ✓ Rising Edge
- 1 High logic level
- 0 Low logic level
- X Don't care
- ? Unknown

**HDL Usage**

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

**Default Signal Values**

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

- Input D: Logic ‘0’
- Input C: Logic ‘0’
- Input S: Logic ‘0’

**Verilog Instantiation**

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

SB_DFFS   SB_DFFS_inst ( //Registered Output
   .Q(Q),
   .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 1</td>
<td>E 1</td>
</tr>
<tr>
<td>X 0</td>
<td>X X</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>1 1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X X 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_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.

![Flip-Flop Diagram]

<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>
</tbody>
</table>

Power on State: X X X 0

Key:
- R: 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_DFFESESS**
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_DFFESESS](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**

```verilog
// SB_DFFESESS - D Flip-Flop, Set is synchronous with rising clock edge and Clock Enable.
SB_DFFESESS  SB_DFFESESS_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
VHDL Instantiation

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

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

-- End of SB_DFFESS instantiation
**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
);
```

Key

- Rising Edge
- 1 High logic level
- 0 Low logic level
- X Don’t care
- ? Unknown
// 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></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Power on State

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></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’

**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.

![D Flip-Flop Diagram]

<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>

**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
);```

ICE Technology Library
Lattice Semiconductor Corporation Confidential
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.

![D Flip-Flop Diagram](image)

<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**

```verilog
// 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

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>Input</th>
<th>D</th>
<th>CLK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Power on State</td>
<td>X</td>
<td>X</td>
<td>0</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 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.

<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**

```verilog
// SB_DFFNSS - D Flip-Flop – Negative Edge Clock, Set is synchronous with the falling clock edge,
SB_DFFNSS SB_DFFNSS_inst ( // Registered Output
 .Q(Q),
 .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, -- with .

    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.

<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>

<table>
<thead>
<tr>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>➜  Falling Edge</td>
</tr>
<tr>
<td>1   High logic level</td>
</tr>
<tr>
<td>0   Low logic level</td>
</tr>
<tr>
<td>X   Don't care</td>
</tr>
<tr>
<td>?   Unknown</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

```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.

![Diagram of SB_DFFNESR](image)

<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>
</tbody>
</table>

Power on State: X X X 0

**Key**
- Falling Edge
- 1: High logic level
- 0: Low logic level
- X: Don’t care
- ?: Unknown

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

**Default Signal Values**
The iCEcube2 software assigns the following signal values to unconnected input ports:

- Input D: Logic ‘0’
- Input C: Logic ‘0’
- Input R: Logic ‘0’
- Input E: Logic ‘1’

Note that explicitly connecting a Logic ‘1’ value to port E 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 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

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**
- F 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>

**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**

```verilog
// 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**

```vhdl
-- 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.

![LUT Diagram]

**Initialization values**
LUT state initialization parameter LUT_INIT = 16'hxxxx;

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>LUT_INIT[0]</td>
</tr>
<tr>
<td>I2</td>
<td>LUT_INIT[1]</td>
</tr>
<tr>
<td>I1</td>
<td>LUT_INIT[2]</td>
</tr>
<tr>
<td>I0</td>
<td>LUT_INIT[3]</td>
</tr>
<tr>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>O</td>
<td>LUT_INIT[15]</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_LUT4 : 4-input Look-Up Table

SB_LUT4 inst (SB_LUT4_inst ( O (O), // output
   .I0 (I0), // data input 0
   .I1 (I1), // data input 1
   .I2 (I2), // data input 2
   .I3 (I3)  // data input 3
);

defparam SB_LUT4 inst.LUT_INIT=X"0001";
   //LUT state initialization parameter, 16 bits.

//End of SB_LUT4 instantiation

VHDL Instantiation

-- SB_LUT4 : 4-input Look-Up Table

SB_LUT4 inst: SB_LUT4
generic map(
   LUT_INIT => X"0001" -- LUT state initialization parameter, 16 bits
)
port map (I0 => I0,
   I1 => I1,
   I2 => I2,
   I3 => I3,
   O => O
);
**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>I0</td>
<td>I1</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 (  
    CO => 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.

iCE65 Block RAM

Each iCE65 device includes multiple high-speed synchronous RAM blocks (RAM4K), each 4Kbit in size. A RAM4K block has separate write and read ports, each with independent control signals. Additionally, the write port has an Active-Low bit-line write-enable control; each write-port data bit has an individual write-enable control. By default, input and output data is 16 bits wide, although the data width is configurable using programmable logic and, if needed, multiple RAM4K blocks. The data contents of the RAM4K block are optionally pre-loaded during ICE device configuration.

RAM4K Naming Convention Rules

The SiliconBlue Technologies convention for the RAM4K 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.

<table>
<thead>
<tr>
<th>RAM Primitive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM4K</td>
<td>Posedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM4KNR</td>
<td>Negedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM4KNW</td>
<td>Posedge Read clock, Negedge Write clock</td>
</tr>
<tr>
<td>SB_RAM4KNRNW</td>
<td>Negedge Read clock, Negedge Write clock</td>
</tr>
</tbody>
</table>

RAM4K blocks have separate write and read ports, each with independent control signals.

The data contents of the RAM4K block are optionally pre-loaded during ICE device configuration. If the RAM4K blocks are not pre-loaded during configuration, then the resulting configuration bitstream image is smaller.
If an uninitialized RAM4K block is used in the application, then the application must initialize the RAM contents to guarantee the data value.

The following table lists the signals for both ports. Additionally, the write port has an active-Low bit-line write-enable control:

<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</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>INIT_0, …, INIT_F</td>
<td>Verilog parameter</td>
<td>RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16x256=4096 total bits)</td>
</tr>
</tbody>
</table>
Write Operation

1. Supply a valid address on the WADDR[7:0] address input port

2. Supply valid data on the WDATA[15:0] data input port

To write or mask selected data bits, set the associated bit write MASK accordingly. For example, write operations on data bit Data[i] is controlled by the associated MASK[i] input: MASK[i] = 0: Enable write operations for data line Data[i]

- MASK[i] = 1: Mask write operations for data line Data[i]
- Enable the RAM4K write port (WE = 1)

3. Apply a rising clock edge on WCLK

<table>
<thead>
<tr>
<th>Operation</th>
<th>WDATA[15:0] Data</th>
<th>MASK[15:0] Bit Enable</th>
<th>WADDR[7:0] Address</th>
<th>WE Enable</th>
<th>WCLK Clock</th>
<th>RAM Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>No Change</td>
</tr>
<tr>
<td>Disabled</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>No Change</td>
</tr>
<tr>
<td>Write Data</td>
<td>D[i]</td>
<td>MASK[i]=0</td>
<td>WADDR</td>
<td>1</td>
<td>†</td>
<td>RAM[WADDR[i]] = D[i]</td>
</tr>
<tr>
<td>Masked Write</td>
<td>X</td>
<td>MASK[i]=1</td>
<td>WADDR</td>
<td>1</td>
<td>†</td>
<td>RAM[WADDR[i]] = No Change</td>
</tr>
</tbody>
</table>

Read Operation

The following table describes various read operations for a RAM4K block. All RAM4K read operations are synchronized to the rising edge of RCLK.

<table>
<thead>
<tr>
<th>Operation</th>
<th>RADDR[7:0] Address</th>
<th>RE Enable</th>
<th>RCLK Clock</th>
<th>RDATA[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>After configuration, before first valid Read Data operation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Undefined</td>
</tr>
<tr>
<td>Disabled</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>No Change</td>
</tr>
<tr>
<td>Read Data</td>
<td>RA</td>
<td>1</td>
<td>†</td>
<td>RAM[RADDR]</td>
</tr>
</tbody>
</table>

To read data from the RAM4K block

1. Supply a valid address on the RADDR[7:0] address input port

2. Enable the RAM4K read port (RE = 1)

3. Apply a rising clock edge on RCLK

Default Signal Values

The iCEcube2 software assigns logic value ‘0’ to all unconnected input ports, with the exception of the RCLKE and WCLKE ports.
The RCLKE and WCLKE ports are always enabled by default i.e. if left unconnected the software will automatically assign a logic value ‘1’ to these ports. Note that explicitly connecting a logic ‘1’ value to ports RCLKE and WCLKE 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 always maintain the clocks in an enabled state, it is recommended that these ports be left unconnected.

Note that the Read Enable (RE) and Write Enable (WE) ports are always disabled by default, since they are tied-off to logic ‘0’ by the software, unless explicitly enabled by the user.

**Verilog Instantiation**

The following instantiation is for the base SB_RAM4K, all other RAM4K based primitives share the same format with the only difference being the port name changes. All primitives share the same parameter for data initialization after power on reset.

// SB_RAM4K with data initialization after power on reset

```
SB_RAM4K_with_INIT : SB_RAM4K
```

defparam SB_RAM4K_with_INIT.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;

**VHDL Instantiation**

```vhdl```
- - SB_RAM4K with data initialization after power on reset

```
SB_RAM4K_with_INIT : SB_RAM4K
```
```
defparam SB_RAM4K_with_INIT.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam SB_RAM4K_with_INIT.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;
```
```
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,  
RADDR => RADDR,  
RCLK => RCLK,  
RCLKE => RCLKE,  
RE => RE,  
WADDR => WADDR,  
WCLK => WCLK,  
WCLKE => WCLKE,  
WDATA => WDATA,  
MASK => MASK,  
WE => WE  
);
The following are the complete list of RAM4K based primitives

**SB_RAM4K**

SB_RAM4K //Pozedge clock RCLK WCLK
(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

**SB_RAM4KNR**

SB_RAM4KNR // Negative edged Read Clock – i.e. RCLKN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

**SB_RAM4KNW**

SB_RAM4KNW // Negative edged Write Clock – i.e. WCLKN
(RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

**SB_RAM4KNRW**

SB_RAM4KNRW // Negative edged Read and Write – i.e. RCLKN WRCKLN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
**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>16 [15:0]</td>
</tr>
<tr>
<td>SB_RAM256x16NR</td>
<td>512x8 (4K)</td>
<td>9 [8:0]</td>
<td>8 [7:0]</td>
<td>8 [8:0]</td>
<td>8 [7:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</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>
</tbody>
</table>

The SiliconBlue 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_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

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

SB_RAM256x16NR

SB_RAM256x16NR  // Negative edged Read Clock – i.e. RCLKN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

SB_RAM256x16NR  ram256x16NR_inst (  
  .RDATA(RDATA_c[15:0]),
  .RADDR(RADDR_c[7:0]),
  .RCLKN(RCLKN_c),
  .RCLKE(RCLKE_c),
  .RE(RE_c),
  .WADDR(WADDR_c[7:0]),
  .WCLK(WCLK_c),
  .WCLKE(WCLKE_c),
  .WDATA(WDATA_c[15:0]),
  .WE(WE_c),
  .MASK(MASK_c[15:0])
);

defparam ram256x16nr_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nr_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
VHDL Instantiation:

ram256x16nr_inst : SB_RAM256x16NR

-- Generic map
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"
);
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]),  
  .WCLKN(WCLKN_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";
```
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
);

port map (  
  RDATA => RDATA_c,  
  RADDR => RADDR_c,  
  RCLK => RCLK_c,  
  RCLKE => RCLKE_c,  
  RE => RE_c,  
  WADDR => WADDR_c,  
  WCLKN => WCLKN_c,  
  WCLKE => WCLKE_c,  
  WDATA => WDATA_c,  
  MASK => MASK_c,  
  WE => WE_c  
);

SB_RAM256x16NRNW

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

Verilog Instantiation:

SB_RAM256x16NRNW  ram256x16nrnw_inst (  
  .RDATA(RDATA_c[15:0]),  
  .RADDR(RADDR_c[7:0]),  
  .RCLK(RCLKN_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[7:0]),  
  .WCLKN(WCLKN_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[15:0]),  

VIC

Lattice Semiconductor Corporation
Confidential

% WE(WE_c),
% .MASK(MASK_c[15:0])

); defparam ram256x16nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

ram256x16nrnw_inst : SB_RAM256x16NRNW
generic map (
    INIT_0 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_6 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_7 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_8 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_9 =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_A =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_B =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_C =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_D =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_E =>
    x"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_F =>
    x"0000000000000000000000000000000000000000000000000000000000000000"
INIT_B => "000000000000000000000000000000000000000000000",
INIT_C => "0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => "0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => "0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => "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
);
SB_RAM512x8

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:

SB_RAM512x8   ram512x8_inst (  
   .RDATA(RDATA_c[7:0]),
   .RADDR(RADDR_c[8:0]),
   .RCLK(RCLK_c),
   .RCLKE(RCLKE_c),
   .RE(RE_c),
   .WADDR(WADDR_c[8:0]),
   .WCLK(WCLK_c),
   .WCLKE(WCLKE_c),
   .WDATA(WDATA_c[7:0]),
   .WE(WE_c)
);

defparam ram512x8_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

ram512x8_inst : SB_RAM512x8
generic map (  
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
  )
port map (  
  RDATA => RDATA_c,
  RADDR => RADDR_c,
  RCLK => RCLK_c,
  RCLKE => RCLKE_c,
  RE => RE_c,
  )
WADDR => WADDR_c,
WCLK=> WCLK_c,
WCLKE => WCLKE_c,
WDATA => WDATA_c,
WE => WE_c
);

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, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```verilog
SB_RAM512x8NW ram512x8nw_inst (  
    .RDATAR (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 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F =>
"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_RAM512x8NRNW

SB_RAM512x8NRNW     // Negative edged Read and Write – i.e. RCLKN WRCKLN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
Verilog Instantiation:

SB_RAM512x8NRNW     ram512x8nrnw_inst (  
    .RDATA(RDATA_c[7:0]),
    .RADDR(RADDR_c[8:0]),
    .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 ram512x8nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

ram512x8nrnw_inst : SB_RAM512x8NRNW
generic map (  
  INIT_0 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_1 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_2 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_3 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_4 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_5 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_6 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_7 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_8 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_9 => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_A => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_B => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_C => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_D => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_E => x"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_F => x"0000000000000000000000000000000000000000000000000000000000000000")
port map (  
  RDATA => RDATA_c,  
  RADDR => RADDR_c,  
  RCLKN => RCLKN_c,
RCLKE => RCLKE_c,  
RE => RE_c,  
WADDR => WADDR_c,  
WCLKN => WCLKN_c,  
WCLKE => WCLKE_c,  
WDATA => WDATA_c,  
WE => WE_c  
);
SB_RAM1024x4

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:

```verilog
SB_RAM1024x4   ram1024x4_inst (  
   .RDATA(RDATA_c[3:0]),  
   .RADDR(RADDR_c[9:0]),  
   .RCLK(RCLK_c),  
   .RCLKE(RCLKE_c),  
   .RE(RE_c),  
   .WADDR(WADDR_c[3:0]),  
   .WCLK(WCLK_c),  
   .WCLKE(WCLKE_c),  
   .WDATA(WDATA_c[9:0]),  
   .WE(WE_c)  
);
```

defparam ram1024x4_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
```
defparam ram1024x4_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

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

SB_RAM1024x4NR

SB_RAM1024x4NR // Negative edged Read Clock – i.e. RCLKN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

SB_RAM1024x4NR ram1024x4nr_inst (  
  .RDATA(RDATA_c[3:0]),  
  .RADDR(RADDR_c[9:0]),  
  .RCLKN(RCLKN_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[3:0]),  
  .WCLK(WCLK_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[9:0]),  
  .WE(WE_c)  
);

defparam ram1024x4nr_inst.INIT_0 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_1 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_2 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_3 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_4 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_5 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_6 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_7 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_8 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_9 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_A =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_B =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_C =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_D =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_E =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nr_inst.INIT_F =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
VHDL Instantiation:

```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,  
        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:

```verilog
SB_RAM1024x4NW  ram1024x4nw_inst (  
    .RDATA(RDATA_c[3:0]),
    .RADDR(RADDR_c[9:0]),
    .RCLK(RCLK_c),
    .RCLKE(RCLKE_c),
    .RE(RE_c),
    .WADDR(WADDR_c[3:0]),
    .WCLKN(WCLKN_c),
    .WCLKE(WCLKE_c),
    .WDATA(WDATA_c[9:0]),
    .WE(WE_c)
);
    defparam ram1024x4_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_10 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_11 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_12 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_13 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_14 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_15 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
);
```

VHDL Instantiation:

```vhdl
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_10 =>
  x"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_11 =>
  x"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_12 =>
  x"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_13 =>
  x"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_14 =>
  x"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_15 =>
  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_RAM1024x4NRNW

SB_RAM1024x4NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN
(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

SB_RAM1024x4NRNW  ram1024x4nrnw_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 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"
);
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

SB_RAM2048x2

Verilog Instantiation:

SB_RAM2048x2    ram2048x2_inst (  
    .RDAT(RDATA_c[2:0]),  
    .RADDR(RADDR_c[10:0]),  
    .RCLK(RCLK_c),  
    .RCLKE(RCLKE_c),  
    .RE(RE_c),  
    .WADDR(WADDR_c[2:0]),  
    .WCLK(WCLK_c),  
    .WCLKE(WCLKE_c),  
    .WDATA(WDATA_c[10:0]),  
    .WE(WE_c)  
);  
defparam ram2048x2_inst.INIT_0 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2_inst.INIT_1 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2_inst.INIT_2 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2_inst.INIT_3 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2_inst.INIT_4 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2_inst.INIT_5 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;  
defparam ram2048x2_inst.INIT_6 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

Ram2048x2_inst :  SB_RAM2048x2
generic map (  
  INIT_0 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
  )
port map (  
  RDATA  =>  RDATA_c,
  RADDR  =>  RADDR_c,
  RCLK   =>  RCLK_c,
  RCLKE  =>  RCLK_c,
  RE     =>  RE_c,
  WADDR  =>  WADDR_c,
  WCLK   =>  WCLK_c,
  WCLKE  =>  WCLK_c,
WDATA => WDATA_c,  
WE => WE_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[2:0]),  
  .RADDR(RADDR_c[10:0]),  
  .RCLKN(RCLKN_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[2:0]),  
  .WCLK(WCLK_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[10:0]),  
  .WE(WE_c)
);

defparam ram2048x2nr_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nr_inst.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

ram2048x2nr_inst :  SB_RAM2048x2NR

generic map (  
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
  ...
```
INIT_1 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => x"0000000000000000000000000000000000000000000000000000000000000000"
);
port map (  
    RDATA => RDATA_c, 
    RADDR => RADDR_c, 
    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. WCLK
(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM2048x2NW  ram2048x2nw_inst (  
    .RDATAD(RDATA_c[2:0]),  
    .RADDRD(RADDR_c[10:0]),  
    .RCLKRD(RCLK_c),  
    .RCLKER(RCLKE_c),  
    .RED(RE_c),  
    .WADDRD(WADDR_c[2:0]),  
    .WCLKN(WCLKN_c),  
    .WCLKE(WCLKE_c),  
    .WDATAD(WDATA_c[10:0]),  
    .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)
);

**SB_RAM2048x2NRNW**

SB_RAM2048x2NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN
(RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM2048x2NRNW    ram2048x2nrcnw_inst (  
    .RDATA(RDATA_c[2:0]),  
    .RADDR(RADDR_c[10:0]),  
    .RCLKN(RCLKN_c),  
    .RCLKE(RCLKE_c),  
    .RE(RE_c),  
    .WADDR(WADDR_c[2:0]),  
    .WCLKN(WCLKN_c),  
    .WCLKE(WCLKE_c),  
    .WDATA(WDATA_c[10:0]),  
    .WE(WE_c)
);

defparam ram2048x2nrcnw_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrcnw_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:

```verilog
// 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:

```vhdl
-- Physical RAM Instance without Pre Initialization

ram40_4kinst_physical : SB_RAM40_4K  
generic map (  
  READ_MODE => 0,  
  WRITE_MODE => 0  
)  
port map (  
  RDATA=>RDATA,  
  RADDR=>RADDR,  
  WADDR=>WADDR,  
  MASK=>MASK,  
  WDATA=>WDATA,  
  RCLKE=>RCLKE,  
  RCLK=>RCLK,  
  RE=>RE,  
  WCLKE=>WCLKE,  
  WCLK=>WCLK,  
  WE=>WE  
);
```

ICE Technology Library
Lattice Semiconductor Corporation Confidential
SB_RAM40_4KNR

Verilog Instantiation:

// Physical RAM Instance without Pre Initialization

SB_RAM40_4KNR ram40_4knrinst_physical (  
  .RDATA(RDATA),  
  .RADDR(RADDR),  
  .WADDR(WADDR),  
  .MASK(MASK),  
  .WDATA(WDATA),  
  .RCLKE(RCLKE),  
  .RCLKN(RCLKN),  
  .RE(RE),  
  .WCLKE(WCLKE),  
  .WCLK(WCLK),  
  .WE(WE)
);
defparam ram40_4knrinst_physical.READ_MODE=0;
defparam ram40_4knrinst_physical.WRITE_MODE=0;

VHDL Instantiation:

-- Physical RAM Instance without Pre Initialization

ram40_4knrinst_physical : SB_RAM40_4KNR  
generic map (  
  READ_MODE => 0,  
  WRITE_MODE=>0  
)  
port map (  
  RDATA=>RDATA,  
  RADDR=>RADDR,  
  WADDR=>WADDR,  
  MASK=>MASK,  
  WDATA=>WDATA,  
  RCLKE=>RCLKE,  
  RCLKN=>RCLKN,  
  RE=>RE,  
  WCLKE=>WCLKE,  
  WCLK=>WCLK,  
  WE=>WE  
);
SB_RAM40_4KNW

Verilog Instantiation:

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

defparam ram40_4knwinst_physical.READ_MODE=0;
defparam ram40_4knwinst_physical.WRITE_MODE=0;

VHDL Instantiation:

-- Physical RAM Instance without Pre Initialization
ram40_4knwinst_physical : SB_RAM40_4KNW  
generic map (  
  READ_MODE => 0,  
  WRITE_MODE=> 0  
)  
port map (  
  RDATA=>RDATA,  
  RADDR=>RADDR,  
  WADDR=>WADDR,  
  MASK=>MASK,  
  WDATA=>WDATA,  
  RCLKE=>RCLKE,  
  RCLK=>RCLK,  
  RE=>RE,  
  WCLKE=>WCLKE,  
  WCLKN=>WCLKN,  
  WE=>WE  
);
**SB_RAM40_4KNRNW**

Verilog Instantiation:

```verilog
// Physical RAM Instance without Pre Initialization

SB_RAM40_4KNRNW_instance_physical (  
  .RDATA(RDATA),  
  .RADDR(RADDR),  
  .WADDR(WADDR),  
  .MASK(MASK),  
  .WDATA(WDATA),  
  .RCLKE(RCLKE),  
  .RCLKN(RCLK),  
  .RE(RE),  
  .WCLKE(WCLKE),  
  .WCLKN(WCLKN),  
  .WE(WE)
);

defparam ram40_4knrnrwinst_physical.READ_MODE=0;
defparam ram40_4knrnrwinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

```vhdl
-- Physical RAM Instance without Pre Initialization

ram40_4knrnrwinst_physical : SB_RAM40_4KNRNW

generic map (  
  READ_MODE => 0,  
  WRITE_MODE => 0  
)

generic map (  
  RDATA=>RDATA,  
  RADDR=>RADDR,  
  WADDR=>WADDR,  
  MASK=>MASK,  
  WDATA=>WDATA,  
  RCLKE=>RCLKE,  
  RCLKN=>RCLK,  
  RE=>RE,  
  WCLKE=>WCLKE,  
  WCLKN=>WCLK,  
  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.

High Drive SB_IO
The iCE40LP1KHD and iCE40LP640HD device SB_IO’s 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 specify the “DRIVE_STRENGTH” synthesis attribute on the SB_IO instance and the IO needs to be configured as output-only registered IO. Refer iCEcube2_userguide for more details.
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 iCE40LP1KHD and iCE40640HD devices.

**Verilog Instantiation**

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

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

Input and Output functions are independently selectable via PIN_TYPE [1:0] and PIN_TYPE [5:2] 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_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**

defparam my_generic_IO.PIN_TYPE = 6'b(Output Pin Function, Input Pin Function);

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]

**Example**

defparam my_DDR_IO.PIN_TYPE = 6'b110000;  //PIN_TYPE[5:2] = 1100, PIN_TYPE[1:0] = 00

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

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

```verilog
SB_GB_IO My_Clock_Buffer_Package_Pin ( // A users external Clock reference
    .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
```

ICE Technology Library
Lattice Semiconductor Corporation Confidential
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
D_OUT_0 (d_out_0), // Data 0 – out to Pin/Rising clk
d_edge
.D_OUT_1 (d_out_1), // Data 1 – out to Pin/Falling clk
d_edge
.D_IN_0 (d_in_0), // Data 0 – Pin input/Rising clk
d_edge
.D_IN_1 (d_in_1) // Data 1 – Pin input/Falling clk
d_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 = Various;
// For details on PIN_TYPE and Pin Function
// Tables, refer to Section on SB_IO

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**

![](image)

**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.

.DUSER_SIGNAL_TO_GLOBAL_BUFFER (Users_internal_clk),
.GLOBAL_BUFFER_OUTPUT (Global_Buffered_User_Signal)
);
PLL Primitives

The Phase Lock Loop (PLL) function is offered as a feature in certain devices of the iCE65 and iCE40 device family.

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.

iCE65 PLL Primitives

There are 3 primitives that represent the PLL function in the iCEcube2 software viz. SB_PLL_CORE, SB_PLL_PAD, and SB_PLL_2_PAD. A short description of each primitive and its ports/parameters is provided in the following sections.

SB_PLL_CORE

The SB_PLL_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).

<table>
<thead>
<tr>
<th>Ports</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFERENCECLK</td>
<td>PLL source clock that serves as the input to the SB_PLL_CORE primitive.</td>
</tr>
<tr>
<td>PLLOUTGLOBAL</td>
<td>Output clock generated by the PLL, drives a global clock network on the FPGA.</td>
</tr>
<tr>
<td>PLLOUTCORE</td>
<td>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.</td>
</tr>
<tr>
<td>LOCK</td>
<td>Output port, when HIGH, indicates that the signal on PLLOUTGLOBAL/PLLOUTCORE is locked to the PLL source on REFERENCECLK.</td>
</tr>
<tr>
<td>EXTFEEDBACK</td>
<td>External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.</td>
</tr>
<tr>
<td>DYNAMICDELAY</td>
<td>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...</td>
</tr>
</tbody>
</table>
PLLOUTGLOBAL/PLLOUTCORE with respect to REFERENCECLK. 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 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_PLL_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</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 FIXED_DELAY_ADJUSTMENT 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>FIXED_DELAY_ADJUSTMENT</td>
<td>Sets a constant value for the Fine Delay Adjust Block.</td>
<td>0, 1,…,15</td>
<td>The PLLOUTGLOBAL &amp; PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FIXED_DELAY_ADJUSTMENT, only if the setting of the DELAY_ADJUSTMENT_MODE is FIXED.</td>
</tr>
<tr>
<td>PLL_OUT_PHASE</td>
<td>Controls the phase alignment of the PLLOUTCORE &amp; PLLOUTFGLOBAL signals relative to REFERENCECLK</td>
<td>NONE</td>
<td>No phase alignment. No duty cycle correction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0deg</td>
<td>0° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90deg</td>
<td>90° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180deg</td>
<td>180° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>270deg</td>
<td>270° 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>DIVF</td>
<td>Feedback divider</td>
<td>0,1,…,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>0,1,…,7</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</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_PLL_PAD**

The SB_PLL_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), and the source clock is not required inside the FPGA.

![Diagram of SB_PLL_PAD](image)

**Ports**

*PACKAGEPIN:* PLL source clock that serves as the input to the SB_PLL_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:* 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.

*RESET:* 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_PLL_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</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 FIXED_DELAY_ADJUSTMENT 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>FIXED_DELAY_ADJUSTMENT</td>
<td>Sets a constant value for the Fine Delay Adjust Block.</td>
<td>0, 1,..,15</td>
<td>The PLLOUTGLOBAL &amp; PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FIXED_DELAY_ADJUSTMENT, only if the setting of the DELAY_ADJUSTMENT_MODE is FIXED.</td>
</tr>
<tr>
<td>PLL_OUT_PHASE</td>
<td>Controls the phase alignment of the PLLOUTCORE &amp; PLLOUTGLOBAL signals relative to REFERENCECLK</td>
<td>NONE</td>
<td>No phase alignment. No duty cycle correction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0deg</td>
<td>0° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90deg</td>
<td>90° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180deg</td>
<td>180° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>270deg</td>
<td>270° 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>DIVF</td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User specified</td>
<td></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_PLL_2_PAD**

The SB_PLL_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), and in addition to the PLL output, the source clock is also required inside the FPGA.

**Ports**

*PACKAGEPIN*: PLL source clock that serves as the input to the SB_PLL_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 REFERENCECLK. 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 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_PLL_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</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 FIXED_DELAY_ADJUSTMENT 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>FIXED_DELAY_ADJUSTMENT</td>
<td>Sets a constant value for the Fine Delay Adjust Block.</td>
<td>0, 1,…,15</td>
<td>The PLLOUTGLOBAL &amp; PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FIXED_DELAY_ADJUSTMENT, only if the setting of the DELAY_ADJUSTMENT_MODE is FIXED.</td>
</tr>
<tr>
<td>PLL_OUT_PHASE</td>
<td>Controls the phase alignment of the PLLOUTCORE &amp; PLLOUTGLOBAL signals relative to REFERENCECLK</td>
<td>NONE</td>
<td>No phase alignment. No duty cycle correction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0deg</td>
<td>0° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90deg</td>
<td>90° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180deg</td>
<td>180° phase shift</td>
</tr>
<tr>
<td></td>
<td></td>
<td>270deg</td>
<td>270° 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>DIVF</td>
<td>Feedback divider</td>
<td>0,1,…63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>0,1,…..,7</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</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>Separate power-down controls for Port A and Port B outputs</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTB</td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
**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 Adjus 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 Adj 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>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 Adj 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 PLLOUTGLOBAL &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><strong>DIVF</strong></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><strong>DIVQ</strong></td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td><strong>FILTER_RANGE</strong></td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td><strong>EXTERNAL_DIVIDE_FACTOR</strong></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><strong>ENABLE_ICEGATE</strong></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.

![SB_PLL40_PAD Diagram](image)

### Ports

**PACKAGEPIN**: PLL source clock 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 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_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 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 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 additionally delayed 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→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 &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 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_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](image)

**Ports**

**PACKAGEPIN**: PLL source clock that serves as the input to the SB_PLL_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 REFERENCECLK. 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 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_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 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) \times 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) \times 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 -&gt; Divide by 4 \n1 -&gt; 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></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_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.

#### 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 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 REFERENCECLK. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.

RESETB: Active low input that asynchronously resets the PLL.

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

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

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

Parameters
The SB_PLL40_2F_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.
<table>
<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-&gt;Divide by 4 1-&gt;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</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
<td>Description</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>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 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>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 source clock 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 REFERENCECLK. The DYNAMICDELAY port controls are enabled when the DELAY_ADJUSTMENT_MODE parameter is set to DYNAMIC.
- **RESETB**: Active low input that asynchronously resets the PLL.
- **BYPASS**: Input signal, when asserted, connects the signal on REFERENCECLK to PLLOUTCORE/PLLOUTGLOBAL pins.
**LATCHINPUTVALUE**: Active high input, when enabled, forces the PLL into low-power mode. The PLLOUTGLOBALA/PLLOUTCOREA pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTA is set to ‘1’, and the LATCHINPUTVALUE signal is asserted. The PLLOUTGLOBALB/PLLOUTCOREB pins are held static at their last value only when the parameter ENABLE_ICEGATE_PORTB is set to ‘1’, and the LATCHINPUTVALUE signal is asserted.

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

**Parameters**
The SB_PLL40_2F_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</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>_FEEDBACK</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</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>_RELATIVE</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>
</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_PORTA    | Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports | SHIFTREG_0deg   | 0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” |
|                        |                                                  | SHIFTREG_90deg  | 90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTREG_DIV_MODE=0 |
|                        |                                                  | GENCLK          | The internally generated PLL frequency will be output to PortA. No phase shift. |
|                        |                                                  | GENCLK_HALF     | The internally generated PLL frequency will be divided by 2 and then output to PORTA. No phase shift. |</p>
<table>
<thead>
<tr>
<th><strong>PLLOUT_SELECT_PORTB</strong></th>
<th>Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports</th>
<th><strong>SHIFTREG_0deg</strong></th>
<th>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SHIFTREG_90deg</strong></td>
<td></td>
<td></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><strong>GENCLK</strong></td>
<td></td>
<td></td>
<td>The internally generated PLL frequency will be output to PortB. No phase shift.</td>
</tr>
<tr>
<td><strong>GENCLK_HALF</strong></td>
<td></td>
<td></td>
<td>The internally generated PLL frequency will be divided by 2 and then output to PORTB. No phase shift.</td>
</tr>
<tr>
<td><strong>DIVR</strong></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><strong>DIVF</strong></td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td></td>
</tr>
<tr>
<td><strong>DIVQ</strong></td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td><strong>FILTER_RANGE</strong></td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td><strong>EXTERNAL_DIVIDE_FACTOR</strong></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><strong>ENABLE_ICEGATE_PORTA</strong></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><strong>ENABLE_ICEGATE_PORTB</strong></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.

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

```verbatim
/* 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(CLKM) ) /* 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

```verilog
SB_LSOSC OSCInst0 (
    .ENACLKK(ENACLKK),
    .CLKK(CLK)
) /* 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.</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.</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.

```c
/* synthesis I2C_CLK_DIVIDER=[Divide Range] */

Divide Range : 0, 1, 2, 3 ... 1023. Default is 0.
```
Verilog Instantiation

SB_I2C i2cInst0 (  
.SBCLKI(sbclki),  
.SBRWI(sbwrwi),  
.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(sdati),  
.SBDATO7(sbdato[7]),  
.SBDATO6(sbdato[6]),  
.SBDATO5(sbdato[5]),  
.SBDATO4(sbdato[4]),  
.SBDATO3(sbdato[3]),  
.SBDATO2(sbdato[2]),  
.SBDATO1(sbdato[1]),  
.SBDATO0(sbdato[0]),  
.SBACKO(sbacko),  
.I2CIRQ(i2cirq),  
.I2CWKUP(i2cwkup),  
.SCLO(sclo),  
.SCLOE(scloe),  
.SDAO(sdao),  
.SDAOE(sdaoe)  
)/* synthesis I2C_CLK_DIVIDER= 1 */;
defparam i2cInst0.I2C_SLAVE_INIT_ADDR = "0b1111100001";
defparam i2cInst0.BUS_ADDR74 = "0b0001";

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.
### SB_SPI 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>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>Parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPI Primitive requires configuring a parameter for selecting the SPI IP location.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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 r 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>

<table>
<thead>
<tr>
<th>Synthesis Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

  /* synthesis SPI_CLK_DIVIDER= [Divide Range] */

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

SB_SPI sspinst0 ( .SBCLKI(sbclki),
    .SBRWI(sbrwi),
    .SBSTBI(sbstbi),
    .SBADRI7(sbadri[7]),
    .SBADRI6(sbadri[6]),
    .SBADRI5(sbadri[5]),
    .SBADRI4(sbadri[4]),
    .SBADRI3(sbadri[3]),
    .SBADRI2(sbadri[2]),
    .SBADRI1(sbadri[1]),
    .SBADRI0(sbadri[0]),
    .SBDATI7(sbdati[7]),
    .SBDATI6(sbdati[6]),
    .SBDATI5(sbdati[5]),
    .SBDATI4(sbdati[4]),
    .SBDATI3(sbdati[3]),
    .SBDATI2(sbdati[2]),
    .SBDATI1(sbdati[1]),
    .SBDATI0(sbdati[0]),
    .MI(mi),
    .SI(si),
    .SCKI(scki),
    .SCSNI(scsn),
    .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 sspinst0.BUS_ADDR74 = "0b0000";
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.

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

    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  
                                );