LATTICE
ICE™
Technology Library

Version 2.6
April 12, 2014

Version	Changes
2.0	Added Version Number to document. Added sections on Default Signal Values for unconnected ports. Modified
2.1	Added PLL primitives
2.2	Corrected SB_CARRY connections to LUT inputs
2.3	Added iCE40 RAM, PLL primitives.
2.4	Added PLL_DS, SB_MIPI_RX_2LANE, SB_TMDS_deserializer primitives.
2.5	Added SB_MAC16 Primitive details.
2.6	Added iCE40LM Hard Macro details. Removed PLL_DS, SB_MIPI, SB_TMDS, SB_MAC16 primitive details.

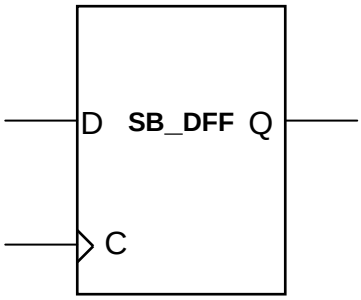
Table of Contents

Register Primitives

SB_DFF

D Flip-Flop

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



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

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

HDL use

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

Verilog Instantiation

// SB_DFF - D Flip-Flop.

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

// End of SB_DFF instantiation

VHDL Instantiation

-- SB_DFF - D Flip-Flop.

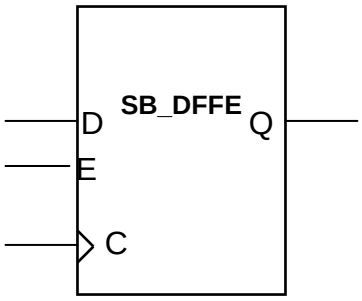
```
SB_DFF_inst: SB_DFF
      port map (
      Q => Q,           -- Registered Output
      C => C,           -- Clock
      D => D,           -- Data
      );
```

-- End of SB_DFF instantiation

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

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input E: Logic '1'

Note that explicitly connecting a logic '1' value to port E 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

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

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

// End of SB_DFFE instantiation

VHDL Instantiation

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

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

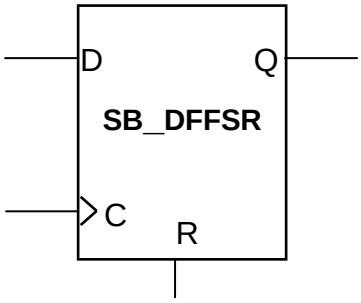
-- End of SB_DFFE instantiation

SB_DFFSR

D Flip-Flop with Synchronous Reset

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

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



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

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFSR instantiation

VHDL Instantiation

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

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

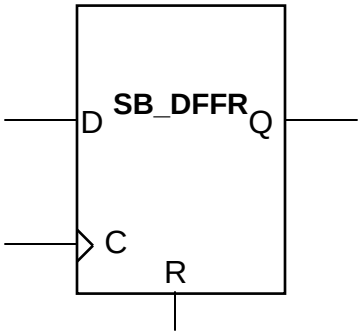
-- End of SB_DFFSR instantiation

SB_DFFR

D Flip-Flop with Asynchronous Reset

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

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



Inputs			Output
R	**D**	**C**	**Q**
1	X	X	0
0	0	↗	0
0	1	↗	1
Power on State	X	X	0

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'

Verilog Instantiation

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

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

// 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
      );
```

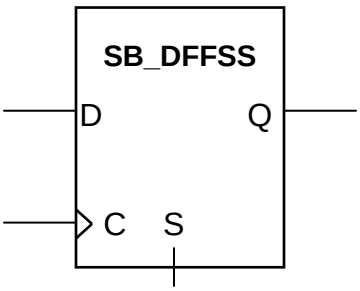
-- End of SB_DFFR instantiation

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.



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

Key

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFSS instantiation

VHDL Instantiation

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

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

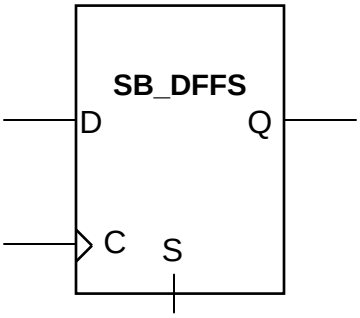
-- End of SB_DFFSS instantiation

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.



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

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFS instantiation

VHDL Instantiation

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

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

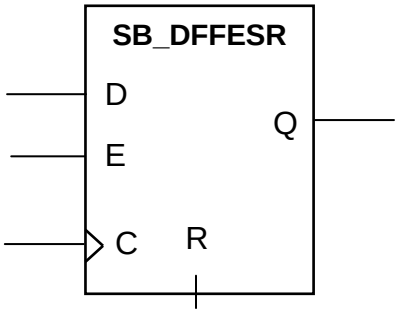
-- End of SB_DFFS instantiation

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.



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

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'
Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E 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

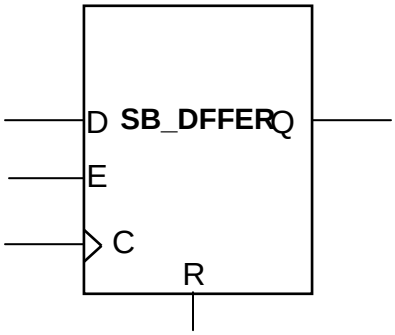
```
-- SB_DFFESR - D Flip-Flop, Reset is synchronous with rising clock edge
-- Clock Enable.

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.



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

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'
Input E: Logic '1'

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

Verilog Instantiation

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

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

// End of SB_DFFER instantiation

VHDL Instantiation

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

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

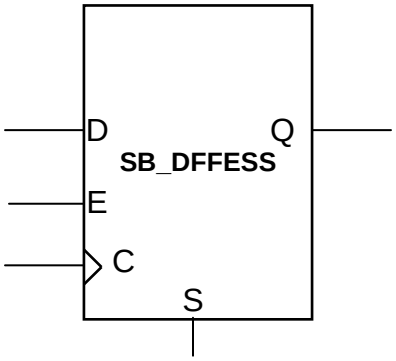
-- End of SB_DFFER instantiation

SB_DFFESS

D Flip-Flop with Clock Enable and Synchronous Set

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

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



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

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

HDL Usage

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

Default Signal Values

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'
Input S: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFESS instantiation

VHDL Instantiation

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

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

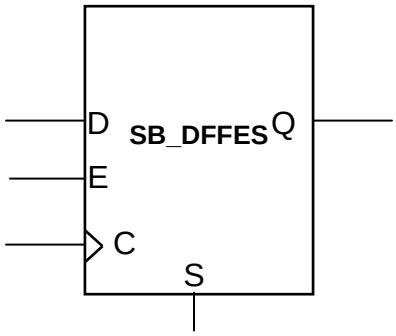
-- End of SB_DFFESS instantiation

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.



Inputs				Output
S	**E**	**D**	**CLK**	**Q**
1	X	X	X	1
0	0	X	X	Previous **Q**
0	1	0	↗	0
0	1	1	↗	1
Power on State	X	X	X	0

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'
Input E: Logic '1'

Verilog Instantiation

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

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

// End of SB_DFFES instantiation

VHDL Instantiation

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

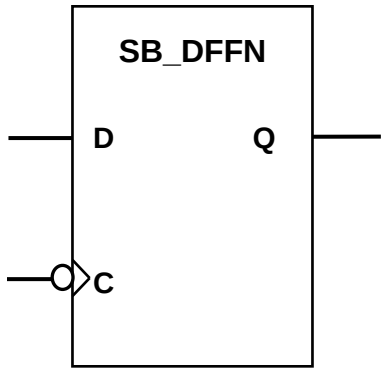
```
SB_DFFES_inst : SB_DFFES
      port map (
      Q => Q,           -- Registered Output
      C => C,           -- Clock
      E => E,                 -- Clock Enable
      D => D,           -- Data
      S => S                  -- Asynchronously Set
      );
```

-- End of SB_DFFES instantiation

SB_DFFN

D Flip-Flop – Negative Edge Clock

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



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFN instantiation

VHDL Instantiation

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

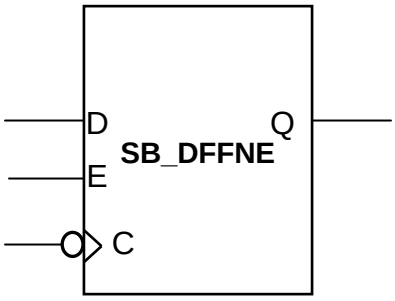
```
SB_DFFN_inst : SB_DFFN
      port map (
      Q => Q,           -- Registered Output
      C => C,           -- Clock
      D => D,           -- Data
      );
```

-- End of SB_DFFN instantiation

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.



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E 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_DFFNE - D Flip-Flop – Negative Edge Clock and Clock Enable.

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

```
      .E(E),            // Clock Enable
      );
```

// End of SB_DFFNE instantiation

VHDL Instantiation

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

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

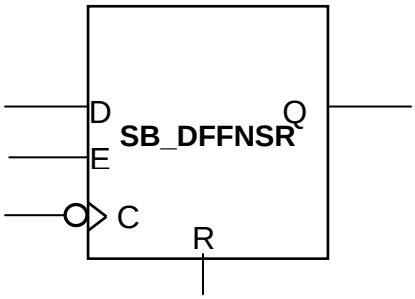
-- End of SB_DFFNE instantiation

SB_DFFNSR

D Flip-Flop – Negative Edge Clock with Synchronous Reset

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

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



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'

Verilog Instantiation

// SB_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
      );
```

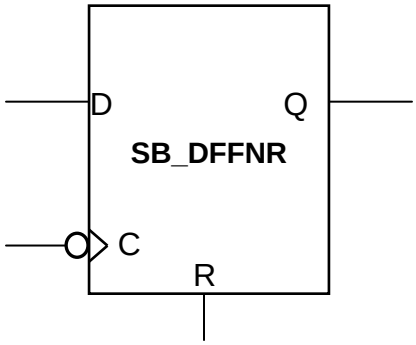
-- End of SB_DFFNSR instantiation

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.



Inputs			Output
R	**D**	**CLK**	**Q**
1	X	X	0
0	0	↘	0
0	1	↘	1
Power on State	X	X	0

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'

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

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

// End of SB_DFFNR instantiation

VHDL Instantiation

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

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

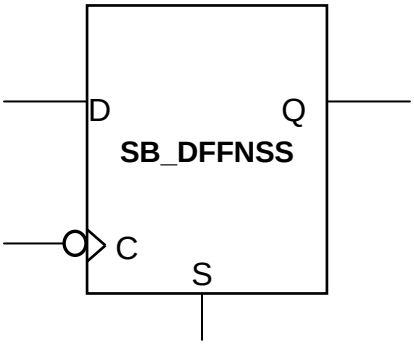
-- End of SB_DFFNR instantiation

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.



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFNSS instantiation
```

VHDL Instantiation

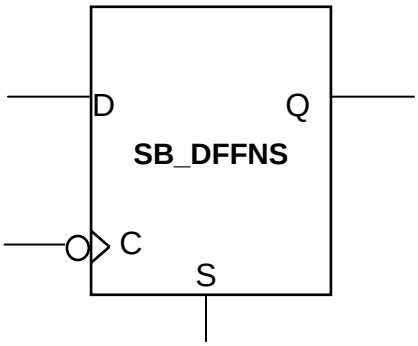
```
-- SB_DFFNSS - D Flip-Flop – Negative Edge Clock, Set is synchronous with the falling clock edge,
-- with .

        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.



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'

Verilog Instantiation

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

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

// End of SB_DFFNS instantiation

VHDL Instantiation

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

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

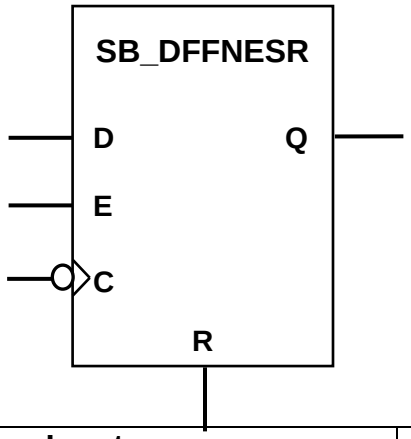
-- End of SB_DFFNS instantiation

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.



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input R: Logic '0'
Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E 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_DFFNESR - D Flip-Flop – Negative Edge Clock, Reset is synchronous with falling clock edge Clock Enable.

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

// End of SB_DFFNESR instantiation

VHDL Instantiation

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

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

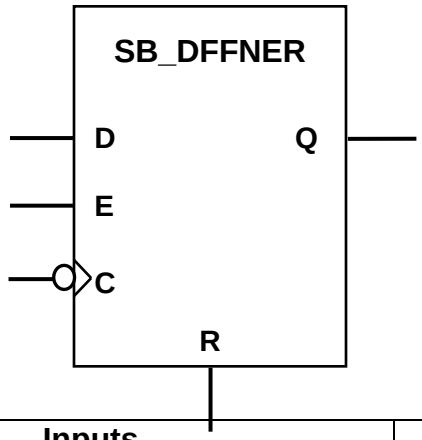
-- End of SB_DFFNESR instantiation

SB_DFFNER

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

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

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



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

Key
↘ 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

```
// 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

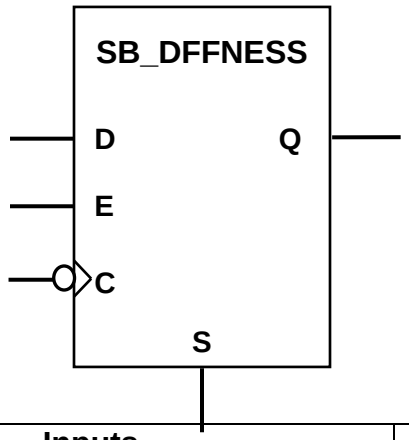
```
-- SB_DFFNER - D Flip-Flop – Negative Edge Clock, Reset is asynchronously
-- on falling clock edge and Clock Enable.

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.



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'
Input E: Logic '1'

Note that explicitly connecting a Logic '1' value to port E 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

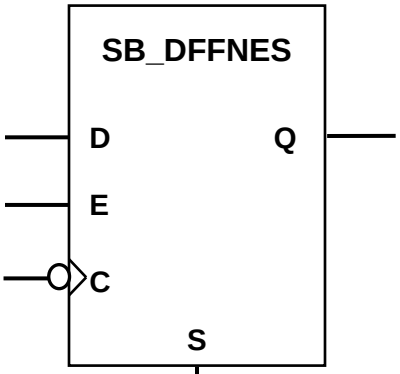
```
-- SB_DFFNESS - D Flip-Flop – Negative Edge Clock, Set is synchronous with falling clock edge,
-- and Clock Enable.

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.



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

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

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

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

Input D: Logic '0'
Input C: Logic '0'
Input S: Logic '0'
Input E: Logic '1'

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

Verilog Instantiation

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

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

// End of SB_DFFNES instantiation
```

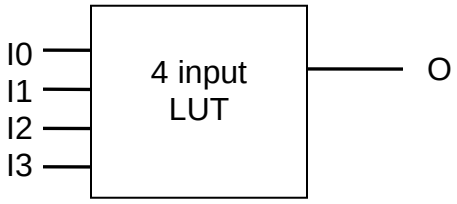
VHDL Instantiation

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

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

-- End of SB_DFFNES instantiation
```

Combinational Logic Primitives

SB_LUT4

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



Initialization values

LUT state initialization parameter LUT_INIT = 16'hxxxx;

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

HDL Usage

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

Default Signal Values

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

Verilog Instantiation

```
// SB_LUT4 : 4-input Look-Up Table

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

 //End of SB_LUT4 instantiation
```

VHDL Instantiation

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

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

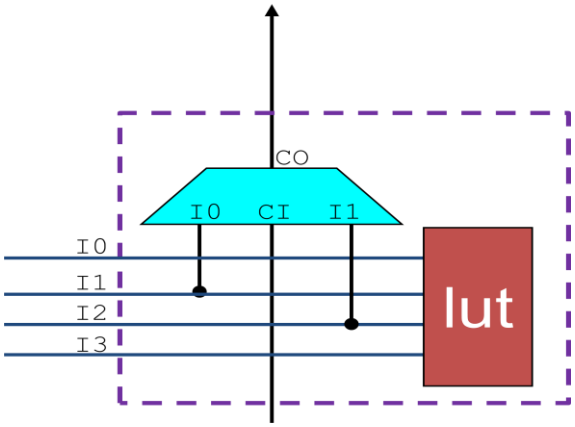
SB_CARRY

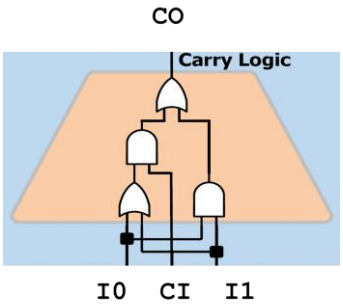
Carry Logic

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





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

HDL Usage

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

Default Signal Values

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

Verilog Instantiation

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

VHDL Instantiation

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

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.

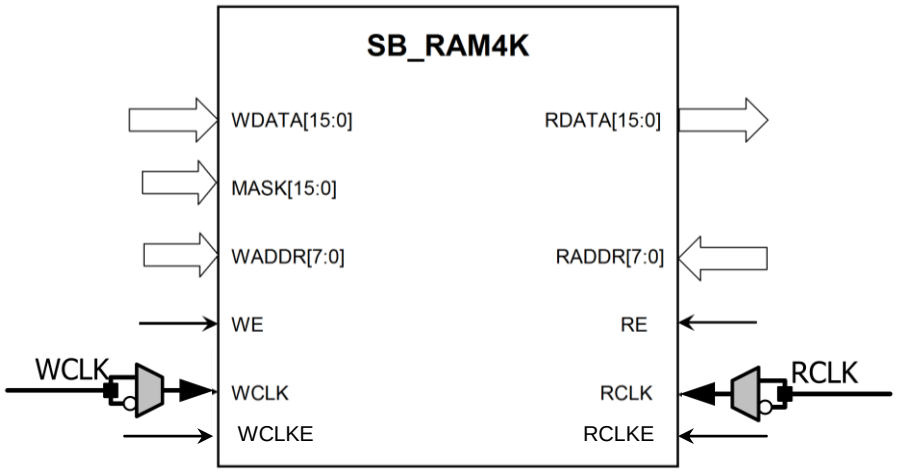
iCE65 Block RAM

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.

RAM Primitive Name	Description
SB_RAM4K	Posedge Read clock, Posedge Write clock
SB_RAM4KNR	Negedge Read clock, Posedge Write clock
SB_RAM4KNW	Posedge Read clock, Negedge Write clock
SB_RAM4KNRNW	Negedge Read clock, Negedge Write clock



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 unitialized 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:

RAM4K Block RAM Signals		
Signal Name	**Direction**	**Description**
WDATA[15:0]	Input	Write Data input
MASK[15:0]	Input	Bit-line Write Enable input, active low
WADDR[7:0]	Input	Write Address input. Selects up to 256 possible locations
WE	Input	Write Enable input, active high
WCLK	Input	Write Clock input, rising-edge active
WCLKE	Input	Write Clock Enable input
RDATA[15:0]	Output	Read Data output
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible locations
RE	Input	Read Enable input, active high
RCLK	Input	Read Clock input, rising-edge active
RCLKE	Input	Read Clock Enable input
INIT_0, … …,INIT_F	Verilog parameter	RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16x256=4096 total bits)

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

Operation	WDATA[15:0] Data	MASK[15:0] Bit Enable	WADDR[7:0] Address	WE Enable	WCLK Clock	RAM Location
Disabled	X	X	X	X	0	No Change
Disabled	X	X	X	0	X	No Change
Write Data	D[i]	MASK[i]=0	WADDR	1	↑	RAM[WADDR[i]] = D[i]
Masked Write	X	MASK[i]=1	WADDR	1	↑	RAM[WADDR[i]] = No Change

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.

Operation	RADDR[7:0] Address	RE Enable	RCLK Clock	RDATA[15:0]
After configuration, before first valid Read Data operation	X	X	X	Undefined
Disabled	X	0	X	No Change
Read Data	RA	1	↑	RAM[RADDR]

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   SB_RAM4K_with_INIT  (.RDATA(RDATA), .RCLK(RCLK), .RCLKE(RCLKE),
.RE(RE), .RADDRRADDR), .WCLK(WCLK), .WCLKE(WCLKE), .WE(WE), .WADDR(WADDR),
.MASK(MASK), .WDATA(WDATA));

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

- - SB_RAM4K with data initialization after power on reset

```
SB_RAM4K_with_INIT : SB_RAM4K
```

```
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 //Posedge 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_RAM4KNRNW // 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.

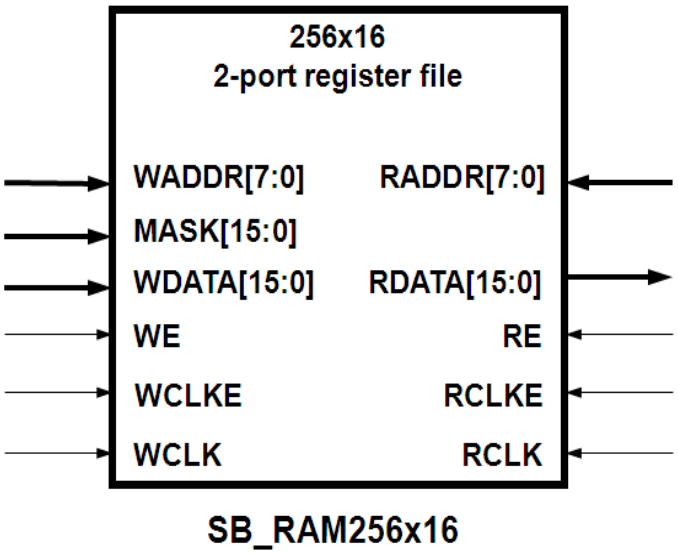
The following table lists the supported dual port synchronous RAM configurations, each of 4Kbits in size. The RAM blocks can be directly instantiated in the top module and taken through iCube2 flow.

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

The 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.

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

SB_RAM256x16



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

SB_RAM256x16NW

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

Verilog Instantiation:

```
SB_RAM256x16NW   ram256x16nw_inst (
      .RDATA(RDATA_c[15:0]),
      .RADDR(RADDR_c[7:0]),
      .RCLK(RCLK_c),
      .RCLKE(RCLKE_c),
      .RE(RE_c),
      .WADDR(WADDR_c[7:0]),
      .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"
      )
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, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

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

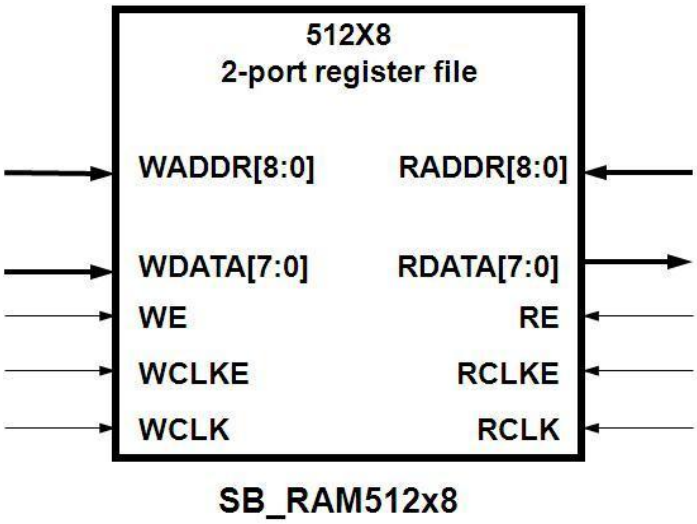
```
      .WE(WE_c),
      .MASK(MASK_c[15:0])
);
defparam ram256x16nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nrnw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram256x16nrnw_inst :  SB_RAM256x16NRNW
generic map (
INIT_0 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
x"0000000000000000000000000000000000000000000000000000000000000000",
```

```
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLKN => RCLKN_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        MASK  => MASK_c,
        WE => WE_c
      );
```

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

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:

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

SB_RAM512x8NW

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

Verilog Instantiation:

```
SB_RAM512x8NW   ram512x8nw_inst (
      .RDATA(RDATA_c[7:0]),
      .RADDR(RADDR_c[8:0]),
```

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

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

VHDL Instantiation:

```
ram512x8nw_inst :  SB_RAM512x8NW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
INIT_8 =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLK => RCLK_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

SB_RAM512x8NRNW

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

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

defparam ram512x8nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

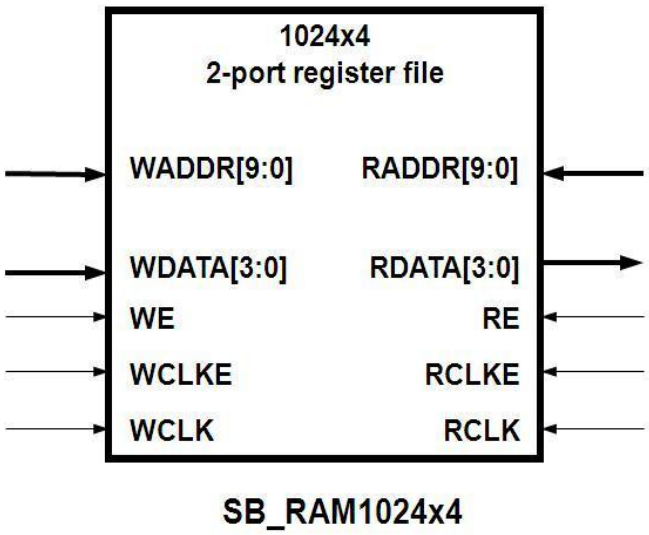
```
defparam ram512x8nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;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

VHDL Instantiation:

```
ram512x8nrnw_inst :  SB_RAM512x8NRNW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLKN => RCLKN_c,
```

```
    RCLKE => RCLKE_c,
    RE => RE_c,
    WADDR => WADDR_c,
    WCLKN=> WCLKN_c,
    WCLKE => WCLKE_c,
    WDATA => WDATA_c,
    WE => WE_c
  );
```

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:

```
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:

```
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:

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

VHDL Instantiation:

```
ram1024x4nw_inst :  SB_RAM1024x4NW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
```

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

SB_RAM1024x4NRNW

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

Verilog Instantiation:

```
SB_RAM1024x4NRNW   ram1024x4nrnw_inst (
      .RDATA(RDATA_c[3:0]),
      .RADDR(RADDR_c[9:0]),
      .RCLKN(RCLK_c),
      .RCLKE(RCLKE_c),
      .RE(RE_c),
      .WADDR(WADDR_c[3:0]),
      .WCLKN(WCLK_c),
      .WCLKE(WCLKE_c),
      .WDATA(WDATA_c[9:0]),
      .WE(WE_c)
);
defparam ram1024x4nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

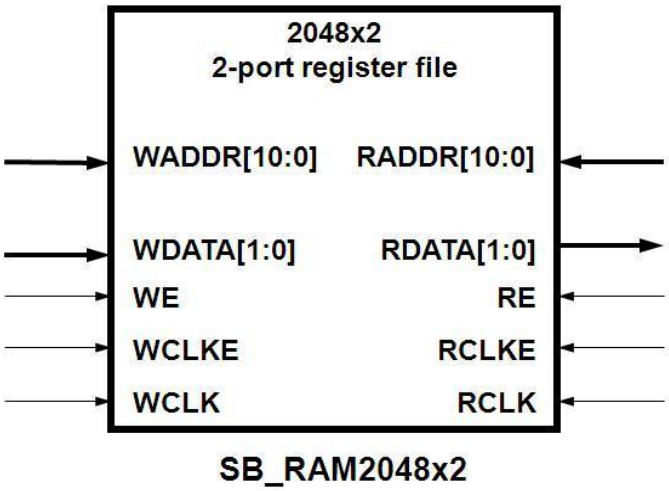
```
defparam ram1024x4nrnw_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4nrnw_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram1024x4nrnw_inst :  SB_RAM1024x4NRNW
generic map (
INIT_0 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
x"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E =>
x"0000000000000000000000000000000000000000000000000000000000000000",
```

```
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLKN => RCLKN_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

SB_RAM2048x2



The following modules are the complete list of SB_RAM2048x2 based primitives

SB_RAM2048x2

SB_RAM2048x2 **//Posedge clock RCLK WCLK**
(RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM2048x2   ram2048x2_inst (
      .RDATA(RDATA_c[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 => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLK=> WCLK_c,
        WCLKE => WCLKE_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,
        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_RAM2048x2NW

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

```
SB_RAM2048x2NW   ram2048x2nw_inst (
      .RDATA(RDATA_c[2:0]),
      .RADDR(RADDR_c[10:0]),
      .RCLK(RCLK_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 ram2048x2nw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram2048x2nw_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nw_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

```
ram2048x2nw_inst :  SB_RAM2048x2NW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
INIT_E =
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
      )
port map (
        RDATA => RDATA_c,
        RADDR => RADDR_c,
        RCLK => RCLK_c,
        RCLKE => RCLKE_c,
        RE => RE_c,
        WADDR => WADDR_c,
        WCLKN=> WCLKN_c,
        WCLKE => WCLKE_c,
        WDATA => WDATA_c,
        WE => WE_c
      );
```

SB_RAM2048x2NRNW

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

```
SB_RAM2048x2NRNW   ram2048x2nrnw_inst (
      .RDATA(RDATA_c[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 ram2048x2nrnw_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram2048x2nrnw_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2nrnw_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation:

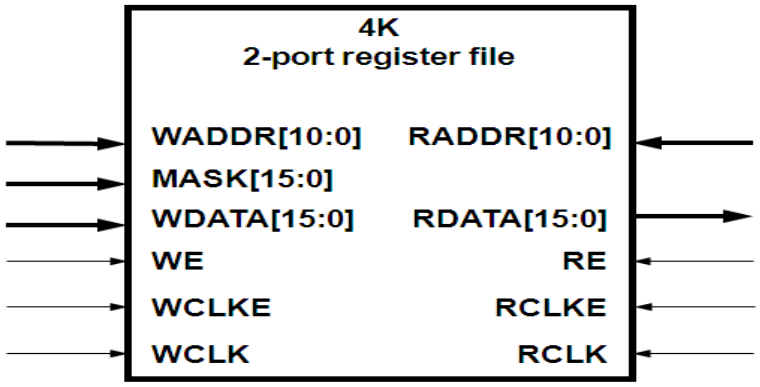
```
ram2048x2nrnw_inst :  SB_RAM2048x2NRNW
generic map (
INIT_0 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 =>
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

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



The following table lists the signals for both ports.

SB_RAM40_4K RAM Port Signals		
Signal Name	**Direction**	**Description**
WDATA[15:0]	Input	Write Data input
MASK[15:0]*	Input	Bit-line Write Enable input, active low. Applicable only when WRITE_MODE parameter is set to 0.
WADDR[7:0]	Input	Write Address input. Selects up to 256 possible locations
WE	Input	Write Enable input, active high
WCLK	Input	Write Clock input, rising-edge active
WCLKE	Input	Write Clock Enable input
RDATA[15:0]	Output	Read Data output
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible locations
RE	Input	Read Enable input, active high
RCLK	Input	Read Clock input, rising-edge active
RCLKE	Input	Read Clock Enable input

Parameter Name	Description	Parameter Value	Configuration
INIT_0, … …,INIT_F	RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16x256=4096 total bits)	INIT_0 to INIT_F	Initialize the RAM with predefined value
WRITE_MODE	Sets the RAM block write port configuration	0	256x16
		1	512x8
		2	1024x4
		3	2048x2
READ_MODE	Sets the RAM block read port configuration	0	256x16
		1	512x8
		2	1024x4
		3	2048x2

SB_RAM40_4K

Verilog Instantiation:

// Physical RAM Instance without Pre Initialization

```
SB_RAM40_4K ram40_4kinst_physical (
            .RDATA(RDATA),
            .RADDR(RADDR),
            .WADDR(WADDR),
            .MASK(MASK),
            .WDATA(WDATA),
            .RCLKE(RCLKE),
            .RCLK(RCLK),
            .RE(RE),
            .WCLKE(WCLKE),
            .WCLK(WCLK),
            .WE(WE)
      );
defparam ram40_4kinst_physical.READ_MODE=0;
defparam ram40_4kinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

-- Physical RAM Instance without Pre Initialization

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

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:

```
// Physical RAM Instance without Pre Initialization

SB_RAM40_4KNRNW ram40_4knrnwinst_physical (
            .RDATA(RDATA),
            .RADDR(RADDR),
            .WADDR(WADDR),
            .MASK(MASK),
            .WDATA(WDATA),
            .RCLKE(RCLKE),
            .RCLKN(RCLKN),
            .RE(RE),
            .WCLKE(WCLKE),
            .WCLKN(WCLKN),
            .WE(WE)
      );
defparam ram40_4knrnwinst_physical.READ_MODE=0;
defparam ram40_4knrnwinst_physical.WRITE_MODE=0;
```

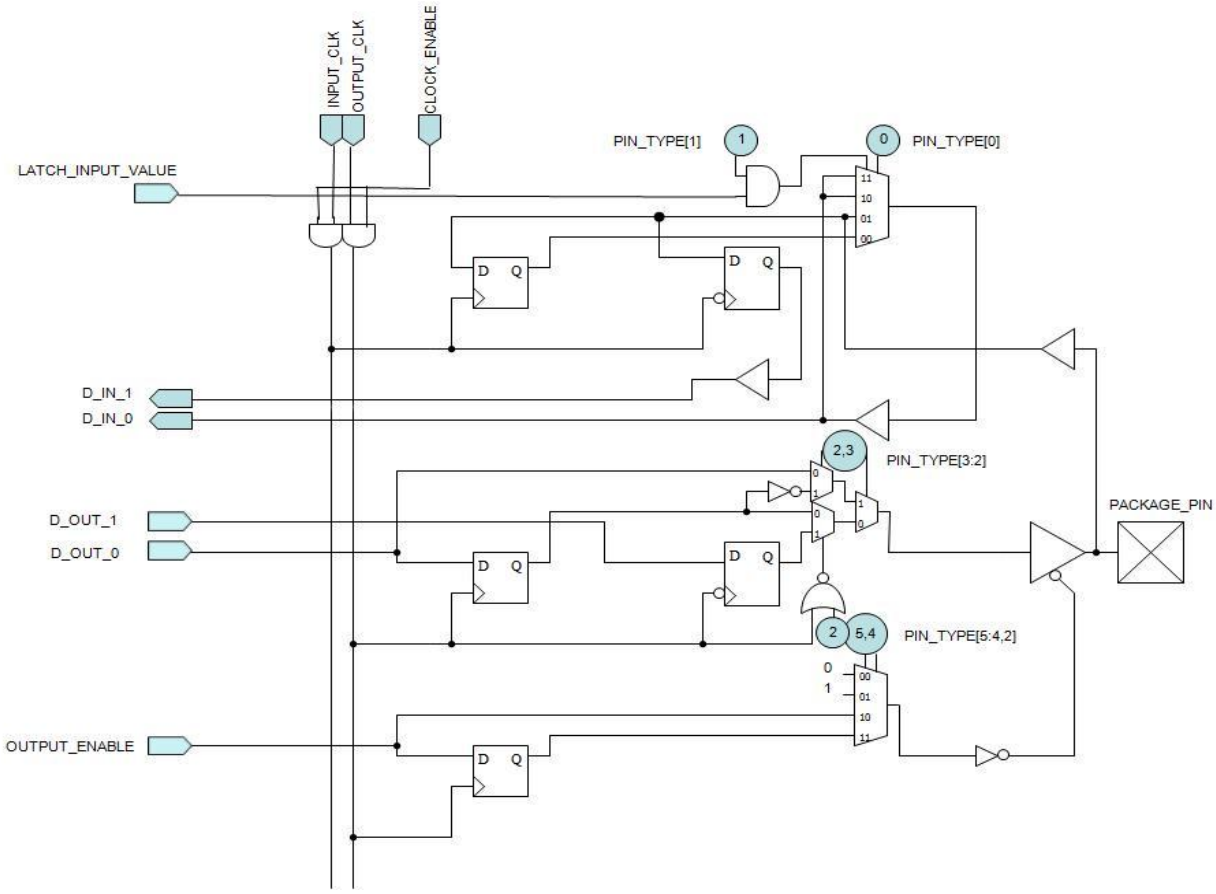
VHDL Instantiation:

```
-- Physical RAM Instance without Pre Initialization

ram40_4knrnwinst_physical :  SB_RAM40_4KNRNW
generic map (
      READ_MODE => 0,
      WRITE_MODE= >0
                  )
port map  (
      RDATA=>RDATA,
      RADDR=>RADDR,
      WADDR=>WADDR,
      MASK=>MASK,
      WDATA=>WDATA,
      RCLKE=>RCLKE,
      RCLKN=>RCLKN,
      RE=>RE,
      WCLKE=>WCLKE,
      WCLKN=>WCLKN,
      WE=>WE
      );
```

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.



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:

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

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

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

Syntax Verilog Use

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

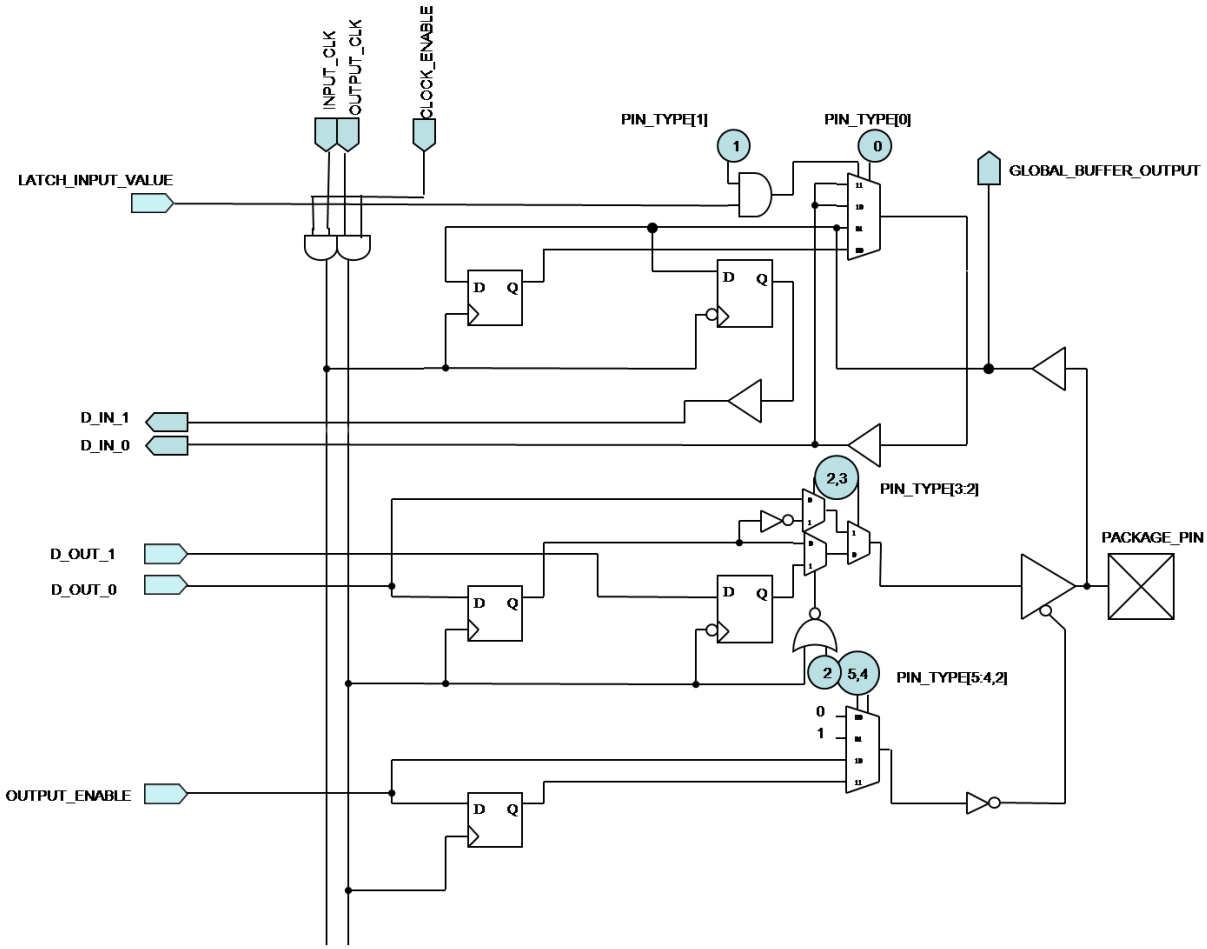
defparam my_DDR_IO.PIN_TYPE = 6'b110000; //PIN_TYPE[5:2] = 1100, PIN_TYPE[1:0] = 00

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.

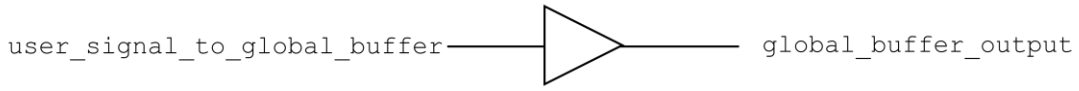
Verilog Instantiation

```
SB_GB_IO My_Clock_Buffer_Package_Pin (     // A users external Clock reference
                                              pin
.PACKAGE_PIN (Package_Pin),                // User's Pin signal name
.LATCH_INPUT_VALUE (latch_input_value),    // Latches/holds the Input value
.CLOCK_ENABLE (clock_enable),              // Clock Enable common to input and
                                           // output clock
```

```
.INPUT_CLK (input_clk),                    // Clock for the input registers
.OUTPUT_CLK (output_clk),                  // Clock for the output registers
.OUTPUT_ENABLE (output_enable),            // Output Pin Tristate/Enable
                                           // control
.D_OUT_0 (d_out_0),                        // Data 0 - out to Pin/Rising clk
                                           // edge
.D_OUT_1 (d_out_1),                        // Data 1 - out to Pin/Falling clk
                                           // edge
.D_IN_0 (d_in_0),                          // Data 0 - Pin input/Rising clk
                                           // edge
.D_IN_1 (d_in_1)                           // Data 1 - Pin input/Falling clk
                                           // edge

.GLOBAL_BUFFER_OUTPUT (Global_Buffered_User_Clock)
                                           // Example use - clock buffer
                                           //driven from the input pin
);

defparam  My_Clock_Buffer_Package_Pin.PIN_TYPE = 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



Verilog Instantiation

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

.USER_SIGNAL_TO_GLOBAL_BUFFER (Users_internal_Clk),
.GLOBAL_BUFFER_OUTPUT ( Global_Buffered_User_Signal)

);
```

PLL Primitives

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

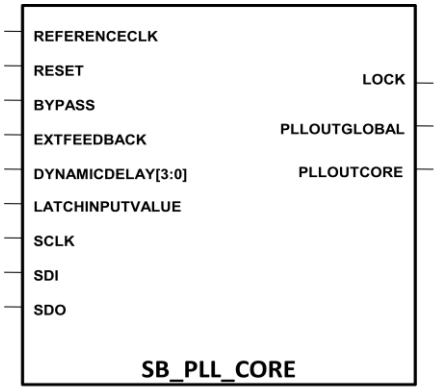
It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

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).



Ports

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

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

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

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

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

DYNAMICDELAY: 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.

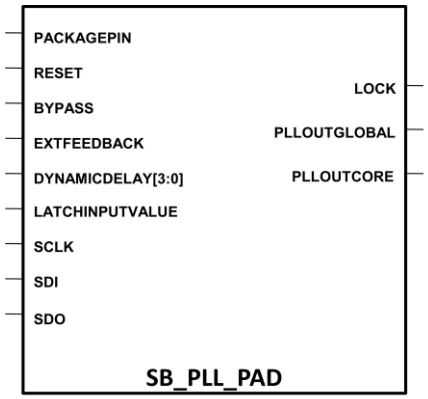
Parameters

The SB_PLL_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.

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE	Selects the mode for the Fine Delay Adjust block.	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FIXED_DELAY_ADJUSTMENT parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FIXED_DELAY_ADJUSTMENT	Sets a constant value for the Fine Delay Adjust Block.	0, 1,…,15	The PLLOUTGLOBAL & 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.
PLL_OUT_PHASE	Controls the phase alignment of the PLLOUTCORE & PLLOUTGLOBAL signals relative to REFERENCECLK	NONE	No phase alignment. No duty cycle correction
		0deg	0^o phase shift
		90deg	90^o phase shift
		180deg	180^o phase shift
		270deg	270^o phase shift
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	0,1,…,7	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

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.



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 PLLOUTLGOBAL port.

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

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

DYNAMICDELAY: 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.

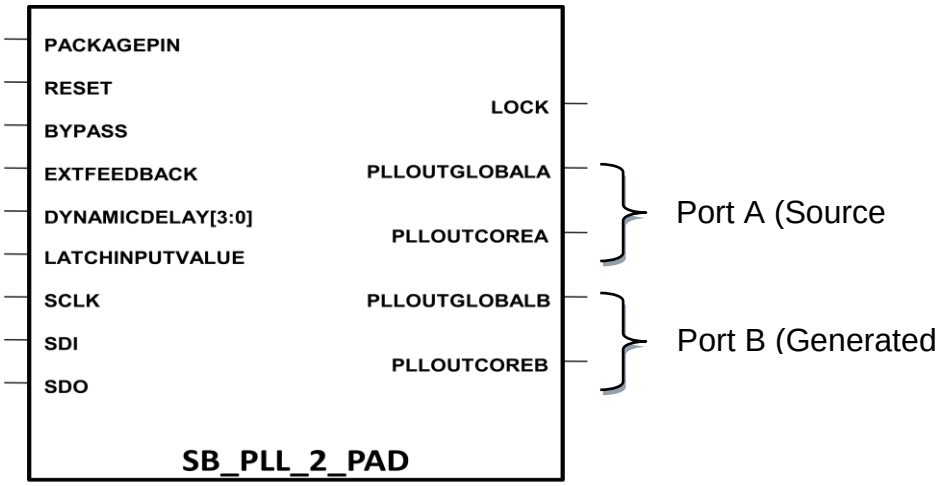
Parameters

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.

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE	Selects the mode for the Fine Delay Adjust block.	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FIXED_DELAY_ADJUSTMENT parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FIXED_DELAY_ADJUSTMENT	Sets a constant value for the Fine Delay Adjust Block.	0, 1,…,15	The PLLOUTGLOBAL & 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.
PLL_OUT_PHASE	Controls the phase alignment of the PLLOUTCORE & PLLOUTGLOBAL signals relative to REFERENCECLK	NONE	No phase alignment. No duty cycle correction
		0deg	0^o phase shift
		90deg	90^o phase shift
		180deg	180^o phase shift
		270deg	270^o phase shift
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	0,1,…,7	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

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 PLLOUTLGOBAL port.

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

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

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

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE	Selects the mode for the Fine Delay Adjust block.	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FIXED_DELAY_ADJUSTMENT parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FIXED_DELAY_ADJUSTMENT	Sets a constant value for the Fine Delay Adjust Block.	0, 1,…,15	The PLLOUTGLOBAL & 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.
PLL_OUT_PHASE	Controls the phase alignment of the PLLOUTCORE & PLLOUTGLOBAL signals relative to REFERENCECLK	NONE	No phase alignment. No duty cycle correction
		0deg	0^o phase shift
		90deg	90^o phase shift
		180deg	180^o phase shift
		270deg	270^o phase shift
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	0,1,…,7	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE_PORTA/ ENABLE_ICEGATE_PORTB	Separate power-down controls for Port A and Port B outputs	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

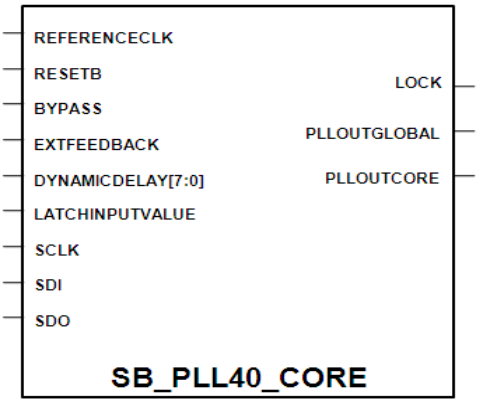
iCE40 PLL Primitives

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).



Ports

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

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

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

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

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

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

RESETB: Active low input that asynchronously resets the PLL.

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

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

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

Parameters

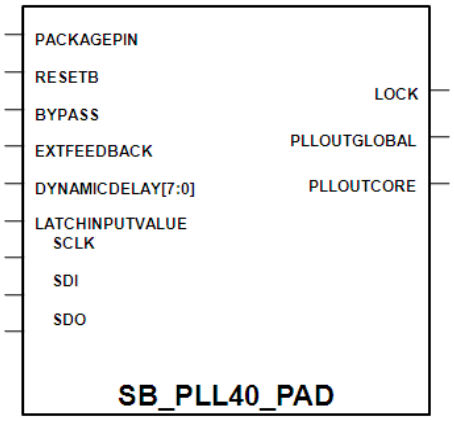
The SB_PLL40_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL & PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA & PLLOUTCOREA signals are additionally delayed by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0→Divide by 4 1→Divide by 7
PLLOUT_SELECT	Selects the signal to be output at the PLLOUTCORE and PLLOUTGLOBAL ports	SHIFTREG_0deg	0^{o} phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90^{o} 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 without any phase shift.
		GENCLK_HALF	The internally generated PLL frequency will be divided by 2 and then output. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to

DIVF	Feedback divider	0,1,..,63	control the output frequency, depending on the FEEDBACK_PATH setting.
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

SB_PLL40_PAD

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



Ports

PACKAGEPIN: PLL 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 PLLOUTLGOBAL port.

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

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

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

RESETB: Active low input that asynchronously resets the PLL.

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

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

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

Parameters

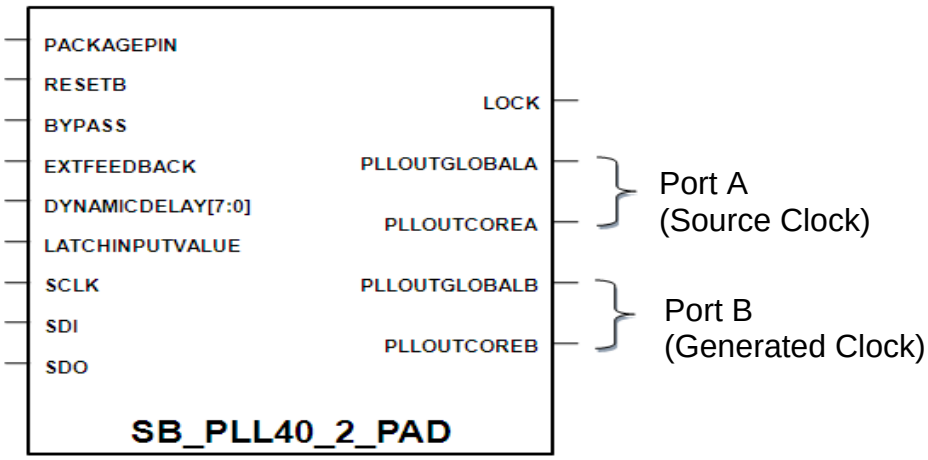
The SB_PLL40_PAD primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL & PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA & PLLOUTCOREA signals are additionally delayed by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0→Divide by 4 1→Divide by 7
PLLOUT_SELECT	Selects the signal to be output at the PLLOUTCORE and PLLOUTGLOBAL ports	SHIFTREG_0deg	0^{o} phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90^{o} 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 without any phase shift.
		GENCLK_HALF	The internally generated PLL frequency will be divided by 2 and then output. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to

DIVF	Feedback divider	0,1,..,63	control the output frequency, depending on the FEEDBACK_PATH setting.
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

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.



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 PLLOUTLGOBAL port.

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

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

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

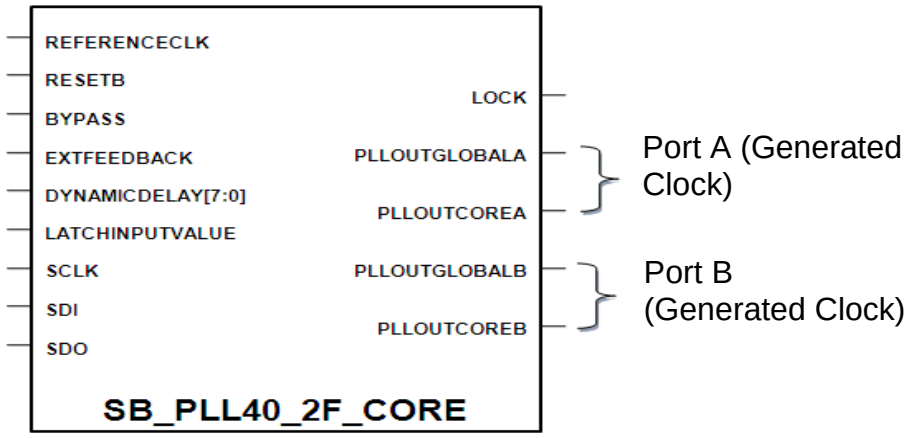
The SB_PLL40_2_PAD primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL & PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA & PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0→Divide by 4 1→Divide by 7
PLLOUT_SELECT_PORTB	Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports	SHIFTREG_0deg	0^o phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90^o 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 PortB. No phase shift.
		GENCLK_HALF	The internally generated PLL frequency will be divided by 2 and then output to PORTB. No phase shift.

DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO Divider	1,2,…,6	
FILTER_RANGE	PLL Filter Range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified value. Default 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE_PORTA	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input
ENABLE_ICEGATE_PORTB	Enables the PLL power-down control	0	Power-down control disabled
		1	Power-down controlled by LATCHINPUTVALUE input

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 PLLOUTLGOBALA port.

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

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

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

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

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

RESETB: Active low input that asynchronously resets the PLL.

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

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

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

Parameters

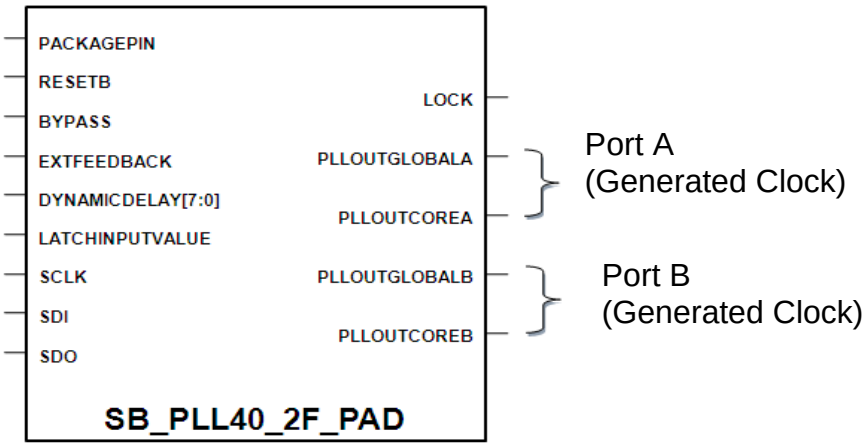
The SB_PLL40_2F_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

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

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 PLLOUTLGOBALA port.

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

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

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

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

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

RESETB: Active low input that asynchronously resets the PLL.

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

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

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

Parameters

The SB_PLL40_2F_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.

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block
		EXTERNAL	Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBALA & PLLOUTCOREA signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA & PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0→Divide by 4 1→Divide by 7
PLLOUT_SELECT_PORTA	Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports	SHIFTREG_0deg	0^o phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”
		SHIFTREG_90deg	90^o 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.

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

Hard Macro Primitives

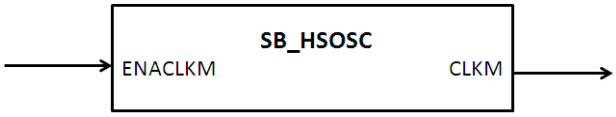
iCE40LM Hard Macros

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

SB_HSOSC Ports		
Signal Name	**Direction**	**Description**
ENACLKM	Input	Enable High Speed Strobe Generator. Active High.
CLKM	Output	Strobe Generator Output (12Mhz).

Clock Network Selection

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

Synthesis Attribute

/* synthesis ROUTE_THROUGH_FABRIC=<value> */

Value:

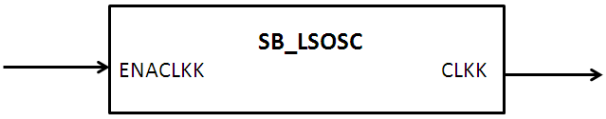
0: Use dedicated clock network. Default option.
1: Use fabric routes.

Verilog Instantiation

```
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.



Ports

SB_LSOSC Ports		
Signal Name	**Direction**	**Description**
ENACLKK	Input	Enable Low Power Strobe Generator. Active High.
CLKK	Output	Strobe Generator Output (10Khz).

Clock Network Selection

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

Synthesis Attribute:

/* synthesis ROUTE_THROUGH_FABRIC=<value> */

Value:

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

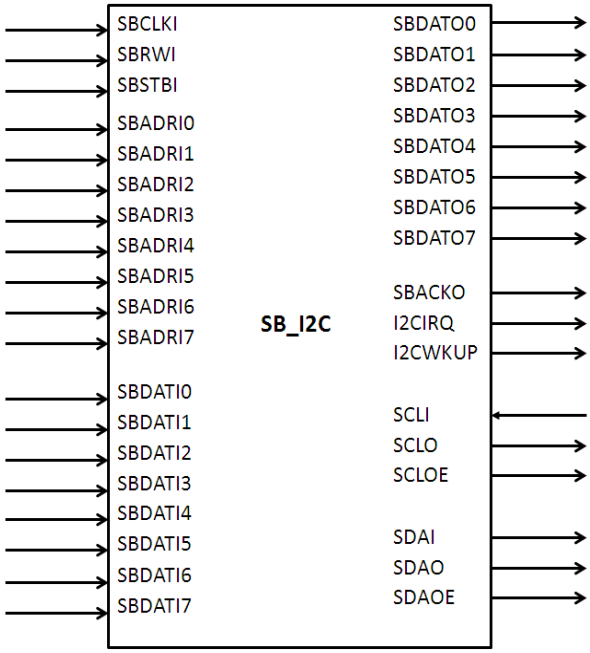
Verilog Instantiation

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

SB_I2C

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

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



Ports

SB_I2C Ports		
Signal Name	**Direction**	**Description**
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Output	System Data Output. Bit 0.
SBDATO1	Output	System Data Output. Bit 1.
SBDATO2	Output	System Data Output. Bit 2.
SBDATO3	Output	System Data Output. Bit 3.
SBDATO4	Output	System Data Output. Bit 4.
SBDATO5	Output	System Data Output. Bit 5.
SBDATO6	Output	System Data Output. Bit 6.
SBDATO7	Output	System Data Output. Bit 7.

SBACKO	Output	System Acknowledgement.
I2CIRQ	Output	I2C Interrupt output.
I2CWKUP	Output	I2C Wake Up from Standby signal.
SCLI	Input	Serial Clock Input.
SCLO	Output	Serial Clock Output
SCLOE	Output	Serial Clock Output Enable. Active High.
SDAI	Input	Serial Data Input
SDAO	Output	Serial Data Output
SDAOE	Output	Serial Data Output Enable. Active High.

Parameters

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

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

Synthesis Attribute

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

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

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

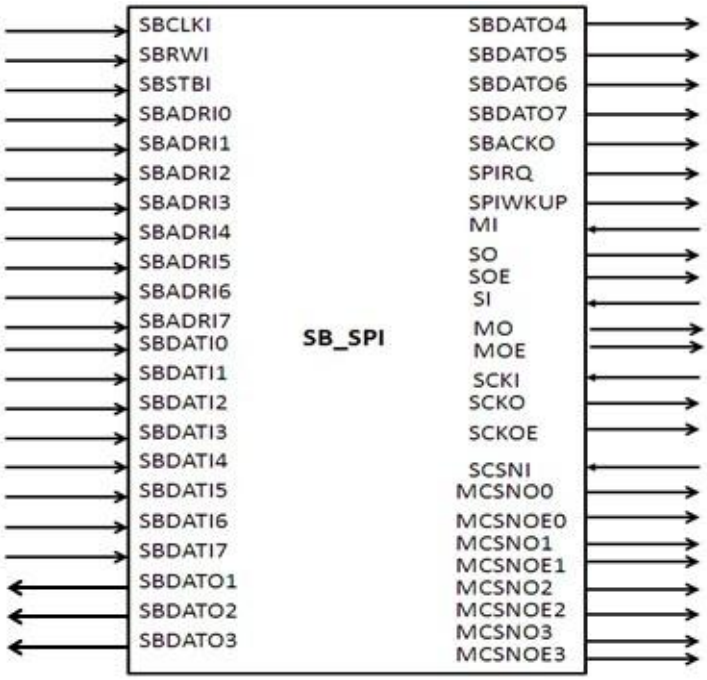
Verilog Instantiation

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

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

SB_SPI

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

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



Ports

SB_SPI Ports		
Signal Name	**Direction**	**Description**
SBCLKI	Input	System Clock input.
SBRWI	Input	System Read/Write Input.
SBSTBI	Input	Strobe Signal
SBADRI0	Input	System Bus Control registers address. Bit 0.
SBADRI1	Input	System Bus Control registers address. Bit 1.
SBADRI2	Input	System Bus Control registers address. Bit 2.
SBADRI3	Input	System Bus Control registers address. Bit 3.
SBADRI4	Input	System Bus Control registers address. Bit 4.
SBADRI5	Input	System Bus Control registers address. Bit 5.
SBADRI6	Input	System Bus Control registers address. Bit 6.
SBADRI7	Input	System Bus Control registers address. Bit 7.
SBDATI0	Input	System Data Input. Bit 0.
SBDATI1	Input	System Data input. Bit 1.
SBDATI2	Input	System Data input. Bit 2.
SBDATI3	Input	System Data input. Bit 3.
SBDATI4	Input	System Data input. Bit 4.
SBDATI5	Input	System Data input. Bit 5.
SBDATI6	Input	System Data input. Bit 6.
SBDATI7	Input	System Data input. Bit 7.
SBDATO0	Input	System Data Output. Bit 0.
SBDATO1	Input	System Data Output. Bit 1.
SBDATO2	Input	System Data Output. Bit 2.
SBDATO3	Input	System Data Output. Bit 3.
SBDATO4	Input	System Data Output. Bit 4.
SBDATO5	Input	System Data Output. Bit 5.
SBDATO6	Input	System Data Output. Bit 6.

SBDATO7	Input	System Data Output. Bit 7.
SBACKO	Output	System Acknowledgement
SPIIRQ	Output	SPI Interrupt output.
SPIWKUP	Output	SPI Wake Up from Standby signal.
MI	Input	Master Input from PAD
SO	Output	Slave Output to PAD
SOE	Output	Slave Output Enable to PAD. Active High.
SI	Input	Slave Input from PAD
MO	Output	Master Output to PAD
MOE	Output	Master Output Enable to PAD. Active High
SCKI	Input	Slave Clock Input From PAD
SCKO	Output	Slave Clock Output to PAD
SCKOE	Output	Slave Clock Output Enable to PAD. Active High.
SCSNI	Input	Slave Chip Select Input From PAD
MCSNO0	Output	Master Chip Select Output to PAD. Line 0.
MCSNO1	Output	Master Chip Select Output to PAD. Line 1.
MCSNO2	Output	Master Chip Select Output to PAD. Line 2.
MCSNO3	Output	Master Chip Select Output to PAD. Line 3.
MCSNOE0	Output	Master Chip Select Output Enable to PAD. Active High. Line 0.
MCSNOE1	Output	Master Chip Select Output Enable to PAD. Active High. Line 1
MCSNOE2	Output	Master Chip Select Output Enable to PAD. Active High. Line 2
MCSNOE3	Output	Master Chip Select Output Enable to PAD. Active High. Line 3

Parameters

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

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

Synthesis Attribute

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

/* synthesis SPI_CLK_DIVIDER= [Divide Range] */

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

Verilog Instantiation

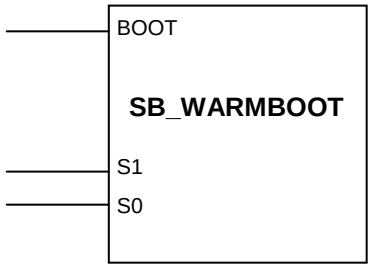
```
SB_SPI spiInst0 (
    .SBCLKI(sbclki),
    .SBRWI(sbrwi),
    .SBSTBI(sbstbi),
    .SBADRI7(sbadri[7]),
    .SBADRI6(sbadri[6]),
    .SBADRI5(sbadri[5]),
    .SBADRI4(sbadri[4]),
    .SBADRI3(sbadri[3]),
    .SBADRI2(sbadri[2]),
    .SBADRI1(sbadri[1]),
    .SBADRI0(sbadri[0]),
    .SBDATI7(sbdati[7]),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

defparam spiInst0.BUS_ADDR74 = "0b0000";
```

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
                              // configuration image
      .S0 (my_sel0)
      );
```