iCE40 Technology Library

Technical Note

FPGA-TN-02026-3.2

November 2021
Disclaimers
Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer’s responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice’s product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.
Contents

Acronyms in This Document ........................................................................................................... 9

1. Register Primitives .................................................................................................................. 10
   1.1. SB_DFF – D Flip-Flop ........................................................................................................ 10
   1.2. SB_DFFE – D Flip-Flop with Clock Enable .................................................................... 11
   1.3. SB_DFFSR – D Flip-Flop with Synchronous Reset ......................................................... 13
   1.4. SB_DFFR – D Flip-Flop with Asynchronous Reset ......................................................... 14
   1.5. SB_DFFSS – D Flip-Flop with Synchronous Set .............................................................. 15
   1.6. SB_DFFS – D Flip-Flop with Asynchronous Set ............................................................. 17
   1.7. SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset ...................... 18
   1.8. SB_DIFF – D Flip-Flop with Clock Enable and Asynchronous Reset ......................... 19
   1.9. SB_DFFESS – D Flip-Flop with Clock Enable and Synchronous Set ............................. 21
   1.10. SB_DFFES – D Flip-Flop with Clock Enable and Asynchronous Set ............................ 22
   1.11. SB_DFFN – D Flip-Flop – Negative Edge Clock ............................................................ 24
   1.12. SB_DFFNE – D Flip-Flop – Negative Edge Clock and Clock Enable ......................... 25
   1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset ............ 27
   1.14. SB_DFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset ............. 29
   1.15. SB_DFFNS – D Flip-Flop – Negative Edge Clock with Synchronous Set ................. 30
   1.16. SB_DFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set ............... 32
   1.17. SB_DFFNESR – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Reset .. 33
   1.18. SB_DFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset ... 35
   1.19. SB_DFFNESS – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Set .... 37
   1.20. SB_DFFNES – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Set ...... 39

2. Combinational Logic Primitives .............................................................................................. 41
   2.1. SB_LUT4 ......................................................................................................................... 41
   2.2. SB_CARRY .................................................................................................................... 42

3. iCE40 Block RAM Primitives ................................................................................................. 44
   3.1. SB_RAM256x16 .............................................................................................................. 45
      3.1.1. SB_RAM256x16 ...................................................................................................... 45
      3.1.2. SB_RAM256x16NR ............................................................................................... 47
      3.1.3. SB_RAM256x16NW ............................................................................................. 49
      3.1.4. SB_RAM256x16NRNW ....................................................................................... 51
   3.2. SB_RAM512x8 ................................................................................................................ 53
      3.2.1. SB_RAM512x8 ...................................................................................................... 53
      3.2.2. SB_RAM512x8NR ............................................................................................... 55
      3.2.3. SB_RAM512x8NW ............................................................................................. 57
      3.2.4. SB_RAM512x8NRNW ....................................................................................... 59
   3.3. SB_RAM1024x4 .............................................................................................................. 60
      3.3.1. SB_RAM1024x4 .................................................................................................... 61
      3.3.2. SB_RAM1024x4NR ............................................................................................. 63
      3.3.3. SB_RAM1024x4NW ............................................................................................ 65
      3.3.4. SB_RAM1024x4NRNW ...................................................................................... 67
   3.4. SB_RAM2048x2 .............................................................................................................. 68
      3.4.1. SB_RAM2048x2 .................................................................................................... 69
      3.4.2. SB_RAM2048x2NR ............................................................................................. 71
      3.4.3. SB_RAM2048x2NW ............................................................................................ 73
      3.4.4. SB_RAM2048x2NRNW ...................................................................................... 75
   3.5. SB_RAM40_4K ............................................................................................................... 77
      3.5.1. SB_RAM40_4KNR ............................................................................................... 79
      3.5.2. SB_RAM40_4KNW ............................................................................................. 80
      3.5.3. SB_RAM40_4KRNW ........................................................................................... 81

4. I/O Primitives ........................................................................................................................... 82
   4.1. SB_IO .............................................................................................................................. 82
4.1.1. DDR I/O Configuration ................................................................. 84
4.1.2. High Drive SB_IO ........................................................................ 84
4.1.3. Pull Up Resistor Configuration .................................................. 84
5. Global Buffer Primitives ................................................................. 86
  5.1. SB_GB_IO ..................................................................................... 86
  5.2. SB_GB Primitive ................................................................. 88
6. PLL Primitives ............................................................................. 89
  6.1. SB_PLL40_CORE ......................................................................... 89
  6.2. SB_PLL40_PAD ........................................................................... 91
  6.3. SB_PLL40_2_PAD ....................................................................... 94
  6.4. SB_PLL40_2F_CORE ............................................................... 97
  6.5. SB_PLL40_2F_PAD ................................................................. 100
7. Hard Macro Primitives ................................................................. 103
  7.1. iCE40LM Hard Macros .............................................................. 103
    7.1.1. SB_HSOSC (For HSSG) ...................................................... 103
    7.1.2. SB_LSOSC (For LPSG) ...................................................... 104
    7.1.3. SB_I2C ............................................................................... 105
    7.1.4. SB_SPI .............................................................................. 108
  7.2. iCE40 Ultra (iCE5LP) Hard Macros .......................................... 111
    7.2.1. SB_HFOSC .......................................................................... 111
    7.2.2. SB_LFOSC ......................................................................... 112
    7.2.3. SB_LED_DRV_CUR ............................................................ 113
    7.2.4. SB_RGB_DRV .................................................................... 114
    7.2.5. SB_IR_DRV ........................................................................ 116
    7.2.6. SB_RGB_IP ........................................................................ 117
    7.2.7. SB_IO_OD .......................................................................... 118
    7.2.8. SB_I2C ............................................................................... 120
    7.2.9. SB_SPI .............................................................................. 123
    7.2.10. SB_MAC16 ................................................................. 125
  7.3. iCE40 UltraLite (ICE40UL) Hard Macros .................................... 137
    7.3.1. SB_HFOSC .......................................................................... 137
    7.3.2. SB_LFOSC ......................................................................... 138
    7.3.3. SB_RGBA_DRV ................................................................. 139
    7.3.4. SB_IR400_DRV ................................................................. 141
    7.3.5. SB_BARCODE_DRV ........................................................... 143
    7.3.6. SB_IR500_DRV ................................................................. 144
    7.3.7. SB_LEDDA_IP ................................................................. 146
    7.3.8. SB_IR_IP ........................................................................... 147
    7.3.9. SB_IO_OD .......................................................................... 149
    7.3.10. SB_I2C_FIFO ................................................................. 152
  7.4. iCE40 UltraPlus Hard Macros .................................................. 154
    7.4.1. SB_HFOSC .......................................................................... 154
    7.4.2. SB_LFOSC ......................................................................... 156
    7.4.3. SB_RGBA_DRV ................................................................. 157
    7.4.4. SB_LEDDA_IP ................................................................. 159
    7.4.5. SB_IO_OD .......................................................................... 160
    7.4.6. SB_SPI .............................................................................. 165
    7.4.7. SB_MAC16 ................................................................. 167
    7.4.8. SB_SRAM256KA ............................................................... 168
    7.4.9. SB_IO_I3C ................................................................. 169
  7.5. Device Configuration Primitives .............................................. 173
    7.5.1. SB_WARMBOOT .............................................................. 173
Technical Support Assistance .......................................................... 174
Revision History ................................................................................ 175
Figures

Figure 1.1. SB_DFF – D Flip-Flop .......................................................... 10
Figure 1.2. SB_DFFE – D Flip-Flop with Clock Enable .......................... 11
Figure 1.3. SB_DFFSR – D Flip-Flop with Synchronous Reset ................. 13
Figure 1.4. SB_DFFR – D Flip-Flop with Asynchronous Reset .................. 14
Figure 1.5. SB_DFFSS – D Flip-Flop with Synchronous Set ...................... 15
Figure 1.6. SB_DFFS – D Flip-Flop with Asynchronous Set .................... 17
Figure 1.7. SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset .......................................................... 18
Figure 1.8. SB_DIFFER – D Flip-Flop with Clock Enable and Asynchronous Reset .......................................................... 19
Figure 1.9. SB_DIFFESS – D Flip-Flop with Clock Enable and Synchronous Set .......................................................... 21
Figure 1.10. SB_DFFES – D Flip-Flop with Clock Enable and Asynchronous Set .......................................................... 22
Figure 1.11. SB_DFFN – D Flip-Flop – Negative Edge Clock ..................... 24
Figure 1.12. SB_DFFNE – D Flip-Flop – Negative Edge Clock and Clock Enable .......................................................... 25
Figure 1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset .......................................................... 27
Figure 1.14. SB_DFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset .......................................................... 29
Figure 1.15. SB_DFFNSS – D Flip-Flop – Negative Edge Clock with Synchronous Set .......................................................... 30
Figure 1.16. SB_DFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set .......................................................... 32
Figure 1.17. SB_DFFNESR – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Reset .......................................................... 33
Figure 1.18. SB_DFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset .......................................................... 35
Figure 1.19. SB_DFFNESS – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Set .......................................................... 37
Figure 1.20. SB_DFFNES – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Set .......................................................... 39
Figure 2.1. SB_LUT4 ........................................................................... 41
Figure 2.2. Carry Logic Structure within a Logic Cell .................................. 43
Figure 3.1. SB_RAM256x16 ................................................................. 45
Figure 3.2. SB_RAM512x8 ................................................................. 53
Figure 3.3. SB_RAM1024x4 ................................................................. 60
Figure 3.4. SB_RAM2048x2 ................................................................. 68
Figure 3.5. SB_RAM40_4K .................................................................. 77
Figure 4.1. SB_IO ............................................................................. 82
Figure 5.1. SB_GB_IO ..................................................................... 86
Figure 5.2. SB_GB Primitive .............................................................. 88
Figure 6.1. SB_PLL40_CORE ............................................................. 89
Figure 6.2. SB_PLL40_PAD ............................................................... 91
Figure 6.3. SB_PLL40_2_PAD ............................................................ 94
Figure 6.4. SB_PLL40_2F_CORE ....................................................... 97
Figure 6.5. SB_PLL40_2F_PAD .......................................................... 100
Figure 7.1. SB_HSOSC .................................................................... 103
Figure 7.2. SB_LSOSC .................................................................... 104
Figure 7.3. SB_I2C ........................................................................ 105
Figure 7.4. SB_SPI ........................................................................ 108
Figure 7.5. SB_HFOSC .................................................................... 111
Figure 7.6. SB_LFOSC .................................................................... 112
Figure 7.7. SB_LED_DRV_CUR ........................................................ 113
Figure 7.8. SB_RGB_DRV ................................................................. 114
Figure 7.9. SB_IR_DRV .................................................................... 116
Figure 7.10. SB_RGB_IP ................................................................. 117
Figure 7.11. SB_IO_OD ................................................................. 118
Figure 7.12. SB_I2C ................................................................. 120
Figure 7.13. SB_SPI ................................................................. 123
Figure 7.14. SB_MAC16 ................................................................. 125
Figure 7.15. SB_MAC16 ................................................................. 126
Figure 7.16. SB_HFOSC ................................................................. 137
Figure 7.17. SB_LFOSC ................................................................. 138
Figure 7.18. SB_RGBA_DRV .......................................................... 139
Figure 7.19. SB_IR400_DRV .............................................................. 141
Figure 7.20. SB_BARCODE_DRV ........................................................ 143
Figure 7.21. SB_IR500_DRV .............................................................. 144
Figure 7.22. SB_LEDDA_IP ............................................................... 146
Figure 7.23. SB_IR_IP .................................................................. 147
Figure 7.24. SB_IO_OD .............................................................. 150
Figure 7.25. SB_I2C_FIFO ................................................................. 152
Figure 7.26. SB_HFOSC ................................................................. 154
Figure 7.27. SB_LFOSC ................................................................. 156
Figure 7.28. SB_RGBA_DRV .......................................................... 157
Figure 7.29. SB_LEDDA_IP ............................................................... 159
Figure 7.30. SB_IO_OD .............................................................. 161
Figure 7.31. SB_I2C .............................................................. 163
Figure 7.32. SB_SPI .............................................................. 165
Figure 7.33. SB_MAC16 ................................................................. 167
Figure 7.34. SB_SPRAM256KA ...................................................... 168
Figure 7.35. SB_IO_I3C ................................................................. 169
Figure 7.36. SB_WARMBOOT ......................................................... 173
### Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.1</td>
<td>SB_OFF – D Flip-Flop</td>
</tr>
<tr>
<td>Table 1.2</td>
<td>SB_DFFE – D Flip-Flop with Clock Enable</td>
</tr>
<tr>
<td>Table 1.3</td>
<td>SB_DFFSR – D Flip-Flop with Synchronous Reset</td>
</tr>
<tr>
<td>Table 1.4</td>
<td>SB_DFFR – D Flip-Flop with Asynchronous Reset</td>
</tr>
<tr>
<td>Table 1.5</td>
<td>SB_DFFSS – D Flip-Flop with Synchronous Set</td>
</tr>
<tr>
<td>Table 1.6</td>
<td>SB_DFFS – D Flip-Flop with Asynchronous Set</td>
</tr>
<tr>
<td>Table 1.7</td>
<td>SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset</td>
</tr>
<tr>
<td>Table 1.8</td>
<td>SB_DIFFER – D Flip-Flop with Clock Enable and Asynchronous Reset</td>
</tr>
<tr>
<td>Table 1.9</td>
<td>SB_DFFESS – D Flip-Flop with Clock Enable and Synchronous Set</td>
</tr>
<tr>
<td>Table 1.10</td>
<td>SB_DIFFES – D Flip-Flop with Clock Enable and Asynchronous Set</td>
</tr>
<tr>
<td>Table 1.11</td>
<td>SB_DIFFN – D Flip-Flop – Negative Edge Clock</td>
</tr>
<tr>
<td>Table 1.12</td>
<td>SB_DIFFNE – D Flip-Flop – Negative Edge Clock and Clock Enable</td>
</tr>
<tr>
<td>Table 1.13</td>
<td>SB_DIFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset</td>
</tr>
<tr>
<td>Table 1.14</td>
<td>SB_DIFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset</td>
</tr>
<tr>
<td>Table 1.15</td>
<td>SB_DIFFNSS – D Flip-Flop – Negative Edge Clock with Synchronous Set</td>
</tr>
<tr>
<td>Table 1.16</td>
<td>SB_DIFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set</td>
</tr>
<tr>
<td>Table 1.17</td>
<td>SB_DIFFNESR – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Reset</td>
</tr>
<tr>
<td>Table 1.18</td>
<td>SB_DIFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset</td>
</tr>
<tr>
<td>Table 1.19</td>
<td>SB_DIFFNESS – D Flip-Flop – Negative Edge Clock, Enable and Synchronous Set</td>
</tr>
<tr>
<td>Table 1.20</td>
<td>SB_DIFFNES – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Set</td>
</tr>
<tr>
<td>Table 2.1</td>
<td>SB_LUT4</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>SB_CARRY</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>iCE40 Block RAM</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>256 x 16 RAM Block Configuration</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>SB_RAM40_4K Naming Convention Rules</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>SB_RAM40_4K RAM Port Signals</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>SB_RAM40_4K RAM Parameters</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Input Pin Function</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Output Pin Function</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Drive Value</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Pull-up Resistor Value</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>SB_PLL40_CORE Parameters</td>
</tr>
<tr>
<td>Table 6.2</td>
<td>SB_PLL40_PAD Parameters</td>
</tr>
<tr>
<td>Table 6.3</td>
<td>SB_PLL40_2_PAD Parameters</td>
</tr>
<tr>
<td>Table 6.4</td>
<td>SB_PLL40_2F_CORE Parameters</td>
</tr>
<tr>
<td>Table 6.5</td>
<td>SB_PLL40_2F_PAD Parameters</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>SB_HOSCP Port Signals</td>
</tr>
<tr>
<td>Table 7.2</td>
<td>SB_LFOSC Port Signals</td>
</tr>
<tr>
<td>Table 7.3</td>
<td>SB_I2C Port Signals</td>
</tr>
<tr>
<td>Table 7.4</td>
<td>SB_I2C Parameters</td>
</tr>
<tr>
<td>Table 7.5</td>
<td>SB_SPI Port Signals</td>
</tr>
<tr>
<td>Table 7.6</td>
<td>SB_SPI Parameters</td>
</tr>
<tr>
<td>Table 7.7</td>
<td>SB_HOFOCP Port Signals</td>
</tr>
<tr>
<td>Table 7.8</td>
<td>SB_HOFOCP Parameters</td>
</tr>
<tr>
<td>Table 7.9</td>
<td>SB_LFOCP Port Signals</td>
</tr>
<tr>
<td>Table 7.10</td>
<td>SB_LED_DRV_CRT Port Signals</td>
</tr>
<tr>
<td>Table 7.11</td>
<td>SB_RGB_DRV Port Signals</td>
</tr>
<tr>
<td>Table 7.12</td>
<td>SB_IR_DRV Port Signals</td>
</tr>
<tr>
<td>Table 7.13</td>
<td>SB_RGB_IP Port Signals</td>
</tr>
<tr>
<td>Table 7.14</td>
<td>SB_I0_O6 Port Signals</td>
</tr>
<tr>
<td>Table 7.15</td>
<td>SB_I2C Port Signals</td>
</tr>
<tr>
<td>Table 7.16</td>
<td>SB_I2C Parameters</td>
</tr>
</tbody>
</table>
Table 7.17. SB_SPI Port Signals ............................................................................. 123
Table 7.18. SB_SPI Parameters ........................................................................... 124
Table 7.19. SB_MAC16 Port Signals .................................................................... 126
Table 7.20. SB_MAC16 Parameters .................................................................... 128
Table 7.21. Multiplier Configurations ................................................................. 131
Table 7.22. MAC Configurations ....................................................................... 132
Table 7.23. ACCUMULATOR Configurations ...................................................... 133
Table 7.24. Add/Subtract Configurations ............................................................ 134
Table 7.25. Multiply Add/Subtract Configurations ............................................. 135
Table 7.26. SB_HFOSC Port Signals ................................................................. 137
Table 7.27. SB_HFOSC Parameters ................................................................ 137
Table 7.28. SB_IFOSC Port Signals ................................................................. 138
Table 7.29. SB_RGBA_DRV Port Signals ........................................................... 139
Table 7.30. SB_IR400_DRV Port Signals ............................................................ 141
Table 7.31. SB_BARCODE_DRV Port Signals ................................................... 143
Table 7.32. SB_IR500_DRV Port Signals ............................................................ 144
Table 7.33. SB_LEDDA_IP Port Signals ............................................................. 146
Table 7.34. SB_IR_IP Port Signals .................................................................... 148
Table 7.35. SB_IO_OD Port Signals .................................................................. 150
Table 7.36. SB_I2C_FIFO Port Signals ............................................................... 152
Table 7.37. SB_I2C_FIFO Parameters ................................................................. 153
Table 7.38. SB_HFOSC Port Signals ................................................................. 155
Table 7.39. SB_PLL40_CORE Parameters ........................................................ 155
Table 7.40. SB_IFOSC Port Signals ................................................................. 156
Table 7.41. SB_RGBA_DRV Port Signals ........................................................... 157
Table 7.42. SB_LEDDA_IP Port Signals ............................................................. 159
Table 7.43. SB_IO_OD Port Signals .................................................................. 161
Table 7.44. SB_I2C Port Signals ....................................................................... 163
Table 7.45. SB_I2C Parameters ....................................................................... 164
Table 7.46. SB_SPI Port Signals ....................................................................... 166
Table 7.47. SB_SPI Parameters ....................................................................... 167
Table 7.48. SB_SPRAM256KA Ports ................................................................. 168
Table 7.49. Input Pin Function ......................................................................... 170
Table 7.50. Output Pin Function ...................................................................... 170
Table 7.51. Resistor Value ............................................................................... 171
Acronyms in This Document
A list of acronyms used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>First In, First Out</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>I^2C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>LED</td>
<td>Light-emitting Diode</td>
</tr>
<tr>
<td>LPSG</td>
<td>Low Power Strobe Generator</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Locked Loop</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
</tbody>
</table>
1. Register Primitives

1.1. SB_DFF – D Flip-Flop

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

![SB_DFF Diagram](image)

**Figure 1.1. SB_DFF – D Flip-Flop**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>†</td>
</tr>
<tr>
<td>1</td>
<td>†</td>
</tr>
<tr>
<td>Power On State</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>† – Rising Edge</td>
</tr>
<tr>
<td>1 – High logic level</td>
</tr>
<tr>
<td>0 – Low logic level</td>
</tr>
<tr>
<td>X – Don’t care</td>
</tr>
<tr>
<td>? – Unknown</td>
</tr>
</tbody>
</table>

**HDL use**

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

**Verilog Instantiation**

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

**VHDL Instantiation**

```vhdl
-- SB_DFF - D Flip-Flop.
SB_DFF_inst: SB_DFF  
  port map (  
    Q => Q, -- Registered Output  
    C => C, -- Clock  
    D => D, -- Data  
  );  
-- End of SB_DFF instantiation
```
1.2. **SB_DFFE — D Flip-Flop with Clock Enable**

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

![SB_DFFE — D Flip-Flop with Clock Enable](image)

**Figure 1.2. SB_DFFE — D Flip-Flop with Clock Enable**

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

**Table 1.2. SB_DFFE — D Flip-Flop with Clock Enable**

**HDL Usage**

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

**Default Signal Values**

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

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

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

**Verilog Instantiation**

```verilog
// SB_DFFE - D Flip-Flop with Clock Enable.
SB_DFFE SB_DFFE_inst (  
  .Q(Q),       // Registered Output  
  .C(C),       // Clock  
  .D(D),       // Data  
  .E(E)        // Clock Enable  
);  
// End of SB_DFFE instantiation
```
**VHDL Instantiation**

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

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

-- End of SB_DFFE instantiation
```
1.3. **SB_DFFSR – D Flip-Flop with Synchronous Reset**

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

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

![SB_DFFSR Diagram](image)

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

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

**Power On State**

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

**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

```verilog
SB_DFFSR SB_DFFSR_inst (  
   .Q(Q),   // Registered Output  
   .C(C),   // Clock  
   .D(D),   // Data  
   .R(R) // Synchronous Reset  
);
// End of SB_DFFSR instantiation
```
VHDL Instantiation

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

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

-- End of SB_DFFSR instantiation

1.4. SB_DFFR – D Flip-Flop with Asynchronous Reset

Data: D is loaded into the flip-flop when R is low during a rising clock edge transition.
Reset: R input is active high, overrides all other inputs and asynchronously resets the Q output.

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

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

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

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

Default Signal Values
The iCEcube2 software assigns the following signal values to unconnected input ports:
Input D: Logic ‘0’
Input C: Logic ‘0’
Input R: Logic ‘0’
Verilog Instantiation

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

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

VHDL Instantiation

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

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

-- End of SB_DFFR instantiation

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

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

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

![SB_DFFSS](image)

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

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

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

<table>
<thead>
<tr>
<th>Key</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Unknown</td>
</tr>
<tr>
<td>0</td>
<td>Low logic level</td>
</tr>
<tr>
<td>1</td>
<td>Rising Edge</td>
</tr>
</tbody>
</table>

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at [www.latticesemi.com/legal](http://www.latticesemi.com/legal).

All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
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
1.6. **SB_DFFS – D Flip-Flop with Asynchronous Set**

Data: D is loaded into the flip-flop when S is low during a rising clock edge transition.
Set: S input is active high, and it overrides all other inputs and asynchronously sets the Q output.

![SB_DFFS Diagram](image)

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

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

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

**HDL Usage**

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

**Default Signal Values**

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

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

**Verilog Instantiation**

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

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

// End of SB_DFFS instantiation
```

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
VHDL Instantiation

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

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

-- End of SB_DFFS instantiation

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

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

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

![Figure 1.7. SB_DFFESR – D Flip-Flop with Clock Enable and Synchronous Reset](image)

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

Key:

- \( \uparrow \) – Rising Edge
- \( 1 \) – High logic level
- \( 0 \) – Low logic level
- \( X \) – Don’t care
- \( ? \) – Unknown

HDL Usage

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

Default Signal Values

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

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

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

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

// End of SB_DFFS instantiation

VHDL Instantiation

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

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

-- End of SB_DFFS instantiation

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

Data: D is loaded into the flip-flop when Reset R is low and Clock Enable E is high during a rising clock edge transition. Reset: R, when asserted with Clock Enable E high, synchronously resets the Q output during a rising clock edge.

![SB_DFFER Diagram](image)

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

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

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

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

Default Signal Values
The iCEcube2 software assigns the following signal values to unconnected input ports:
- Input D: Logic ‘0’
- Input C: Logic ‘0’
- Input R: Logic ‘0’
- Input E: Logic ‘1’

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

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

```verilog
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.

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

-- End of SB_DFFER instantiation
1.9. **SB_DFFESS – D Flip-Flop with Clock Enable and Synchronous Set**

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

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

![SB_DFFESS diagram](image)

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

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

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

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

**HDL Usage**

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

**Default Signal Values**

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

- Input D: Logic ‘0’
- Input C: Logic ‘0’
- Input R: Logic ‘0’
- Input S: Logic ‘0’
Verilog Instantiation

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

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

// End of SB_DFFESS instantiation

VHDL Instantiation

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

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

-- End of SB_DFFESS instantiation

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

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

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

---

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

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

**Key**

- ✓ – 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**

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

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

// End of SB_DFFES instantiation
```

**VHDL Instantiation**

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

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

-- End of SB_DFFES instantiation
```
1.11. SB_DFFN – D Flip-Flop – Negative Edge Clock

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

![SB_DFFN Diagram](image)

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

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

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

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

**HDL Usage**

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

**Default Signal Values**

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

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

**Verilog Instantiation**

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

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

// End of SB_DFFN instantiation
VHDL Instantiation

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

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

-- End of SB_DFFN instantiation

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

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

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

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

HDL Usage

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

Default Signal Values

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

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

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

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

```verilog
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.

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

-- End of SB_DFFNE instantiation
1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset

Data: D is loaded into the flip-flop when R is low during the falling clock edge transition.
Reset: R input is active high, overrides all other inputs and resets the Q output during the falling clock edge transition.

![Figure 1.13. SB_DFFNSR – D Flip-Flop – Negative Edge Clock with Synchronous Reset](image)

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

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

HDL Usage

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

Default Signal Values

The iCEcube2 software assigns the following signal values to unconnected input ports:
Input D: Logic ‘0’
Input C: Logic ‘0’
Input R: Logic ‘0’
Verilog Instantiation

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

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

// End of SB_DFFNSR instantiation

VHDL Instantiation

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

SB_DFFNSR_inst: SB_DFFNSR

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

End of SB_DFFNSR instantiation
1.14. SB_DFFNR – D Flip-Flop – Negative Edge Clock with Asynchronous Reset

Data: D is loaded into the flip-flop when R is low during the falling clock edge transition.
Reset: R input is active high, overrides all other inputs and asynchronously resets the Q output.

![Diagram of SB_DFFNR](image)

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

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

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

**HDL Usage**

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

**Default Signal Values**

The iCEcube2 software assigns the following signal values to unconnected input ports:
Input D: Logic ‘0’
Input C: Logic ‘0’
Input R: Logic ‘0’

**Verilog Instantiation**

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

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

// End of SB_DFFNR instantiation
```
VHDL Instantiation

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

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

-- End of SB_DFFNR instantiation

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

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

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

![Diagram of SB_DFFNSS](image)

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

<table>
<thead>
<tr>
<th>Inputs</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Power On State</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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,

```verilog
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.

```vhdl
SB_DFFNSS_inst : SB_DFFNSS  
    port map (  
        Q => Q, -- Registered Output  
        C => C, -- Clock  
        D => D, -- Data  
        S => S -- Synchronous Set  
    );  
-- End of SB_DFFNSS instantiation
```
1.16. **SB_DFFNS – D Flip-Flop – Negative Edge Clock with Asynchronous Set**

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

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

![SB_DFFNS Diagram](image)

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

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

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

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

**HDL Usage**

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

**Default Signal Values**

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

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

**Verilog Instantiation**

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

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

// End of SB_DFFNS instantiation
```
VHDL Instantiation

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

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

-- End of SB_DFFNS instantiation

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

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

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

![SB_DFFNESR Diagram](image)

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

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

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>R, E, D, C, Q</td>
<td>1</td>
<td>(^{-}) Falling Edge</td>
</tr>
<tr>
<td>0, 0, 0, 0</td>
<td>0</td>
<td>1 – High logic level</td>
</tr>
<tr>
<td>0, X, 0, 1</td>
<td>1</td>
<td>0 – Low logic level</td>
</tr>
<tr>
<td>X, X, X, 0</td>
<td>1</td>
<td>X – Don’t care</td>
</tr>
<tr>
<td>Power On State</td>
<td>0</td>
<td>? – Unknown</td>
</tr>
</tbody>
</table>

1 – High logic level
0 – Low logic level
\(^{-}\) Falling Edge
X - Don’t care
? – Unknown

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

Default Signal Values
The iCEcube2 software assigns the following signal values to unconnected input ports:
Input D: Logic '0'
Input C: Logic ‘0’
Input R: Logic ‘0’
Input E: Logic ‘1’

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

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

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

// End of SB_DFFNESR instantiation
```

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

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

-- End of SB_DFFNESR instantiation
```
1.18. SB_DFFNER – D Flip-Flop – Negative Edge Clock, Enable and Asynchronous Reset

Data: D is loaded into the flip-flop when R is low and E is high during the falling clock edge transition.
Reset: R input is active high, and it overrides all other inputs and asynchronously resets the Q output.

![SB_DFFNER Diagram](image)

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

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

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

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

Default Signal Values
The iCEcube2 software assigns the following signal values to unconnected input ports:
Input D: Logic ‘0’
Input C: Logic ‘0’
Input R: Logic ‘0’
Input E: Logic ‘1’

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

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

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

// End of SB_DFFNER instantiation

VHDL Instantiation

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

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

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

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

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

![SB_DFFNESS Diagram](image)

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

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

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

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

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

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

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

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

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

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

Data: D is loaded into the flip-flop when S is low and E is high during the falling clock edge transition.
Set: S input is active high, and it overrides all other inputs and asynchronously sets the Q output.

```
+---+---+---+---+
|   |   |   |   |
| S | E | D | CLK |
+---+---+---+---+
| 1 | X | X | X  |
+---+---+---+---+
| 0 | 0 | X | X  |
+---+---+---+---+
| 0 | 1 | 0 |    |
+---+---+---+---+
| 0 | 1 | 1 |    |
+---+---+---+---+
Power On State | X | X | X | 0 |
+---+---+---+---+
```

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

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

Default Signal Values
The iCEcube2 software assigns the following signal values to unconnected input ports:
Input D: Logic ‘0’
Input C: Logic ‘0’
Input S: Logic ‘0’
Input E: Logic ‘1’

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

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

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

// End of SB_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
2. Combinational Logic Primitives

2.1. SB_LUT4

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

![Figure 2.1. SB_LUT4](image)

Initialization Values

LUT state initialization parameter LUT_INIT = 16'hxxxx;

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT_INIT[0]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[1]</td>
<td>1</td>
</tr>
<tr>
<td>LUT_INIT[2]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[3]</td>
<td>1</td>
</tr>
<tr>
<td>LUT_INIT[4]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[5]</td>
<td>1</td>
</tr>
<tr>
<td>LUT_INIT[6]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[7]</td>
<td>1</td>
</tr>
<tr>
<td>LUT_INIT[8]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[9]</td>
<td>1</td>
</tr>
<tr>
<td>LUT_INIT[10]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[12]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[13]</td>
<td>1</td>
</tr>
<tr>
<td>LUT_INIT[14]</td>
<td>0</td>
</tr>
<tr>
<td>LUT_INIT[15]</td>
<td>1</td>
</tr>
</tbody>
</table>

HDL Usage

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

Default Signal Values

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

// SB_LUT4 : 4-input Look-Up Table

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

defparam SB_LUT4_inst.LUT_INIT=16'hxxxx;
//LUT state initialization parameter, 16 bits.

//End of SB_LUT4 instantiation

VHDL Instantiation

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

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

2.2. SB_CARRY

Carry Logic

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

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

![Figure 2.2. Carry Logic Structure within a Logic Cell](image)

Table 2.2. SB_CARRY

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

HDL Usage

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

Default Signal Values

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

Verilog Instantiation

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

VHDL Instantiation

```vhdl
my_carry_inst : SB_CARRY  port map (  CO => CO,  CI => CI,  I0 => I0,  I1 => I1 );
```
3. **iCE40 Block RAM Primitives**

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

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

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

### Table 3.1. iCE40 Block RAM

<table>
<thead>
<tr>
<th>Block RAM Configuration</th>
<th>Block RAM Size</th>
<th>WADDR Port Size (Bits)</th>
<th>WDATA Port Size (Bits)</th>
<th>RADDR Port Size (Bits)</th>
<th>RDATA Port Size (Bits)</th>
<th>MASK Port Size (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>256x16 (4K)</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>16 [15:0]</td>
</tr>
<tr>
<td>SB_RAM256x16NR</td>
<td>256x16 (4K)</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>16 [15:0]</td>
</tr>
<tr>
<td>SB_RAM256x16RW</td>
<td>256x16 (4K)</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>16 [15:0]</td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td>256x16 (4K)</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>8 [7:0]</td>
<td>16 [15:0]</td>
<td>16 [15:0]</td>
</tr>
<tr>
<td>SB_RAM512x8</td>
<td>512x8 (4K)</td>
<td>9 [8:0]</td>
<td>8 [7:0]</td>
<td>8 [8:0]</td>
<td>8 [7:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM512x8NR</td>
<td>512x8 (4K)</td>
<td>9 [8:0]</td>
<td>8 [7:0]</td>
<td>8 [8:0]</td>
<td>8 [7:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM2048x2</td>
<td>2048x2 (4K)</td>
<td>11 [10:0]</td>
<td>2 [1:0]</td>
<td>10 [9:0]</td>
<td>2 [1:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM2048x2NR</td>
<td>2048x2 (4K)</td>
<td>11 [10:0]</td>
<td>2 [1:0]</td>
<td>10 [9:0]</td>
<td>2 [1:0]</td>
<td>No Mask Port</td>
</tr>
<tr>
<td>SB_RAM2048x2NRNW</td>
<td>2048x2 (4K)</td>
<td>11 [10:0]</td>
<td>2 [1:0]</td>
<td>10 [9:0]</td>
<td>2 [1:0]</td>
<td>No Mask Port</td>
</tr>
</tbody>
</table>

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

### Table 3.2. 256 × 16 RAM Block Configuration

<table>
<thead>
<tr>
<th>RAM Primitive Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB_RAM256x16</td>
<td>Posedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM4256x16NR</td>
<td>Negedge Read clock, Posedge Write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NW</td>
<td>Posedge Read clock, Negedge Write clock</td>
</tr>
<tr>
<td>SB_RAM256x16NRNW</td>
<td>Negedge Read clock, Negedge Write clock</td>
</tr>
</tbody>
</table>
3.1. \textbf{SB\_RAM256\times16}

The following modules are the complete list of SB\_RAM256\times16 based primitives

\textbf{SB\_RAM256\times16} \quad //Posedge clock RCLK WCLK

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

\textbf{Verilog Instantiation:}

```verilog
SB\_RAM256\times16 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 =
```

\textbf{Figure 3.1. SB\_RAM256\times16}
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:
ram256x16_inst : SB_RAM256x16 generic map ( 
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F =>
X"0000000000000000000000000000000000000000000000000000000000000000"
) 
port map ( 
  RDATA => RDATA_c,
  RADDR => RADDR_c,
  RCLK => RCLK_c,
  RCLKE => RCLKE_c,
  RE => RE_c,
  WADDR => WADDR_c,
  WCLK=> WCLK_c,
  WCLKE => WCLKE_c,
  WDATA => WDATA_c,
  MASK => MASK_c,
  WE => WE_c 
);
3.1.2. SB_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 =>  
  RDATA_c,  
  RADDR_c,  
  RCLKN_c,  
  RCLKE_c,  
  RE_c,  
  WADDR_c,  
  WCLK_c,  
  WCLKE_c,  
  WDATA_c,  
  MASK_c,  
  WE)  
```
3.1.3. SB_RAM256x16NW

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

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

```vhdl
ram256x16nr_inst : SB_RAM256x16NR
generic map (  
    INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",  
    INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"  
  )
port map (  
    RDATA => RDATA_c,  
    RADDR => RADDR_c,  
    RCLKN => RCLKN_c,  
    RCLKE => RCLKE_c,  
    RE => RE_c,  
    WADDR => WADDR_c,  
    WCLK=> WCLK_c,  
    WCLKE => WCLKE_c,  
    WDATA => WDATA_c,  
    MASK => MASK_c,  
    WE => WE_c  
);  
```
3.1.4. SB_RAM256x16NRNW

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

Verilog Instantiation:

SB_RAM256x16NRNW ram256x16nwnw_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 ram256x16nwnw_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16nwnw_inst.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;
VHDL Instantiation:
ram256x16nrnw_inst : SB_RAM256x16NRNW
generic map (  
  INIT_0 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_1 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_2 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_3 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_4 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_5 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_6 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_7 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_8 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_9 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_A =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_B =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_C =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_D =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_E =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (  
  RDATA => RDATA_c,  
  RADDR => RADDR_c,  
  RCLKN => RCLKN_c,  
  RCLKE => RCLKE_c,  
  RE => RE_c,  
  WADDR => WADDR_c,  
  WCLKN => WCLKN_c,  
  WCLKE => WCLKE_c,  
  WDATA => WDATA_c,  
  MASK => MASK_c,  
  WE => WE_c  
);
3.2. **SB_RAM512x8**

The following modules are the complete list of SB_RAM512x8 based primitives

### 3.2.1. **SB_RAM512x8**

**SB_RAM512x8**

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

**Verilog Instantiation:**

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

```verilog
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;
```

---

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
defparam ram512x8_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;

VHDL Instantiation:

ram512x8_inst : SB_RAM512x8
generic map (
  INIT_0 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000" )
port map (  
RDATA => RDATA_c,
RADDR => RADDR_c,
RCLK => RCLK_c,
RCLKE => RCLKE_c,
RE => RE_c,

3.2.2. **SB_RAM512x8NR**

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

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

Verilog Instantiation:

```
SB_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
);
3.2.3. SB_RAM512x8NW

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

Verilog Instantiation:

SB_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  
);```

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

Verilog Instantiation:

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

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

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

### 3.3. SB_RAM1024x4

![2-Port Register File Diagram](image)

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

SB_RAM1024x4

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

Verilog Instantiation:

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

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

Ram1024x4_inst: SB_RAM1024x4 generic map (
  INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (  
  RDATA  =>  RDATA_c,  
  RADDR  =>  RADDR_c,  
  RCLK   =>  RCLK_c,  
  RCLKE  =>  RCLKE_c,  
  RE     =>  RE_c,  
  WADDR  =>  WADDR_c,  
  WCLK   =>  WCLK_c,  
  WCLKE  =>  WCLKE_c,  
  WDATA  =>  WDATA_c,  
  WE     =>  WE_c  
);
3.3.2. SB_RAM1024x4NR

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

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

Verilog Instantiation:

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

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

```vhdl
ram1024x4nr_inst: SB_RAM1024x4NR
  generic map (
    INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_6 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_7 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_8 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_9 =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_A =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_B =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_C =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_D =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_E =>
      X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_F =>
      X"0000000000000000000000000000000000000000000000000000000000000000"
  )
  port map (
    RDATA => RDATA_c,
    RADDR => RADDR_c,
    RCLKN => RCLKN_c,
    RCLKE => RCLKE_c,
    RE => RE_c,
    WADDR => WADDR_c,
    WCLK => WCLK_c,
    WCLKE => WCLKE_c,
    WDATA => WDATA_c,
    WE => WE_c
  );
```
3.3.3. SB_RAM1024x4NW

SB_RAM1024x4NW // Negative edged Write Clock – i.e. WCLKN
(RDATA, RCLK, RCLKE, RE, RADDR, WADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

Verilog Instantiation:

```
SB_RAM1024x4NW     ram1024x4nw_inst (  
    .RDATA(RDATA_c[3:0]),  
    .RADDR(RADDR_c[9:0]),  
    .RCLK(RCLK_c),  
    .RCLKE(RCLKE_c),  
    .RE(RE_c),  
    .WADDR(WADDR_c[3:0]),  
    .WCLKN(WCLKN_c),  
    .WCLKE(WCLKE_c),  
    .WDATA(WDATA_c[9:0]),  
    .WE(WE_c)  
);
```

```
defparam ram1024x4_inst.INIT_0 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_1 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_2 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_3 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_4 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_5 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_6 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_7 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_8 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_9 = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_A = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_B = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_C = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_D = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_E = 256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_F = 256'h0000000000000000000000000000000000000000000000000000000000000000;
```
VHDL Instantiation:
ram1024x4nw_inst : SB_RAM1024x4NW

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

port map (
RDATA => RDATA_c,
RADDR => RADDR_c,
RCLK => RCLK_c,
RCLKE => RCLKE_c,
RE => RE_c,
WADDR => WADDR_c,
WCLKN=> WCLKN_c,
WCLKE => WCLKE_c,
WDATA => WDATA_c,
WE => WE_c
);
3.3.4. SB_RAM1024x4NRNW

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

Verilog Instantiation:

```verilog
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;
3.4. SB_RAM2048x2

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

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

Verilog Instantiation:

```verilog
SB_RAM2048x2 ram2048x2_inst (  
  .RDATA(RDATA_c[1:0]),  
  .RADDR(RADDR_c[10:0]),  
  .RCLK(RCLK_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[10:0]),  
  .WCLK(WCLK_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[1:0]),  
  .WE(WE_c)  
);

defparam ram2048x2_inst.INIT_0 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_1 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_2 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_3 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_4 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_5 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_6 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_7 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_8 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_9 =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_A =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_B =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_C =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_D =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_E =  
256'h0000000000000000000000000000000000000000000000000000000000000000;

defparam ram2048x2_inst.INIT_F =  
256'h0000000000000000000000000000000000000000000000000000000000000000;
```
VHDL Instantiation:
Ram2048x2_inst : SB_RAM2048x2

generic map (  
  INIT_0 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_1 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_2 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  INIT_3 =>  
    X"0000000000000000000000000000000000000000000000000000000000000000",  
  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 => RDAT_A_c,  
  RADDR => RADDR_c,  
  RCLK => RCLK_c,  
  RCLKE => RCLKE_c,  
  RE => RE_c,  
  WADDR => WADDR_c,  
  WCLK=> WCLK_c,  
  WCLKE => WCLKE_c,  
  WDATA => WDATA_c,  
  WE => WE_c
);
3.4.2. SB_RAM2048x2NR

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

Verilog Instantiation

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

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

```vhdl
ram2048x2nr_inst : SB_RAM2048x2NR
generic map (  
  INIT_0 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_1 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_2 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_3 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_4 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_5 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_6 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_7 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_8 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_9 =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_A =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_B =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_C =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_D =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_E =>
  X"0000000000000000000000000000000000000000000000000000000000000000",
  INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
port map (  
  RDATA => RDATA_c,
  RADDR => RADDR_c,
  RCLKN => RCLKN_c,
  RCLKE => RCLKE_c,
  RE => RE_c,
  WADDR => WADDR_c,
  WCLK => WCLK_c,
  WCLKE => WCLKE_c,
  WDATA => WDATA_c,
  WE => WE_c
);```
3.4.3. SB_RAM2048x2NW

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

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

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

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

SB_RAM2048x2NRNW ram2048x2nrnw_inst (  
  .RDATA(RDATA_c[1:0]),  
  .RADDR(RADDR_c[10:0]),  
  .RCLKN(RCLKN_c),  
  .RCLKE(RCLKE_c),  
  .RE(RE_c),  
  .WADDR(WADDR_c[10:0]),  
  .WCLKN(WCLKN_c),  
  .WCLKE(WCLKE_c),  
  .WDATA(WDATA_c[1:0]),  
  .WE(WE_c)  
);

// Defparam

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:

```vhdl
ram2048x2nrnw_inst : SB_RAM2048x2NRNW
  generic map (
    INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
  )
  port map (  
    RDATA => RDATA_c, 
    RADDR => RADDR_c, 
    RCLKN => RCLKN_c, 
    RCLKE => RCLKE_c, 
    RE => RE_c, 
    WADDR => WADDR_c, 
    WCLKN=> WCLKN_c, 
    WCLKE => WCLKE_c, 
    WDATA => WDATA_c, 
    WE => WE_c 
  );
```
3.5. **SB_RAM40_4K**

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

**Table 3.3. SB_RAM40_4K Naming Convention Rules**

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

![Port Register File](image)

**Figure 3.5. SB_RAM40_4K**

The following table lists the signals for both ports.

**Table 3.4. SB_RAM40_4K RAM Port Signals**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDATA[15:0]</td>
<td>Input</td>
<td>Write Data input</td>
</tr>
<tr>
<td>MASK[15:0]*</td>
<td>Input</td>
<td>Bit-line Write Enable input, active low. Applicable only when WRITE_MODE parameter is set to 0.</td>
</tr>
<tr>
<td>WADDR[7:0]</td>
<td>Input</td>
<td>Write Address input. Selects up to 256 possible locations</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>Write Enable input, active high</td>
</tr>
<tr>
<td>WCLK</td>
<td>Input</td>
<td>Write Clock input, rising-edge active</td>
</tr>
<tr>
<td>WCLKE</td>
<td>Input</td>
<td>Write Clock Enable input</td>
</tr>
<tr>
<td>RDATA[15:0]</td>
<td>Output</td>
<td>Read Data output</td>
</tr>
<tr>
<td>RADDR[7:0]</td>
<td>Input</td>
<td>Read Address input. Selects one of 256 possible locations</td>
</tr>
<tr>
<td>RE</td>
<td>Input</td>
<td>Read Enable input, active high</td>
</tr>
<tr>
<td>RCLK</td>
<td>Input</td>
<td>Read Clock input, rising-edge active</td>
</tr>
<tr>
<td>RCLKE</td>
<td>Input</td>
<td>Read Clock Enable input</td>
</tr>
</tbody>
</table>
Table 3.5. SB_RAM40_4K RAM Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_0, ..., INIT_F</td>
<td>RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16 × 256 = 4096 total bits)</td>
<td>INIT_0 to INIT_F</td>
<td>Initialize the RAM with predefined value</td>
</tr>
<tr>
<td>WRITE_MODE</td>
<td>Sets the RAM block write port configuration</td>
<td>0</td>
<td>256×16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>512×8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1024×4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>2048×2</td>
</tr>
<tr>
<td>READ_MODE</td>
<td>Sets the RAM block read port configuration</td>
<td>0</td>
<td>256×16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>512×8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1024×4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>2048×2</td>
</tr>
</tbody>
</table>

Verilog Instantiation:

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

VHDL Instantiation:

```vhdl
-- Physical RAM Instance without Pre Initialization
ram40_4kinst_physical : SB_RAM40_4K  
generic map (  
  READ_MODE => 0,  
  WRITE_MODE=> 0 )  
port map (  
  RDATA=>RDATA,  
  RADDR=>RADDR,  
  WADDR=>WADDR,  
  MASK=>MASK,  
  WDATA=>WDATA,  
  RCLKE=>RCLKE,  
  RCLK=>RCLK,  
  RE=>RE,  
  WCLKE=>WCLK,  
  WCLK=>WCLK,  
  WE=>WE  
);
```
### 3.5.1. SB_RAM40_4KNR

**Verilog Instantiation:**

```verilog
// 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)  
);
```

```verilog
defparam ram40_4knrinst_physical.READ_MODE=0;  
defparam ram40_4knrinst_physical.WRITE_MODE=0;  
```

**VHDL Instantiation:**

```vhdl
-- Physical RAM Instance without Pre Initialization
ram40_4knrinst_physical : SB_RAM40_4KNR  
generic map (  
  READ_MODE => 0,  
  WRITE_MODE=>0  
)
port map (  
  RDATA=>RDATA,  
  RADDR=>RADDR,  
  WADDR=>WADDR,  
  MASK=>MASK,  
  WDATA=>WDATA,  
  RCLKE=>RCLKE,  
  RCLKN=>RCLKN,  
  RE=>RE,  
  WCLKE=>WCLKE,  
  WCLK=>WCLK,  
  WE=>WE  
);
```
3.5.2. SB_RAM40_4KNW

Verilog Instantiation:

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

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

VHDL Instantiation:

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

Verilog Instantiation:

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

defparam ram40_4knrnwinst_physical.READ_MODE=0;
defparam ram40_4knrnwinst_physical.WRITE_MODE=0;
```

VHDL Instantiation:

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

4.1. SB_IO

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

![SB_IO Diagram]

**Default Signal Values**

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

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

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

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

**Table 4.1. Input Pin Function**

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

**Table 4.2. Output Pin Function**

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

**Syntax Verilog Use**

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

```verilog
defparam my_generic_IO.PIN_TYPE = 6'b(Output Pin Function, Input Pin Function);
```
4.1.1. DDR I/O Configuration

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

```vhdl
defparam my_DDR_IO.PIN_TYPE = 6'b100000;
  // PIN_TYPE [5:2] = 1000
  // PIN_TYPE [1:0] = 00
```

This creates a DDR I/O pin whereby the input data is clocked in on both the rising and falling input clock edges. The output 'DDR' data is clocked out on rising and falling output clock edges, and the output may be tri-stated, using the output enable port of the SB_IO.

4.1.2. High Drive SB_IO

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

**Synthesis Attribute Syntax:**

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

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

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

4.1.3. Pull Up Resistor Configuration

For iCE40UL device, you can configure the internal pull-up resistor strength of an I/O to a predefined resistor value through the attribute settings. By default, all the I/O are weakly pulled up by 100k internal resistor. The PULLUP_RESISTOR attribute is effective only when PULLUP parameter is set to 1.

**Synthesis Attribute Syntax:**

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

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

**Note:** PULLUP_RESISTOR attribute is supported only for iCE40UL devices.
Verilog Instantiation

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
 .INPUT_CLK (input_clk),        Clock for the input registers
 .OUTPUT_CLK (output_clk),       Clock for the output registers
 .OUTPUT_ENABLE (output_enable),       Output Pin Tristate/Enable control
 .D_OUT_0 (d_out_0),       Data 0 – out to Pin/Rising clk edge
 .D_OUT_1 (d_out_1),       Data 1 – out to Pin/Falling clk edge
 .D_IN_0 (d_in_0),        Data 0 – Pin input/Rising clk edge
 .D_IN_1 (d_in_1)        Data 1 – Pin input/Falling clk edge
 ) /* synthesis DRIVE_STRENGTH= x2 */;

defparam IO_PIN_INST.PIN_TYPE = 6'b000000; // See Input and Output Pin Function Tables.
    // Default value of PIN_TYPE = 6’000000 i.e. an input pad, with the input signal registered.
defparam IO_PIN_INST.PULLUP = 1'b0; // By default, the IO will have NO pull up.
    // This parameter is used only on bank 0, 1, and 2. Ignored when it is placed at bank 3

defparam IO_PIN_INST.NEG_TRIGGER = 1'b0; // Specify the polarity of all FFs in the IO to be falling edge when NEG_TRIGGER = 1.
    // Default is rising edge.
defparam IO_PIN_INST.IO_STANDARD = "SB_LVCMOS"; // Other IO standards are supported in bank 3 only: SB_SSTL2_CLASS_2, SB_SSTL2_CLASS_1,
    // SB_SSTL18_FULL, SB_SSTL18_HALF, SB_MDDR10,
    // SB_MDDR8, SB_MDDR4, SB_MDDR2 etc.
5. Global Buffer Primitives

5.1. SB_GB_IO

Default Signal Values

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

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

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

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

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

![Diagram of SB_GB Primitive with labels C0, global buffer output, I0, CI, I1.](image)

**Figure 5.2. SB_GB Primitive**

Verilog Instantiation

```verilog
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)
);
```

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
6. PLL Primitives

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

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

6.1. SB_PLL40_CORE

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

![Figure 6.1: SB_PLL40_CORE](image)

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFERENCECLK</td>
<td>PLL source clock that serves as the input to the SB_PLL40_CORE primitive.</td>
</tr>
<tr>
<td>PLLOUTGLOBAL</td>
<td>Output clock generated by the PLL, drives a global clock network on the FPGA.</td>
</tr>
<tr>
<td>PLLOUTCORE</td>
<td>Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTGLOBAL port.</td>
</tr>
<tr>
<td>LOCK</td>
<td>Output port, when HIGH, indicates that the signal on PLLOUTGLOBAL/PLLOUTCORE is locked to the PLL source on REFERENCECLK.</td>
</tr>
<tr>
<td>EXTFEEDBACK</td>
<td>External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.</td>
</tr>
<tr>
<td>DYNAMICDELAY</td>
<td>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.</td>
</tr>
<tr>
<td>LATCHINPUTVALUE</td>
<td>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’.</td>
</tr>
<tr>
<td>SCLK, SDI, SDO</td>
<td>These pins are used only for internal testing purposes, and need not be instantiated by users.</td>
</tr>
</tbody>
</table>

Ports

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

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

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

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

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

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

RESETEB: Active low input that asynchronously resets the PLL.

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

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

SCLK, SDI, SDO: These pins are used only for internal testing purposes, and need not be instantiated by users.
Parameters
The SB_PLL40_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.1. SB_PLL40_CORE Parameters

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

### 6.2. SB_PLL40_PAD

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

![Figure 6.2. SB_PLL40_PAD](image)
Ports

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

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

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

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

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

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

**RESETB:** Active low input that asynchronously resets the PLL.

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

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

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

Parameters

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

**Table 6.2. SB_PLL40_PAD Parameters**

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

6.3. **SB_PLL40_2_PAD**

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

![SB_PLL40_2_PAD](image)

**Figure 6.3. SB_PLL40_2_PAD**

**Ports**

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

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

PLLOUTCOREA: The signal on PACKAGEPIN appears on the FPGA at this pin, which drives regular FPGA routing. Do not use this pin in an external feedback path to the PLL.

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

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

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

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

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

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

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

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

**Parameters**

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

**Table 6.3. SB_PLL40_2_PAD Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td>_FEEDBACK</td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,....,15</td>
<td>The PLLOUTGLOBAL and PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Parameter Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_RELATIVE</td>
<td>Selects the mode for the Fine Delay Adjust block</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,...,15</td>
<td>The PLLOUTGLOBALA and PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)×150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.</td>
</tr>
<tr>
<td>SHIFTREG_DIV_MODE</td>
<td>Selects shift register configuration</td>
<td>0,1</td>
<td>Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0 -&gt; Divide by 4 1-&gt; Divide by 7</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTB</td>
<td>Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports</td>
<td>SHIFTER_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTER_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY” and SHIFTER_DIV_MODE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency outputs to PortB. No phase shift.</td>
</tr>
<tr>
<td>DIVR</td>
<td>REFERENCECLK divider</td>
<td>0,1,2,...,15</td>
<td>These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.</td>
</tr>
<tr>
<td>DIVF</td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by-factor of a divider in external feedback path</td>
<td>User-specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTA</td>
<td>Enables the PLL power- down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTB</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
6.4. **SB_PLL40_2F_CORE**

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

![SB_PLL40_2F_CORE](image)

**Figure 6.4. SB_PLL40_2F_CORE**

**Ports**

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

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

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

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

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

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

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

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

**RESETB**: Active low input that asynchronously resets the PLL.

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

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

**SCLK, SDI, SDO**: These pins are used only for internal testing purposes, and need not be instantiated by users.
Parameters
The SB_PLL40_2F_CORE primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.4. SB_PLL40_2F_CORE Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEEDBACK_PATH</td>
<td>Selects the feedback path to the PLL</td>
<td>SIMPLE</td>
<td>Feedback is internal to the PLL, directly from VCO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELAY</td>
<td>Feedback is internal to the PLL, through the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHASE_AND_DELAY</td>
<td>Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTERNAL</td>
<td>Feedback path is external to the PLL, and connects to EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_FEEDBACK</td>
<td>Selects the mode for the Fine Delay Adjust block in the feedback path</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins</td>
</tr>
<tr>
<td>FDA_FEEDBACK</td>
<td>Sets a constant value for the Fine Delay Adjust Block in the feedback path</td>
<td>0, 1,..,15</td>
<td>The PLLOUTGLOBALA and PLLOUTCOREA signals are delay compensated by (n+1)x150 ps, where n = FDA_FEEDBACK only if the setting of the DELAY_ADJUSTMENT_MODE_FEEDBACK is FIXED.</td>
</tr>
<tr>
<td>DELAY_ADJUSTMENT_MODE_RELATIVE</td>
<td>Selects the mode for the Fine Delay Adjust block</td>
<td>FIXED</td>
<td>Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DYNAMIC</td>
<td>Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins</td>
</tr>
<tr>
<td>FDA_RELATIVE</td>
<td>Sets a constant value for the Fine Delay Adjust Block</td>
<td>0, 1,..,15</td>
<td>The PLLOUTGLOBALA and PLLOUTCOREA signals are delayed w.r.t. the Port B signals, by (n+1)x150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is “FIXED”.</td>
</tr>
<tr>
<td>SHIFTER_DIV_MODE</td>
<td>Selects shift register configuration</td>
<td>0,1</td>
<td>Used when FEEDBACK_PATH is “PHASE_AND_DELAY”. 0-&gt; Divide by 4 1-&gt; Divide by 7</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Parameter Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTA</td>
<td>Selects the signal to be output at the PLLOUTCOREA and PLLOUTGLOBALA ports</td>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency outputs to PortA. No phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency divided by 2 and then output to PORTA. No phase shift.</td>
</tr>
<tr>
<td>PLLOUT_SELECT_PORTB</td>
<td>Selects the signal to be output at the PLLOUTCOREB and PLLOUTGLOBALB ports</td>
<td>SHIFTREG_0deg</td>
<td>0° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHIFTREG_90deg</td>
<td>90° phase shift only if the setting of FEEDBACK_PATH is “PHASE_AND_DELAY”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK</td>
<td>The internally generated PLL frequency outputs to PortB. No phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENCLK_HALF</td>
<td>The internally generated PLL frequency divided by 2 and then output to PORTB. No phase shift.</td>
</tr>
<tr>
<td>DIVR</td>
<td>REFERENCECLK divider</td>
<td>0,1,2,...,15</td>
<td>These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.</td>
</tr>
<tr>
<td>DIVF</td>
<td>Feedback divider</td>
<td>0,1,...,63</td>
<td></td>
</tr>
<tr>
<td>DIVQ</td>
<td>VCO Divider</td>
<td>1,2,...,6</td>
<td></td>
</tr>
<tr>
<td>FILTER_RANGE</td>
<td>PLL Filter Range</td>
<td>0,1,...,7</td>
<td></td>
</tr>
<tr>
<td>EXTERNAL_DIVIDE_FACTOR</td>
<td>Divide-by factor of a divider in external feedback path</td>
<td>User-specified value. Default 1</td>
<td>Specified only when there is a user-implemented divider in the external feedback path.</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTA</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
<tr>
<td>ENABLE_ICEGATE_PORTB</td>
<td>Enables the PLL power-down control</td>
<td>0</td>
<td>Power-down control disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Power-down controlled by LATCHINPUTVALUE input</td>
</tr>
</tbody>
</table>
### 6.5. **SB_PLL40_2F_PAD**

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

![Diagram of SB_PLL40_2F_PAD](image)

**Figure 6.5. SB_PLL40_2F_PAD**

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGEPIN</td>
<td>PLL REFERENCE CLOCK pin that serves as the input to the SB_PLL40_2F_PAD primitive.</td>
</tr>
<tr>
<td>RESETB</td>
<td>elayed input that asynchronously resets the PLL.</td>
</tr>
<tr>
<td>BYPASS</td>
<td>Input signal, when asserted, connects the signal on PACKAGEPIN to PLLOUTCORE/PLLOUTGLOBAL pins.</td>
</tr>
<tr>
<td>EXTFEEDBACK</td>
<td>External feedback input to PLL. Enabled when the FEEDBACK_PATH parameter is set to EXTERNAL.</td>
</tr>
<tr>
<td>DYNAMICDELAY</td>
<td>4-bit input bus that enables dynamic control of the delay contributed by the Fine Delay Adjust Block.</td>
</tr>
<tr>
<td>LATCHINPUTVALUE</td>
<td>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.</td>
</tr>
<tr>
<td>SCLK, SDI, SDO</td>
<td>These pins are used only for internal testing purposes, and need not be instantiated by users.</td>
</tr>
</tbody>
</table>

---

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at [www.latticesemi.com/legal](http://www.latticesemi.com/legal). All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
Parameters
The SB_PLL40_2F_PAD primitive requires configuration through the specification of the following parameters. It is strongly recommended that the configuration of the PLL primitives be accomplished through the use of the PLL Configuration tool that is offered as part of the iCEcube2 software.

Table 6.5. SB_PLL40_2F_PAD Parameters

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

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

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

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

![SB_HSOSC Diagram](image)

**Figure 7.1. SB_HSOSC**

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

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

**Synthesis Attribute**

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

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

**Verilog Instantiation**

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

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

![SB_LSOSC Diagram](image)

**Figure 7.2. SB_LSOSC**

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

**Table 7.2. SB_LSOSC Port Signals**

**Clock Network Selection**

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

**(Synthesis Attribute):**

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

**Value:**

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

**Verilog Instantiation**

```verilog
SB_LSOSC OSCInst0 (    
    .ENACLKK  (ENACLKK),
    .CLKK (CLKK)
  ) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```
7.1.3. SB_I2C

The I^C hard IP provides industry standard two pin communication interface that conforms to the version 2.1 of the I^C 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 I^C hard primitives, located at upper left corner and upper right corner of the chip.

![Figure 7.3. SB_I2C](image)

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBDATI0</td>
<td>Input</td>
<td>System Data input. Bit 0.</td>
</tr>
<tr>
<td>SBDATI1</td>
<td>Input</td>
<td>System Data input. Bit 1.</td>
</tr>
<tr>
<td>SBDATI2</td>
<td>Input</td>
<td>System Data input. Bit 2.</td>
</tr>
<tr>
<td>SBDATI3</td>
<td>Input</td>
<td>System Data input. Bit 3.</td>
</tr>
<tr>
<td>SBDATI5</td>
<td>Input</td>
<td>System Data input. Bit 5.</td>
</tr>
<tr>
<td>SBDATI7</td>
<td>Input</td>
<td>System Data input. Bit 7.</td>
</tr>
<tr>
<td>SBDATO0</td>
<td>Output</td>
<td>System Data output. Bit 0.</td>
</tr>
<tr>
<td>SBDATO1</td>
<td>Output</td>
<td>System Data output. Bit 1.</td>
</tr>
<tr>
<td>SBDATO2</td>
<td>Output</td>
<td>System Data output. Bit 2.</td>
</tr>
<tr>
<td>SBDATO3</td>
<td>Output</td>
<td>System Data output. Bit 3.</td>
</tr>
<tr>
<td>SBDATO5</td>
<td>Output</td>
<td>System Data output. Bit 5.</td>
</tr>
<tr>
<td>SBDATO7</td>
<td>Output</td>
<td>System Data output. Bit 7.</td>
</tr>
</tbody>
</table>

Table 7.3. SB_I2C Port Signals
iCE40 Technology Library
Technical Note

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

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBDAT00</td>
<td>Output</td>
<td>System Data Output. Bit 0.</td>
</tr>
<tr>
<td>SBDAT01</td>
<td>Output</td>
<td>System Data Output. Bit 1.</td>
</tr>
<tr>
<td>SBDAT02</td>
<td>Output</td>
<td>System Data Output. Bit 2.</td>
</tr>
<tr>
<td>SBDAT03</td>
<td>Output</td>
<td>System Data Output. Bit 3.</td>
</tr>
<tr>
<td>SBDAT05</td>
<td>Output</td>
<td>System Data Output. Bit 5.</td>
</tr>
<tr>
<td>SBACKO</td>
<td>Output</td>
<td>System Acknowledgement.</td>
</tr>
<tr>
<td>I2CIRQ</td>
<td>Output</td>
<td>I²C Interrupt output.</td>
</tr>
<tr>
<td>I2CWKUP</td>
<td>Output</td>
<td>I²C Wake Up from Standby signal.</td>
</tr>
<tr>
<td>SCLI</td>
<td>Input</td>
<td>Serial Clock Input.</td>
</tr>
<tr>
<td>SCLLO</td>
<td>Output</td>
<td>Serial Clock Output</td>
</tr>
<tr>
<td>SCLOE</td>
<td>Output</td>
<td>Serial Clock Output Enable. Active High.</td>
</tr>
<tr>
<td>SDAI</td>
<td>Input</td>
<td>Serial Data Input</td>
</tr>
<tr>
<td>SDAO</td>
<td>Output</td>
<td>Serial Data Output</td>
</tr>
<tr>
<td>SDAOE</td>
<td>Output</td>
<td>Serial Data Output Enable. Active High.</td>
</tr>
</tbody>
</table>

Parameters

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

Table 7.4. SB_I2C Parameters

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

Synthesis Attribute

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

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

SB_I2C i2cInst0 (  
  .SBCLKI(sbclk),  
  .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),  
  .SBATO7(sbdato[7]),  
  .SBATO6(sbdato[6]),  
  .SBATO5(sbdato[5]),  
  .SBATO4(sbdato[4]),  
  .SBATO3(sbdato[3]),  
  .SBATO2(sbdato[2]),  
  .SBATO1(sbdato[1]),  
  .SBATO0(sbdato[0]),  
  .SBACKO(sbacko),  
  .I2CIRQ(i2cirq),  
  .I2CWKUP(i2cwkup),  
  .SCLO(sclo),  
  .SCLOE(scloe),  
  .SDAO(sdao),  
  .SDAOE(sdao)  
)  
  /* synthesis I2C_CLK_DIVIDER= 1 */;

defparam i2cInst0.I2C_SLAVE_INIT_ADDR = "0b1111100001";
defparam i2cInst0.BUS_ADDR74 = "0b0001";
7.1.4. SB_SPI

The SPI hard IP provide industry standard four-pin communication interface with 8-bit wide System Bus to communicate with System Host. It could be configured as 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, and others for easy communicate with system host. The iCE40LM device supports two SPI hard IP primitives, located at lower left corner and lower right corner of the chip.

![Figure 7.4. SB_SPI](image)

### Table 7.5. SB_SPI Port Signals

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

**Parameters**

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

**Table 7.6. SB_SPI Parameters**

<table>
<thead>
<tr>
<th>SPI Location</th>
<th>Parameter</th>
<th>Parameter Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Left Corner</td>
<td>BUS_ADDR74</td>
<td>0b0000</td>
<td>Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0b0010</td>
<td>Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.</td>
</tr>
</tbody>
</table>

**Synthesis Attribute**

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

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

SB_SPI

```verilog
SB_SPI spiInst0 {
  .SBCLKI(sbclkki),
  .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(sbcko),
  .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 = "0b00000";
```
7.2. iCE40 Ultra (iCE5LP) Hard Macros

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

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

7.2.1. SB_HFOSC

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

Table 7.7. SB_HFOSC Port Signals

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

Table 7.8. SB_HFOSC Parameters

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

Default Signal Values

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

- Input CLKHFEN: Logic “0”
- Input CLKHFPU: Logic “0”

Clock Network Selection

By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.
Synthesis Attributes
/* synthesis ROUTE_THROUGH_FABRIC = <value> */ Value:
0: Use dedicated clock network. Default option.
1: Use fabric routes.

Verilog Instantiation
SB_HFOSC OSCInst0 ( .CLKHFEN(ENCLKHF), .CLKHFPUS(CLKHF_POWERUP), .CLKHF(CLKHF) ) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
defparam OSCInst0.CLKHF_DIV = "0b00";

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

Table 7.9. SB_LFOSC Port Signals

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

Default Signal Values
The iCEcube2 software assigns the following signal value to unconnected port:
Input CLKLFEN: Logic "0"
Input CLKLFPU: Logic "0"

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

Synthesis Attributes
/* synthesis ROUTE_THROUGH_FABRIC = <value> */
Value:
0: Use dedicated clock network. Default option.
1: Use fabric routes.
Verilog Instantiation

```
SB_LFOSC_OSCInst1 (  
    .CLKLFIN(ENCLKLF),  
    .CLKLPUB(CLKLF_POWERUP),  
    .CLKLF(CLKLF)  
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

7.2.3. SB_LED_DRV_CUR

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

![SB_LED_DRV_CUR Diagram](image)

**Table 7.10. SB_LED_DRV_CUR Port Signals**

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

Default Signal Values

The iCeCube2 software assigns the following signal value to unconnected port:

Input EN: Logic “0”

Verilog Instantiation

```
SB_LED_DRV_CUR 를 LED_CUR_inst(  
    .EN(enable_led_current),  
    .LEDPU(led_power_up)  
) ;
```
7.2.4. SB_RGB_DRV
SB_RGB_DRV primitive contains three dedicated open drain I/O pins for RGB LED outputs. Each of the RGB LED output is bonded out together with an SB_IO_OD primitive to the package pin. You can either use SB_RGB_DRV primitive or the SB_IO_OD primitive to drive the package pin, but not both.
The primitive allows configuration of each of the 3 RGB LED outputs individually. When the RGBx_CURRENT parameter of RGBx output is set to "0b000000", then SB_IO_OD can be used to drive the package pin.

![Figure 7.8. SB_RGB_DRV](image)

**Table 7.11. SB_RGB_DRV Port Signals**

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

**Default Signal Values**
The iCEcube2 software assigns the following signal value to unconnected port:
Input RGBLEDEN : Logic “0”
Input RGB0PWM : Logic “0”
Input RGB1PWM : Logic “0”
Input RGB2PWM : Logic “0”
Input RGBPU : Logic “0”
Parameters
The SB_RGB_DRV primitive contains the following parameter and their default values:
Parameter RGB0_CURRENT = "0b000000";
Parameter RGB1_CURRENT = "0b000000";
Parameter RGB2_CURRENT = "0b000000";
Parameter values:
"0b000000" = 0 mA. // Set this value to use the associated SB_IO_OD instance at RGB
// LED location.
"0b000001" = 4 mA.
"0b000011" = 8 mA.
"0b000111" = 12 mA.
"0b001111" = 16 mA.
"0b011111" = 20 mA.
"0b111111" = 24 mA.

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

defparam RGB_DRIVER.RGB0_CURRENT = "0b111111"; defparam RGB_DRIVER.RGB1_CURRENT = "0b111111"; defparam RGB_DRIVER.RGB2_CURRENT = "0b111111";
7.2.5. SB_IR_DRV

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

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

![SB_IR_DRV Diagram]

Figure 7.9. SB_IR_DRV

Table 7.12. SB_IR_DRV Port Signals

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

Default Signal Values

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

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

Parameter

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

Parameter IR_CURRENT = "0b0000000000";

Parameter Values:

"0b00000000000"; = 0 mA. // Set this value to use the associated SB_IO_OD instance at // IR LED location.
"0b00000000001"; = 50 mA
"0b00000000011"; = 100 mA
"0b00000000111"; = 150 mA
"0b00000001111"; = 200 mA
"0b00000011111"; = 250 mA
"0b00000111111"; = 300 mA
"0b00001111111"; = 350 mA
"0b00111111111"; = 400 mA
"0b01111111111"; = 450 mA
"0b11111111111"; = 500 mA
Verilog Instantiation

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

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

7.2.6. SB_RGB_IP

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

![SB_RGB_IP Diagram]

**Figure 7.10. SB_RGB_IP**

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

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

Verilog Instantiation

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

7.2.7. SB_IO_OD

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

![Figure 7.11. SB_IO_OD](image-url)
Table 7.14. SB_IO_OD Port Signals

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

Default Signal Values

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

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

Parameter Values

Parameter PIN_TYPE = 6'b000000;
   // See Input and Output Pin Function Tables in SB_IO.
   // Default value of PIN_TYPE = 6'b000000

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

Input and Output Pin Function Tables

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

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

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

### 7.2.8. SB_I2C

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

[Figure 7.12. SB_I2C](#)
Table 7.15. SB_I2C Port Signals

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

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

Table 7.16. SB_I2C Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Left Corner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2C_SLAVE_INIT_ADDR</td>
<td>0b111100001</td>
<td>Upper Bits &lt;9:2&gt; can be changed through control registers. Lower bits &lt;1:0&gt; are fixed.</td>
</tr>
<tr>
<td>BUS_ADDR74</td>
<td>0b0001</td>
<td>Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.</td>
</tr>
<tr>
<td>Upper Right Corner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2C_SLAVE_INIT_ADDR</td>
<td>0b111100010</td>
<td>Upper Bits &lt;9:2&gt; can be changed through control registers. Lower bits &lt;1:0&gt; are fixed.</td>
</tr>
<tr>
<td>BUS_ADDR74</td>
<td>0b0011</td>
<td>Fixed value. SBADRI [7:4] bits also should match with this value to activate the IP.</td>
</tr>
</tbody>
</table>

Synthesis Attribute

I2C_CLK_DIVIDER

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

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

SDA_INPUT_DELAYED

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

/* synthesis SDA_INPUT_DELAYED= value */

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

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

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

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

### 7.2.9. SB_SPI

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

![SB_SPI Diagram](image)

**Figure 7.13. SB_SPI**

#### Table 7.17. SB_SPI Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCLKI</td>
<td>Input</td>
<td>System Clock input.</td>
</tr>
<tr>
<td>SBRWI</td>
<td>Input</td>
<td>System Read/Write Input.</td>
</tr>
<tr>
<td>SBSTBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>SBADRI0</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 0.</td>
</tr>
<tr>
<td>SBADRI1</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 1.</td>
</tr>
<tr>
<td>SBADRI2</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 2.</td>
</tr>
<tr>
<td>SBADRI3</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 3.</td>
</tr>
<tr>
<td>SBADRI5</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 5.</td>
</tr>
<tr>
<td>SBADRI6</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 6.</td>
</tr>
<tr>
<td>SBADRI7</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 7.</td>
</tr>
<tr>
<td>SBDATO1</td>
<td>Input</td>
<td>System Data Input. Bit 0.</td>
</tr>
<tr>
<td>SBDATO2</td>
<td>Input</td>
<td>System Data Input. Bit 1.</td>
</tr>
<tr>
<td>SBDATO3</td>
<td>Input</td>
<td>System Data Input. Bit 2.</td>
</tr>
<tr>
<td>SBDATO6</td>
<td>Input</td>
<td>System Data Input. Bit 5.</td>
</tr>
<tr>
<td>SBDATO7</td>
<td>Input</td>
<td>System Data Input. Bit 6.</td>
</tr>
<tr>
<td>SBDAT05</td>
<td>Input</td>
<td>System Data Input. Bit 7.</td>
</tr>
<tr>
<td>SBDAT06</td>
<td>Input</td>
<td>System Data Input. Bit 8.</td>
</tr>
<tr>
<td>SBDAT08</td>
<td>Input</td>
<td>System Data Input. Bit 10.</td>
</tr>
<tr>
<td>SBDAT09</td>
<td>Input</td>
<td>System Data Input. Bit 11.</td>
</tr>
<tr>
<td>SBDAT10</td>
<td>Input</td>
<td>System Data Input. Bit 12.</td>
</tr>
<tr>
<td>SBDAT14</td>
<td>Input</td>
<td>System Data Input. Bit 16.</td>
</tr>
<tr>
<td>SBDAT15</td>
<td>Input</td>
<td>System Data Input. Bit 17.</td>
</tr>
<tr>
<td>SBDAT16</td>
<td>Input</td>
<td>System Data Input. Bit 18.</td>
</tr>
<tr>
<td>SCNI</td>
<td>Input</td>
<td>System Control signal.</td>
</tr>
<tr>
<td>MSCNO0</td>
<td>Input</td>
<td>System Master clock input. Bit 0.</td>
</tr>
<tr>
<td>MSCNO1</td>
<td>Input</td>
<td>System Master clock input. Bit 1.</td>
</tr>
<tr>
<td>MSCNO2</td>
<td>Input</td>
<td>System Master clock input. Bit 2.</td>
</tr>
<tr>
<td>MSCNO3</td>
<td>Input</td>
<td>System Master clock input. Bit 3.</td>
</tr>
<tr>
<td>MSCNOE0</td>
<td>Input</td>
<td>System Master clock enable input. Bit 0.</td>
</tr>
<tr>
<td>MSCNOE1</td>
<td>Input</td>
<td>System Master clock enable input. Bit 1.</td>
</tr>
<tr>
<td>MSCNOE2</td>
<td>Input</td>
<td>System Master clock enable input. Bit 2.</td>
</tr>
<tr>
<td>MSCNOE3</td>
<td>Input</td>
<td>System Master clock enable input. Bit 3.</td>
</tr>
</tbody>
</table>
### Signal Name | Direction | Description
--- | --- | ---
SBDATI1 | Input | System Data input. Bit 1.
SBDATI2 | Input | System Data input. Bit 2.
SBDATI3 | Input | System Data input. Bit 3.
SBDATI5 | Input | System Data input. Bit 5.
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.
SBDATO5 | Input | System Data Output. Bit 5.
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
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.

**Table 7.18. SB_SPI Parameters**

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

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

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

### 7.2.10. SB_MAC16

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

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

![SB_MAC16 Diagram](image)

**Figure 7.14. SB_MAC16**
Table 7.19. SB_MAC16 Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Input</td>
<td>Clock input. Applies to all clocked elements in the MAC16 Block.</td>
<td>1'b1</td>
</tr>
<tr>
<td>CE</td>
<td>Input</td>
<td>Clock Enable Input. Active High.</td>
<td></td>
</tr>
<tr>
<td>C[15:0]</td>
<td>Input</td>
<td>Input to the C Register / Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>A[15:0]</td>
<td>Input</td>
<td>Input to the A Register / Direct input to the multiplier blocks /Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>B[15:0]</td>
<td>Input</td>
<td>Input to the B Register / Direct input to the multiplier blocks /Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>Input</td>
<td>Input to the D Register / Direct input to the adder accumulator.</td>
<td>16'b0</td>
</tr>
<tr>
<td>AHO LD</td>
<td>Input</td>
<td>Hold A registers Data. Controls data flow into the input register A. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.</td>
<td>1'b0</td>
</tr>
<tr>
<td>BHO LD</td>
<td>Input</td>
<td>Hold B registers Data. Controls data flow into the B input register. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.</td>
<td>1'b0</td>
</tr>
<tr>
<td>Signal Name</td>
<td>Direction</td>
<td>Description</td>
<td>Default Values</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>CHOLD</td>
<td>Input</td>
<td>Hold C registers Data. Controls data flow into the A input register. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.</td>
<td>1'b0</td>
</tr>
<tr>
<td>D HOLD</td>
<td>Input</td>
<td>Hold D registers Data. Controls data flow into the A input register. Active High. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.</td>
<td>1'b0</td>
</tr>
<tr>
<td>IRSTTOP</td>
<td>Input</td>
<td>Reset input to the A and C input registers, and the pipeline registers in the upper half of the multiplier block. Active High</td>
<td>1'b0</td>
</tr>
<tr>
<td>IRSTBOT</td>
<td>Input</td>
<td>Reset input to the B and C input registers, and the pipeline registers in the lower half of the multiplier block, and the 32-bit multiplier result pipeline register. Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>ORSTTOP</td>
<td>Input</td>
<td>Reset the high-order bits of the accumulator register ([31:16]). Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>ORSTBOT</td>
<td>Input</td>
<td>Reset the low-order accumulator register bits ([15:0]). Active High.</td>
<td>1'b0</td>
</tr>
<tr>
<td>OLOADTOP</td>
<td>Input</td>
<td>High-order Accumulator Register Accumulate/Load. Controls whether the accumulator register accepts the output of the adder/subtractor or whether the register is loaded with the value from Input C (or Register C, if configured). 0: Accumulator Register [31:16] loaded with output from adder/subtractor. 1: Accumulator Register [31:16] loaded with Input C or Register C, depending on primitive parameter value.</td>
<td>1'b0</td>
</tr>
<tr>
<td>OLOADBOT</td>
<td>Input</td>
<td>Low-order Accumulator Register Accumulate/Load. Controls whether the low-order accumulator register bits [15:0] accepts the output of the adder/subtractor or whether the register is loaded with the value from Input D (or Register D, if configured). 0: Accumulator Register [15:0] loaded with output from adder/subtractor. 1: Accumulator Register [15:0] loaded with Input D or Register D, depending on primitive parameter value.</td>
<td>1'b0</td>
</tr>
<tr>
<td>ADDSUBTOP</td>
<td>Input</td>
<td>High-order Add/Subtract. Controls whether the adder/subtractor adds or subtracts. 0: Add: W+X+HCI 1: Subtract: W-X+HCI</td>
<td>1'b0: Add</td>
</tr>
<tr>
<td>ADDSUBBOT</td>
<td>Input</td>
<td>Low-order Add/Subtract. Controls whether the adder/subtractor adds or subtracts. 0: Add: Y+Z+LCI 1: Subtract: Y-Z+LCI</td>
<td>1'b0: Add</td>
</tr>
<tr>
<td>OHOLDTOP</td>
<td>Input</td>
<td>High-order Accumulator Register Hold. Controls data flow into the high-order ([31:16]) bits of the accumulator. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.</td>
<td></td>
</tr>
<tr>
<td>OHOLDBOT</td>
<td>Input</td>
<td>Low-order Accumulator Register Hold. Controls data flow into the high-order ([15:0]) bits of the accumulator. 0: Update (load) register at next active clock edge. 1: Hold (retain) current register value, regardless of clock.</td>
<td></td>
</tr>
<tr>
<td>CI</td>
<td>Input</td>
<td>Carry/borrow input from lower logic tile.</td>
<td></td>
</tr>
<tr>
<td>ACCUMCI</td>
<td>Input</td>
<td>Cascade Carry/borrow input from lower MAC16 block.</td>
<td></td>
</tr>
<tr>
<td>SIGNEXTIN</td>
<td>Input</td>
<td>Sign extension input from lower MAC16 block.</td>
<td></td>
</tr>
</tbody>
</table>
### Signal Name | Direction | Description | Default Values
--- | --- | --- | ---
O[31:0] | Output | 32-bit MAC16 Output. O[31:0]: 32-bit result of a 16x16 multiply operation or a 32-bit adder/accumulate function. O[31:16]: 16-bit result of an 8x8 multiply operation or a 32-bit adder/accumulate function. O[15:0]: 16-bit result of an 8x8 multiply operation or a 32-bit adder/accumulate function. |
CO | Output | Carry/borrow output to higher logic tile. |
ACCUMCO | Output | Cascade Carry/borrow output to higher MAC16 block. |
SIGNEXTOUT | Output | Sign extension output to higher MAC16 block. |

**Parameters**

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

**Table 7.20. SB_MAC16 Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Configuration Bits</th>
<th>Parameter Description</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG_TRIGGERS</td>
<td></td>
<td>Controls input clock polarity</td>
<td>0</td>
<td>All the registers are rising_edge triggered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>All the registers are falling_edge triggered.</td>
</tr>
<tr>
<td>C_REG</td>
<td>C0</td>
<td>Input C register Control.</td>
<td>0</td>
<td>Input C not registered</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Input C registered</td>
</tr>
<tr>
<td>A_REG</td>
<td>C1</td>
<td>Input A register Control</td>
<td>0</td>
<td>Input A not registered</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Input A registered</td>
</tr>
<tr>
<td>B_REG</td>
<td>C2</td>
<td>Input B register Control</td>
<td>0</td>
<td>Input B not registered</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Input B registered</td>
</tr>
<tr>
<td>D_REG</td>
<td>C3</td>
<td>Input D register Control</td>
<td>0</td>
<td>Input D not registered</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Input D registered</td>
</tr>
<tr>
<td>TOP_8x8_MULT_REG</td>
<td>C4</td>
<td>Top 8x8 multiplier output register control (point F)</td>
<td>0</td>
<td>Top 8x8 multiplier output is not registered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Top 8x8 multiplier output is registered.</td>
</tr>
<tr>
<td>BOT_8x8_MULT_REG</td>
<td>C5</td>
<td>Bottom 8x8 multiplier output register control. (point G)</td>
<td>0</td>
<td>Bottom 8x8 multiplier output is not registered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Bottom 8x8 multiplier output is registered</td>
</tr>
<tr>
<td>PIPELINE_16x16_MULT_REG1</td>
<td>C6</td>
<td>Intermediate 8x8 multiplier pipeline register controls (points J and K). These multipliers are only used for 16x16 multiply operations. For 8x8 multiply operations, set C6 and C7 to 1, which reduces power consumption.</td>
<td>0</td>
<td>Intermediate 8x8 multiplier outputs are not registered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Intermediate 8x8 multiplier outputs are registered.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Configuration Bits</td>
<td>Parameter Description</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------</td>
<td>---------------------------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PIPELINE_16x16_MULT_REG2</td>
<td>C7</td>
<td>16x16 multiplier Pipeline registers control. (point H)</td>
<td>0</td>
<td>32-bit output from 16x16 multiplier is not registered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>32-bit output from 16x16 multiplier is registered.</td>
</tr>
<tr>
<td>TOPOUTPUT_SELECT</td>
<td>C9,C8</td>
<td>Selects Top SB_MAC16 output O[31:16].</td>
<td>00</td>
<td>16-bit output of Multiplexer P, from top adder/subtractor,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>16-bit output from upper accumulator register, Q</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>16-bit output from upper 8x8 multiplier, F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Upper 16-bit output from 16x16 multiplier, H</td>
</tr>
<tr>
<td>TOPADDSUB_LOWERINPUT</td>
<td>C11,C10</td>
<td>Selects input X for the upper adder/subtracter</td>
<td>00</td>
<td>16-bit input from A input or associated input register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>16-bit output from upper 8x8 multiplier, F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>Upper 16-bit output from 16x16 multiplier, H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Sign extension input from lower adder/subtractor, Z[15]. Duplicate 16 bits.</td>
</tr>
<tr>
<td>TOPADDSUB_UPPERINPUT</td>
<td>C12</td>
<td>Selects input W for the upper adder/subtracter</td>
<td>0</td>
<td>16-bit feedback from upper accumulator register, Q</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>16-bit input from C input or associated pipeline register</td>
</tr>
<tr>
<td>TOPADDSUB_CARRYSELECT</td>
<td>C14,C13</td>
<td>Carry/borrow input select to upper adder/subtracter.</td>
<td>00</td>
<td>Constant 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>Constant 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>Cascade Carry/borrow output from lower adder/subtractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Carry/borrow output from lower adder/subtractor</td>
</tr>
<tr>
<td>BOTOUTPUT_SELECT</td>
<td>C16,C15</td>
<td>Selects Lower SB_MAC16 output O[15:0].</td>
<td>00</td>
<td>16-bit output of multiplexer R from lower adder/subtractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>16-bit output from lower accumulator register, S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>16-bit output from lower 8x8 multiplier, G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Lower 16-bit output from 16x16 multiplier, H</td>
</tr>
<tr>
<td>BOTADDSUB_LOWERINPUT</td>
<td>C18,C17</td>
<td>Selects Input Z for the lower adder/subtracter</td>
<td>00</td>
<td>16-bit input from B input or associated pipeline register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>16-bit output from lower 8x8 multiplier, G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>Lower 16-bit output from 16x16 multiplier, H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Sign extension input SIGNEXTIN. Duplicate 16 bits.</td>
</tr>
<tr>
<td>BOTADDSUB_UPPERINPUT</td>
<td>C19</td>
<td>Selects Input Y for the lower adder/subtracter</td>
<td>0</td>
<td>16-bit feedback from lower accumulator register, S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>16-bit input from D input or associated input register.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Configuration Bits</td>
<td>Parameter Description</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BOTADDSUB_CARRYSELECT</td>
<td>C21, C20</td>
<td>Carry/borrow input select to lower adder/subtractor</td>
<td>00</td>
<td>Constant 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>Constant 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>Cascade Carry/borrow input from ACCUMCI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Carry/borrow input from CI</td>
</tr>
<tr>
<td>MODE_8x8</td>
<td>C22</td>
<td>Selects 8x8 Multiplier mode and 8x8 Low-Power Multiplier Blocking Option</td>
<td>0</td>
<td>No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Selects 8x8 Multiplier mode. Holds the pipelining registers associate with a 16x16 multiplier in clock disable mode. Helps reduce the dynamic power consumption within the multiplier function. Used in conjunction with C6, C7 settings.</td>
</tr>
<tr>
<td>A_SIGNED</td>
<td>C23</td>
<td>Indicates whether multiplier input A is signed or unsigned. Applies regardless if input A is 16- or 32-bits wide.</td>
<td>0</td>
<td>Multiplier input A is unsigned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Multiplier input A is signed.</td>
</tr>
<tr>
<td>B_SIGNED</td>
<td>C24</td>
<td>Indicates whether multiplier input B is signed or unsigned. Applies regardless if input B is 16- or 32-bits wide.</td>
<td>0</td>
<td>Multiplier input B is unsigned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Multiplier input B is signed.</td>
</tr>
</tbody>
</table>

7.2.10.1. SB_MAC16 Configurations

The following SB_MAC16 functional blocks are supported in iCEcube2.

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

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

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

8x8 Multiplier: 8x8 input multiplier with 16-bit output.

16x16 Multiplier: 16x16 input multiplier with 32-bit output.

### Table 7.21. Multiplier Configurations

<table>
<thead>
<tr>
<th>SB_MAC16 Function/Parameter Settings</th>
<th>8_SIGNED</th>
<th>A_SIGNED</th>
<th>MODE_8x8</th>
<th>BOTADDSUB__CARRYSELECT</th>
<th>BOTADDSUB__UPPERINPUT</th>
<th>BOTADDSUB__LOWERINPUT</th>
<th>BOTOUTPUT_SELECT</th>
<th>TOPADDSUB__CARRYSELECT</th>
<th>TOPADDSUB__UPPERINPUT</th>
<th>TOPADDSUB__LOWERINPUT</th>
<th>TOPOUTPUT_SELECT</th>
<th>PIPELINE_16x16_MULT_REG2</th>
<th>PIPELINE_16x16_MULT_REG1</th>
<th>BOT__8x8_MULT_REG</th>
<th>TOP__8x8_MULT_REG</th>
<th>D_REG</th>
<th>B_REG</th>
<th>A_REG</th>
<th>C_REG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>8x8 Multiplier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_all_pipelined_unsigned</td>
<td>0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_all_pipelined_signed</td>
<td>1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_bypass_unsigned</td>
<td>0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_8x8_bypass_signed</td>
<td>1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>16x16 Multiplier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_all_pipelined_unsigned</td>
<td>0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_all_pipelined_signed</td>
<td>0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_intermediate_register_bypassed_unsigned</td>
<td>1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_intermediate_register_bypassed_signed</td>
<td>0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_bypass_unsigned</td>
<td>1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mult_16x16_bypass_signed</td>
<td>0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MAC Configurations

16-bit MAC: 8x8 input, 16-bit output MAC.

32-bit MAC: 16x16 input, 32-bit output MAC.

#### Table 7.22. MAC Configurations

<table>
<thead>
<tr>
<th>SB_MAC16 Function/Parameter Settings</th>
<th>B.SIGNED</th>
<th>A.SIGNED</th>
<th>MODE</th>
<th>8x8</th>
<th>BOTADDSUB.CARRYSELECT</th>
<th>BOTADDSUB.UPPERINPUT</th>
<th>BOTADDSUB.LOWERINPUT</th>
<th>BOTOUTPUT_SELECT</th>
<th>TOPADDSUB.CARRYSELECT</th>
<th>TOPADDSUB.UPPERINPUT</th>
<th>TOPADDSUB.LOWERINPUT</th>
<th>TOPOUTPUT_SELECT</th>
<th>PIPELINE.16x16.MULT_REG2</th>
<th>PIPELINE.8x8.MULT_REG1</th>
<th>BOT.8x8.MULT_REG</th>
<th>TOP.8x8.MULT_REG</th>
<th>D_REG</th>
<th>B_REG</th>
<th>A_REG</th>
<th>C_REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit MAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_16_all_pipeline_en</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_16_intermediate_register_</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_16_bypassed_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit MAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_all_pipeline_en</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_all_pipeline_cascaded_</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_unsigned</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_bypassed_cascaded_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_cin_</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mac_32_intermediate_register</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_cascaded_unsigned</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bypassed_cin_</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
ACCUMULATOR Configurations

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

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

Table 7.23. ACCUMULATOR Configurations

<table>
<thead>
<tr>
<th>SB_MAC16 Function/ Parameter Settings</th>
<th>B_SIGNED</th>
<th>A_SIGNED</th>
<th>MODE_8X8</th>
<th>BOTADDSUB_CARRYSELECT</th>
<th>BOTADDSUB_UPPERINPUT</th>
<th>BOTADDSUB_LOWERINPUT</th>
<th>BOTOUTPUT_SELECT</th>
<th>TOPADDSUB_CARRYSELECT</th>
<th>TOPADDSUB_UPPERINPUT</th>
<th>TOPADDSUB_LOWERINPUT</th>
<th>TOPOUTPUT_SELECT</th>
<th>PIPELINE_16x16_MULT_REG2</th>
<th>PIPELINE_16x16_MULT_REG1</th>
<th>D_REG</th>
<th>B_REG</th>
<th>A_REG</th>
<th>C_REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit ACCUMULATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acc_16_all_pipelined_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acc_16_bypassed_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32-bit ACCUMULATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acc_32_all_pipelined_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acc_32_all_pipelined_cascaded_uns</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acc_32_all_pipelined_cin_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>acc_32_bypassed_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>acc_32_bypassed_cascaded_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>acc_32_bypassed_cin_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Add/Subtract Configurations  
16-bit ADDSUB: 16-bit input, 16-bit output ADDSUB.  
32-bit ADDSUB: 32-bit input, 32-bit output ADDSUB.  

<table>
<thead>
<tr>
<th>SB_MAC16 Function/ Parameter Settings</th>
<th>B_SIGNED</th>
<th>A_SIGNED</th>
<th>MODE_8x8</th>
<th>BOTADDSUB__CARRYSELECT</th>
<th>BOTADDSUB__UPPERINPUT</th>
<th>BOTADDSUB__LOWERINPUT</th>
<th>BOTOUTPUT_SELECT</th>
<th>TOPADDSUB__CARRYSELECT</th>
<th>TOPADDSUB__UPPERINPUT</th>
<th>TOPADDSUB__LOWERINPUT</th>
<th>TOPOUTPUT_SELECT</th>
<th>PIPELINE_16x16__MULT_REG2</th>
<th>PIPELINE_16x16__MULT_REG1</th>
<th>BOT_8x8__MULT_REG</th>
<th>D_REG</th>
<th>B_REG</th>
<th>A_REG</th>
<th>C_REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit ADDSUB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_16_all_pipelined_unsigned</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_16_bypassed_unsigned</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit ADDSUB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_32_all_pipelined_unsigned</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_32_all_pipelined_cascaded_unsigned</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_32_all_pipelined_cin_unsigned</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_32_bypassed_unsigned</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_32_bypassed_cascaded_unsigned</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_sub_32_bypassed_cin_unsigned</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
### Multiply Add/Subtract Configurations

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

Table 7.25. Multiply Add/Subtract Configurations

| SB_MAC16          | B_SIGNED | A_SIGNED | MODE | 8×8 | BOTADDSUB_CARRYSELECT | BOTADDSUB_UPPERINPUT | BOTADDSUB_LOWERINPUT | TOPADDSUB_CARRYSELECT | TOPADDSUB_UPPERINPUT | TOPADDSUB_LOWERINPUT | TOPOUTPUT_SELECT | PIPELINE_16x16 | MULT_REG1 | MULT_REG2 | B_REG | A_REG | C_REG |
|-------------------|----------|----------|------|-----|------------------------|----------------------|----------------------|-----------------------|----------------------|----------------------|----------------|-----------|----------|--------|-------|-------|
| 16-bit Mult-Add/Sub |          |          |      |     |                        |                      |                      |                       |                      |                      |                |           |          |        |       |       |
| mult_add_sub_16_all_pipeline unsigned | 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_16_intermediate_register_bypassed unsigned | 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 1 1 1 1 1 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_16_bypassed unsigned | 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 | | | | | | | | | | | | | | | | | | |
| 32-bit Mult-Add/Sub |          |          |      |     |                        |                      |                      |                       |                      |                      |                |           |          |        |       |       |
| mult_add_sub_32_all_pipeline unsigned | 0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_32_all_pipeline_cascaded unsigned | 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_32_all_pipeline_cin unsigned | 0 0 0 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_32_intermediate_register_bypassed unsigned | 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1 1 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_32_intermediate_register_bypassed_cascaded unsigned | 0 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_32_intermediate_register_bypassed_cin unsigned | 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 | | | | | | | | | | | | | | | | | | |
| mult_add_sub_32_intermediate_register_bypassed signed | 1 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | |
Verilog Instantiation

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

// mult_8x8_all_pipelined_unsigned [24:0] = 001_0000010_0000010_0111_0110
// Read configuration settings [24:0] from left to right while filling the instance parameters.
defparam i_sbmac16. B_SIGNED = 1'b0;
defparam i_sbmac16. A_SIGNED = 1'b0;
defparam i_sbmac16. MODE_8x8 = 1'b1;
defparam i_sbmac16. BOTADDSUB_CARRYSELECT = 2'b00;
defparam i_sbmac16. BOTADDSUB_UPPERINPUT = 1'b0;
defparam i_sbmac16. BOTADDSUB_LOWERINPUT = 2'b00;
defparam i_sbmac16. BOTOUTPUT_SELECT = 2'b10;
defparam i_sbmac16. TOPADDSUB_CARRYSELECT = 2'b00;
defparam i_sbmac16. TOPADDSUB_UPPERINPUT = 1'b0;
defparam i_sbmac16. TOPADDSUB_LOWERINPUT = 2'b00;
defparam i_sbmac16. TOPOUTPUT_SELECT = 2'b10;
defparam i_sbmac16. PIPELINE_16x16_MULT_REG2 = 1'b0;
7.3. iCE40 UltraLite (iCE40UL) Hard Macros

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

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

7.3.1. SB_HFOSC

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

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

![Figure 7.16. SB_HFOSC](image)

Table 7.26. SB_HFOSC Port Signals

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKHF_DIV</td>
<td>&quot;0b00&quot;</td>
<td>Sets 48 MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b01&quot;</td>
<td>Sets 24 MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b10&quot;</td>
<td>Sets 12 MHz HFOSC output.</td>
</tr>
<tr>
<td></td>
<td>&quot;0b11&quot;</td>
<td>Sets 6 MHz HFOSC output.</td>
</tr>
</tbody>
</table>
Default Signal Values
The iCEcube2 software assigns the following signal value to unconnected port:
- Input CLKHFEN: Logic “0”
- Input CLKHFPU: Logic “0”

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

Synthesis Attributes
/* synthesis ROUTE_THROUGH_FABRIC = <value> */ Value:
0: Use dedicated clock network. Default option.
1: Use fabric routes.

Verilog Instantiation
```
SB_HFOSC OSCInst0 (  
    .CLKHFEN(ENCLKHF),
    .CLKHFPU(CLKHF_POWERUP),
    .CLKHF(CLKHF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```
defparam OSCInst0.CLKHF_DIV = "0b00";

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

![Figure 7.17. SB_LFOSC](image)

Table 7.28. SB_LFOSC Port Signals
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKLPU</td>
<td>Input</td>
<td>Power up the LFOSC circuit. After power up, the oscillator output is stable after 100 µs. Active High.</td>
</tr>
<tr>
<td>CLKLKEN</td>
<td>Input</td>
<td>Enable the clock output. Enable should be low for the 100 µs power up period. Active High.</td>
</tr>
<tr>
<td>CLKLF</td>
<td>Output</td>
<td>LF Oscillator output.</td>
</tr>
</tbody>
</table>

Default Signal Values
The iCEcube2 software assigns the following signal value to unconnected port:
- Input CLKLKEN: Logic “0”
- Input CLKLPU: Logic “0”
Clock Network Selection
By default, the oscillator use one of the dedicated clock networks in the device to drive the elements. You may configure the oscillator to use the fabric routes instead of global clock network using the synthesis attribute.

Synthesis Attributes
/* synthesis ROUTE_THROUGH_FABRIC = <value> */ Value:
0: Use dedicated clock network. Default option.
1: Use fabric routes.

Verilog Instantiation
SB_LFOSC OSCInst1 {
  .CLKLFEN(ENCLKLF),
  .CLKLFPU(CLKL_POWERUP),
  .CLKLF(CLKL)
} /* synthesis ROUTE_THROUGH_FABRIC=[0|1] */;

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

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

```
SB_RGBA_DRV
    CURREN
    RGBLEDEN
    RGB0PWM
    RGB1PWM
    RGB2PWM

    RGB0
    RGB1
    RGB2
```

Figure 7.18. SB_RGBA_DRV

Table 7.29. SB_RGBA_DRV Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURREN</td>
<td>Input</td>
<td>Enable the mixed signal control block to supply reference current to the IR drivers. When it is not enabled, (CURREN=0), no current is supplied, and the IR drivers are powered down. Enabling the mixed signal control block takes 100 µs to reach a stable reference current value.</td>
</tr>
<tr>
<td>RGBLEDEN</td>
<td>Input</td>
<td>Enable the SB_RGBA_DRV primitive. Active High.</td>
</tr>
<tr>
<td>RGB0PWM</td>
<td>Input</td>
<td>Input data to drive RGB0 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
</tr>
<tr>
<td>RGB1PWM</td>
<td>Input</td>
<td>Input data to drive RGB1 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
</tr>
<tr>
<td>RGB2PWM</td>
<td>Input</td>
<td>Input data to drive RGB2 LED pin. This input is usually driven from the SB_LEDD_IP.</td>
</tr>
<tr>
<td>RGB0</td>
<td>Output</td>
<td>RGB0 LED output.</td>
</tr>
<tr>
<td>RGB1</td>
<td>Output</td>
<td>RGB1 LED output.</td>
</tr>
<tr>
<td>RGB2</td>
<td>Output</td>
<td>RGB2 LED output.</td>
</tr>
</tbody>
</table>
### Default Signal Values
The iCEcube2 software assigns the following signal value to unconnected port:

- **Input CURREN**: Logic “0”
- **Input RBGLEDEN**: Logic “0”
- **Input RGB0PWM**: Logic “0”
- **Input RGB1PWM**: Logic “0”
- **Input RGB2PWM**: Logic “0”

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

- **Parameter CURRENT_MODE = "0b0"**; Parameter values:
  - "0b0" = Full Current Mode (Default).
  - "0b1" = Half Current Mode.

- **Parameter RGB0_CURRENT = "0b000000"**;
- **Parameter RGB1_CURRENT = "0b000000"**;
- **Parameter RGB2_CURRENT = "0b000000"**;

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

### Verilog Instantiation
```verilog
SB_RGBA_DRV RGBA_DRIVER (  
    .CURREN(ENABLE_CURR),  
    .RBGLEDEN(ENABLE_RGBDRV),  
    .RGB0PWM(RGB0),  
    .RGB1PWM(RGB1),  
    .RGB2PWM(RGB2),  
    .RGB0(LED0),  
    .RGB1(LED1),  
    .RGB2(LED2)  
);  
defparam RGBA_DRIVER.CURRENT_MODE = "0b0"; defparam RGBA_DRIVER.RGB0_CURRENT = "0b111111"; defparam RGBA_DRIVER.RGB1_CURRENT = "0b111111"; defparam RGBA_DRIVER.RGB2_CURRENT = "0b111111";
```
7.3.4. **SB_IR400_DRV**

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

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

![Figure 7.19. SB_IR400_DRV](image)

**Table 7.30. SB_IR400_DRV Port Signals**

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

**Default Signal Values**

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

- **Input CURREN**: Logic “0”
- **Input IRLEDEN**: Logic “0”
- **Input IRPWM**: Logic “0”
Parameter
The SB_IR400_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0";
Parameter values:
"0b0" = Full Current Mode (Default).
"0b1" = Half Current Mode.

Parameter IR400_CURRENT = "0b000000000";
Parameter Values:
"0b0000000000" = 0 mA. //This is the setting to tristate the IR output to allow it to be used as GPIO (SB_IO_OD)
"0b000000001"; = 50 mA for Full Mode; 25 mA for Half Mode.
"0b000000011"; = 100 mA for Full Mode; 50 mA for Half Mode.
"0b000000111"; = 150 mA for Full Mode; 75 mA for Half Mode.
"0b000001111"; = 200 mA for Full Mode; 100 mA for Half Mode.
"0b000011111"; = 250 mA for Full Mode; 125 mA for Half Mode.
"0b000111111"; = 300 mA for Full Mode; 150 mA for Half Mode.
"0b011111111"; = 350 mA for Full Mode; 175 mA for Half Mode.
"0b111111111"; = 400 mA for Full Mode; 200 mA for Half Mode.

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

defparam IRDRVinst.CURRENT_MODE = "0b0";
defparam IRDRVinst.IR400_CURRENT = "0b1111111111";
7.3.5. SB_BARCODE_DRV

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

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

![Figure 7.20. SB_BARCODE_DRV](image)

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

**Default Signal Values**

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

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

**Parameter**

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

Parameter CURRENT_MODE = "0b0";
Parameter values:
"0b0" = Full Current Mode (Default).
"0b1" = Half Current Mode.

Parameter BARCODE_CURRENT = "0b0000";
Parameter values:
"0b0000" = 0 mA. //This is the setting to tri-state the BARCODE output to allow it to be used // as GPIO (SB_IO_OD)
"0b0001" = 16.6 mA for Full Mode; 8.3 mA for Half Mode,
"0b0011" = 33.3 mA for Full Mode; 16.6 mA for Half Mode,
"0b1001" = 66.6 mA for Full Mode; 33.3 mA for Half Mode,
"0b1010" = 83.3 mA for Full Mode; 41.6 mA for Half Mode,
"0b0111" = 50 mA for Full Mode; 25 mA for Half Mode,
"0b1111" = 100 mA for Full Mode; 50 mA for Half Mode
Verilog Instantiation

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

7.3.6. SB_IR500_DRV

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

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

![SB_IR500_DRV](image)

**Figure 7.21. SB_IR500_DRV**

SB_IR500_DRV DRC Rule

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

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

Default Signal Values

The iCeCube2 software assigns the following signal value to unconnected port:

- Input CURREN : Logic “0”
- Input IRLEDEN : Logic “0”
- Input IRPWM  : Logic “0”
Parameters
The SB_IR500_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0"
Parameter values:
"0b0" = Full Current Mode (Default).
"0b1" = Half Current Mode.

Parameter IR500_CURRENT = “0b000000000000”
Parameter values:
"0b000000000000" = 0 mA. // This is the setting to tristate the BARCODE output to allow it to // be used as GPIO (SB_IO_OD).
"0b000000000111" = 50 mA for Full Mode; 25 mA for Half Mode.
"0b000000001111" = 100 mA for Full Mode; 50 mA for Half Mode.
"0b000000011111" = 150 mA for Full Mode; 75 mA for Half Mode.
"0b000000111111" = 200 mA for Full Mode; 100 mA for Half Mode.
"0b000001111111" = 250 mA for Full Mode; 125 mA for Half Mode.
"0b000011111111" = 300 mA for Full Mode; 150 mA for Half Mode.
"0b000111111111" = 350 mA for Full Mode; 175 mA for Half Mode.
"0b001111111111" = 400 mA for Full Mode; 200 mA for Half Mode.
"0b011111111111" = 450 mA for Full Mode; 225 mA for Half Mode.
"0b111111111111" = 500 mA for Full Mode; 250 mA for Half Mode.

Verilog Instantiation
SB_IR500_DRV IRDRVinst (  
  .CU
  .RREN(ENABLE_CURRENT),
  .IRLEDEN(ENABLE_IRDRV),
  .IRPWM(IR_PWMINPUT),
  .IRLED(IR_LEDOUT)
);
defparam IRDRVinst.CURRENT_MODE = "0b0";
defparam IRDRVinst.IR500_CURRENT = "0b1111111111";
7.3.7. SB_LEDDA_IP

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

![SB_LEDDA_IP Diagram]

**Table 7.33. SB_LEDDA_IP Port Signals**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEDDCS</td>
<td>Input</td>
<td>CS to write LEDD IP registers</td>
</tr>
<tr>
<td>LEDDCLK</td>
<td>Input</td>
<td>Clock to write LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT7</td>
<td>Input</td>
<td>Bit 7 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT6</td>
<td>Input</td>
<td>Bit 6 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT5</td>
<td>Input</td>
<td>Bit 5 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT4</td>
<td>Input</td>
<td>Bit 4 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT3</td>
<td>Input</td>
<td>Bit 3 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT2</td>
<td>Input</td>
<td>Bit 2 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT1</td>
<td>Input</td>
<td>Bit 1 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT0</td>
<td>Input</td>
<td>Bit 0 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDADDR3</td>
<td>Input</td>
<td>LEDD IP register address bit 3</td>
</tr>
<tr>
<td>LEDDADDR2</td>
<td>Input</td>
<td>LEDD IP register address bit 2</td>
</tr>
<tr>
<td>LEDDADDR1</td>
<td>Input</td>
<td>LEDD IP register address bit 1</td>
</tr>
<tr>
<td>LEDDADDR0</td>
<td>Input</td>
<td>LEDD IP register address bit 0</td>
</tr>
<tr>
<td>LEDDEN</td>
<td>Input</td>
<td>Data enable input to indicate data and address are stable.</td>
</tr>
<tr>
<td>LEDDEXE</td>
<td>Input</td>
<td>Enable the IP to run the blinking sequence. When it is LOW, the sequence stops at the nearest OFF state.</td>
</tr>
<tr>
<td>LEDDRST</td>
<td>Input</td>
<td>Device level reset signal to reset all internal registers during the device configuration. This port is not accessible to user signals.</td>
</tr>
<tr>
<td>PWMOUT0</td>
<td>Output</td>
<td>PWM output 0</td>
</tr>
<tr>
<td>PWMOUT1</td>
<td>Output</td>
<td>PWM output 1</td>
</tr>
<tr>
<td>PWMOUT2</td>
<td>Output</td>
<td>PWM output 2</td>
</tr>
<tr>
<td>LEDDON</td>
<td>Output</td>
<td>Indicating the LED is on</td>
</tr>
</tbody>
</table>
Default Signal Values
The iCEcube2 software assigns the logic “0” value to all unconnected input ports.

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

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

![Figure 7.23. SB_IR_IP](image-url)
### Table 7.34. SB_IR_IP Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKI</td>
<td>Input</td>
<td>Clock input for IR IP</td>
</tr>
<tr>
<td>CSI</td>
<td>Input</td>
<td>Select Signal. Active High to activate the IP. This usually connects to the output of the decoding logic from MSB of the address bus</td>
</tr>
<tr>
<td>DENI</td>
<td>Input</td>
<td>Data Enable. When asserted, indicates that the data and address on the IR Transceiver Control Bus are stabilized and ready to be captured.</td>
</tr>
<tr>
<td>WEI</td>
<td>Input</td>
<td>Data Write Enable. Asserted during WRITE and deasserted during READ cycle.</td>
</tr>
<tr>
<td>ADR13</td>
<td>Input</td>
<td>Control Register Address Bit 3</td>
</tr>
<tr>
<td>ADR12</td>
<td>Input</td>
<td>Control Register Address Bit 2</td>
</tr>
<tr>
<td>ADR11</td>
<td>Input</td>
<td>Control Register Address Bit 1</td>
</tr>
<tr>
<td>ADR10</td>
<td>Input</td>
<td>Control Register Address Bit 0</td>
</tr>
<tr>
<td>WDATA7</td>
<td>Input</td>
<td>Write Data Input Bit 7</td>
</tr>
<tr>
<td>WDATA6</td>
<td>Input</td>
<td>Write Data Input Bit 6</td>
</tr>
<tr>
<td>WDATA5</td>
<td>Input</td>
<td>Write Data Input Bit 5</td>
</tr>
<tr>
<td>WDATA4</td>
<td>Input</td>
<td>Write Data Input Bit 4</td>
</tr>
<tr>
<td>WDATA3</td>
<td>Input</td>
<td>Write Data Input Bit 3</td>
</tr>
<tr>
<td>WDATA2</td>
<td>Input</td>
<td>Write Data Input Bit 2</td>
</tr>
<tr>
<td>WDATA1</td>
<td>Input</td>
<td>Write Data Input Bit 1</td>
</tr>
<tr>
<td>WDATA0</td>
<td>Input</td>
<td>Write Data Input Bit 0</td>
</tr>
<tr>
<td>RDATA7</td>
<td>Output</td>
<td>Read Data Output Bit 7</td>
</tr>
<tr>
<td>RDATA6</td>
<td>Output</td>
<td>Read Data Output Bit 6</td>
</tr>
<tr>
<td>RDATA5</td>
<td>Output</td>
<td>Read Data Output Bit 5</td>
</tr>
<tr>
<td>RDATA4</td>
<td>Output</td>
<td>Read Data Output Bit 4</td>
</tr>
<tr>
<td>RDATA3</td>
<td>Output</td>
<td>Read Data Output Bit 3</td>
</tr>
<tr>
<td>RDATA2</td>
<td>Output</td>
<td>Read Data Output Bit 2</td>
</tr>
<tr>
<td>RDATA1</td>
<td>Output</td>
<td>Read Data Output Bit 1</td>
</tr>
<tr>
<td>RDATA0</td>
<td>Output</td>
<td>Read Data Output Bit 0</td>
</tr>
<tr>
<td>EXE</td>
<td>Input</td>
<td>Execute. When asserted, starts the IR Transceiver Hard IP to transmit or receive IR data</td>
</tr>
<tr>
<td>LEARN</td>
<td>Input</td>
<td>Learning Mode control. When asserted, the IR Transceiver is in learning mode. The IR Transceiver receives data instead of transmit data.</td>
</tr>
<tr>
<td>BUSY</td>
<td>Output</td>
<td>Busy status output</td>
</tr>
<tr>
<td>DRDY</td>
<td>Output</td>
<td>Data Buffer Ready status output</td>
</tr>
<tr>
<td>ERR</td>
<td>Output</td>
<td>Data Error status</td>
</tr>
<tr>
<td>RST</td>
<td>Input</td>
<td>Device level reset signal to reset all internal registers and IROUT signal to OFF state during the device configuration. This port is not accessible to user signals.</td>
</tr>
<tr>
<td>IRIN</td>
<td>Input</td>
<td>Modulated ON/OFF pulse from IR sensor.</td>
</tr>
<tr>
<td>IROUT</td>
<td>Output</td>
<td>Modulated ON/OFF pulse for IR Transmit.</td>
</tr>
</tbody>
</table>
Default Signal Values

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

Verilog Instantiation

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

7.3.9. SB_IO_OD

The SB_IO_OD is the open drain I/O primitive. When the Tristate output is enabled, the I/O pulls down the package pin signal to zero. The following figure and Verilog template illustrate the complete user accessible logic diagram, and its Verilog instantiation.
Table 7.35. SB_IO_OD Port Signals

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

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

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

Parameters

Parameter PIN_TYPE = 6'b000000;

// See Input and Output Pin Function Tables in SB_IO.
// Default value of PIN_TYPE = 6'b000000
Parameter NEG_TRIGGER = 1'b0;

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

Input and Output Pin Function Tables

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

Verilog Instantiation

```verilog
SB_IO_OD OpenDrainInst0 (  
  .PACKAGEPIN (PackagePin),  // User’s Pin signal name
  .LATCHINPUTVALUE (latchinputvalue),  // Latches/holds the Input value
  .CLOCKENABLE (clockenable),  // Clock Enable common to input and
  // output clock
  .INPUTCLK (inputclk),  // Clock for the input registers
  .OUTPUTCLK (outputclk),  // Clock for the output registers
  .OUTPUTENABLE (outputenable),  // Output Pin Tristate/Enable
  // control
  .DOUT0 (dout0),  // Data 0 – out to Pin/Rising clk
  // edge
  .DOUT1 (dout1),  // Data 1 – out to Pin/Falling clk
  // edge
  .DIN0 (din0),  // Data 0 – Pin input/Rising clk
  // edge
  .DIN1 (din1)  // Data 1 – Pin input/Falling clk
  // edge
);  
defparam OpenDrainInst0.PIN_TYPE = 6'b000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
```
7.3.10. **SB_I2C_FIFO**

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

![SB_I2C_FIFO Diagram](Image)

**Figure 7.25. SB_I2C_FIFO**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKI</td>
<td>Input</td>
<td>System Clock input</td>
</tr>
<tr>
<td>CSI</td>
<td>Input</td>
<td>Select Signal. Active High to activate the IP. This usually connects to the output of the decoding logic from MSB of the address bus</td>
</tr>
<tr>
<td>STBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>WEI</td>
<td>Input</td>
<td>Write Enable Signal</td>
</tr>
<tr>
<td>ADRI3</td>
<td>Input</td>
<td>Control Register Address Bit 3</td>
</tr>
<tr>
<td>ADRI2</td>
<td>Input</td>
<td>Control Register Address Bit 2</td>
</tr>
<tr>
<td>ADRI1</td>
<td>Input</td>
<td>Control Register Address Bit 1</td>
</tr>
<tr>
<td>ADRI0</td>
<td>Input</td>
<td>Control Register Address Bit 0</td>
</tr>
<tr>
<td>DAT9</td>
<td>Input</td>
<td>Data Input Bit 9</td>
</tr>
<tr>
<td>DAT8</td>
<td>Input</td>
<td>Data Input Bit 8</td>
</tr>
<tr>
<td>DAT7</td>
<td>Input</td>
<td>Data Input Bit 7</td>
</tr>
<tr>
<td>DAT6</td>
<td>Input</td>
<td>Data Input Bit 6</td>
</tr>
<tr>
<td>DAT5</td>
<td>Input</td>
<td>Data Input Bit 5</td>
</tr>
<tr>
<td>DAT4</td>
<td>Input</td>
<td>Data Input Bit 4</td>
</tr>
<tr>
<td>DAT3</td>
<td>Input</td>
<td>Data Input Bit 3</td>
</tr>
<tr>
<td>DAT2</td>
<td>Input</td>
<td>Data Input Bit 2</td>
</tr>
<tr>
<td>DAT1</td>
<td>Input</td>
<td>Data Input Bit 1</td>
</tr>
<tr>
<td>DAT0</td>
<td>Input</td>
<td>Data Input Bit 0</td>
</tr>
<tr>
<td>DAT9</td>
<td>Output</td>
<td>Data Output Bit 9</td>
</tr>
<tr>
<td>ACKO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRWO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CIRQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CWKUP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCLI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCLO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCLOE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDAI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDAO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDAOE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 7.36. SB_I2C_FIFO Port Signals**
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATO8</td>
<td>Output</td>
<td>Data Output Bit 8</td>
</tr>
<tr>
<td>DATO7</td>
<td>Output</td>
<td>Data Output Bit 7</td>
</tr>
<tr>
<td>DATO6</td>
<td>Output</td>
<td>Data Output Bit 6</td>
</tr>
<tr>
<td>DATO5</td>
<td>Output</td>
<td>Data Output Bit 5</td>
</tr>
<tr>
<td>DATO4</td>
<td>Output</td>
<td>Data Output Bit 4</td>
</tr>
<tr>
<td>DATO3</td>
<td>Output</td>
<td>Data Output Bit 3</td>
</tr>
<tr>
<td>DATO2</td>
<td>Output</td>
<td>Data Output Bit 2</td>
</tr>
<tr>
<td>DATO1</td>
<td>Output</td>
<td>Data Output Bit 1</td>
</tr>
<tr>
<td>DATO0</td>
<td>Output</td>
<td>Data Output Bit 0</td>
</tr>
<tr>
<td>ACKO</td>
<td>Output</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>I2CIRQ</td>
<td>Output</td>
<td>I²C Interrupt output</td>
</tr>
<tr>
<td>I2CWKUP</td>
<td>Output</td>
<td>I²C Wake up from Standby signal</td>
</tr>
<tr>
<td>SRWO</td>
<td>Output</td>
<td>Slave RW “1” master receiving / Slave transmitting “0” master transmitting / Slave receiving</td>
</tr>
<tr>
<td>SCLI</td>
<td>Input</td>
<td>Serial Clock Input</td>
</tr>
<tr>
<td>SCLO</td>
<td>Output</td>
<td>Serial Clock Output</td>
</tr>
<tr>
<td>SCLOE</td>
<td>Output</td>
<td>Serial Clock Output Enable. Active High</td>
</tr>
<tr>
<td>SDAI</td>
<td>Input</td>
<td>Serial Data Input</td>
</tr>
<tr>
<td>SDAO</td>
<td>Output</td>
<td>Serial Data Output</td>
</tr>
<tr>
<td>SDAOE</td>
<td>Output</td>
<td>Serial Data Output Enable. Active High</td>
</tr>
<tr>
<td>FIFOST*</td>
<td>Input</td>
<td>RESET for the FIFO logic</td>
</tr>
<tr>
<td>TXFIFOAEMPTY*</td>
<td>Output</td>
<td>TX FIFO Status Signal</td>
</tr>
<tr>
<td>TXFIFOEMPTY*</td>
<td>Output</td>
<td>TX FIFO Status Signal</td>
</tr>
<tr>
<td>TXFIFOFULL*</td>
<td>Output</td>
<td>TX FIFO Status Signal</td>
</tr>
<tr>
<td>RXFIFOAFULL*</td>
<td>Output</td>
<td>RX FIFO Status Signal</td>
</tr>
<tr>
<td>RXFIFOEMPTY*</td>
<td>Output</td>
<td>RX FIFO Status Signal</td>
</tr>
<tr>
<td>MRDCMPL*</td>
<td>Output</td>
<td>Master Read Complete (only valid for Master Read Mode)</td>
</tr>
</tbody>
</table>

*Note: Only available if I2C_FIFO_ENB = ENABLED.

Parameters

Table 7.37. SB_I2C_FIFO Parameters

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

Synthesis Attribute

**I2C_CLK_DIVIDER**

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

/* synthesis I2C_CLK_DIVIDER= Divide Value */

Divide Value: 0, 1, 2, 3 ... 1023. Default is 0.
SDA_INPUT_DELAYED
SDA_INPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAI signal.
/* synthesis SDA_INPUT_DELAYED= value */
Value:
0: No delay.
1: Add 50 ns delay. (Default value).

SDA_OUTPUT_DELAYED
SDA_OUTPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAO signal.
/* synthesis SDA_OUTPUT_DELAYED= value */
Value:
0: No delay (Default value).
1: Add 50 ns delay.

I2C_FIFO_ENB
"I2C_FIFO_ENB" attribute is used to enable or disable FIFO
/* synthesis I2C_FIFO_ENB= [value] */
Value:
ENABLED: FIFO mode is enabled.
DISABLED: FIFO mode is disabled.

7.4. iCE40 UltraPlus Hard Macros
This section describes the following dedicated hard macro primitives available in iCE40 UltraPlus device.
- SB_HFOSC
- SB_LFOSC
- SB_RGBA_DRV
- SB_LEDDA_IP
- SB_IO_OD
- SB_I2C
- SB_SPI
- SB_MAC16
- SB_SPRAM256KA
- SB_IO_I3C

7.4.1. SB_HFOSC
SB_HFOSC primitive generates 48 MHz nominal clock frequency within +/- 10% variation, with user-programmable divider value of 1, 2, 4, and 8. The HFOSC can drive either the global clock network or fabric routes directly based on the clock network selection.
Table 7.38. SB_HFOSC Port Signals

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

Parameters

Table 7.39. SB_PLL40_CORE Parameters

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

Default Signal Values

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

Input CLKHFEN: Logic “0”
Input CLKHFPU: Logic “0”

Clock Network Selection

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

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */

Value:

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

Verilog Instantiation

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

defparam OSCInst0.CLKHF_DIV = "0b00";
7.4.2. SB_LFOSC

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

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

![SB_LFOSC Diagram](image)

**Figure 7.27. SB_LFOSC**

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

**Table 7.40. SB_LFOSC Port Signals**

**Default Signal Values**

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

- Input CLKLFEN: Logic "0"
- Input CLKLFPU: Logic "0"

**Clock Network Selection**

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

**Synthesis Attributes**

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

**Value:**

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

**Verilog Instantiation**

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

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

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

![SB_RGBA_DRV Diagram](image)

**Figure 7.28. SB_RGBA_DRV**

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

**Table 7.41. SB_RGBA_DRV Port Signals**

**Default Signal Values**

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

- Input CURREN : Logic “0”
- Input RGBLEDEN : Logic “0”
- Input RGB0PWM : Logic “0”
- Input RGB1PWM : Logic “0”
- Input RGB2PWM : Logic “0”
Parameters
The SB_RGBA_DRV primitive contains the following parameter and their default values:

Parameter CURRENT_MODE = "0b0 ";
Parameter values:
"0b0" = Full Current Mode (Default).
"0b1" = Half Current Mode.

Parameter RGB0_CURRENT = "0b000000";
Parameter RGB1_CURRENT = "0b000000";
Parameter RGB2_CURRENT = "0b000000";
Parameter values:
"0b000000" = 0 mA.  // Set this value to use the associated SB_IO_OD instance at RGB // LED location.
"0b000001" = 4 mA for Full Mode; 2 mA for Half Mode
"0b000011" = 8 mA for Full Mode; 4 mA for Half Mode
"0b000111" = 12 mA for Full Mode; 6 mA for Half Mode.
"0b011111" = 20 mA for Full Mode; 10 mA for Half Mode.
"0b111111" = 24 mA for Full Mode; 12 mA for Half Mode.

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

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

![Figure 7.29. SB_LEDDA_IP](image)

Table 7.42. SB_LEDDA_IP Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEDDCS</td>
<td>Input</td>
<td>CS to write LEDD IP registers</td>
</tr>
<tr>
<td>LEDDCLK</td>
<td>Input</td>
<td>Clock to write LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT7</td>
<td>Input</td>
<td>Bit 7 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT6</td>
<td>Input</td>
<td>Bit 6 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT5</td>
<td>Input</td>
<td>Bit 5 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT4</td>
<td>Input</td>
<td>Bit 4 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT3</td>
<td>Input</td>
<td>Bit 3 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT2</td>
<td>Input</td>
<td>Bit 2 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT1</td>
<td>Input</td>
<td>Bit 1 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDAT0</td>
<td>Input</td>
<td>Bit 0 data to write into the LEDD IP registers</td>
</tr>
<tr>
<td>LEDDADDR3</td>
<td>Input</td>
<td>LEDD IP register address bit 3</td>
</tr>
<tr>
<td>LEDDADDR2</td>
<td>Input</td>
<td>LEDD IP register address bit 2</td>
</tr>
<tr>
<td>LEDDADDR1</td>
<td>Input</td>
<td>LEDD IP register address bit 1</td>
</tr>
<tr>
<td>LEDDADDR0</td>
<td>Input</td>
<td>LEDD IP register address bit 0</td>
</tr>
<tr>
<td>LEDDEN</td>
<td>Input</td>
<td>Data enable input to indicate data and address are stable</td>
</tr>
<tr>
<td>LEDDEXE</td>
<td>Input</td>
<td>Enable the IP to run the blinking sequence. When it is LOW, the sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stops at the nearest OFF state.</td>
</tr>
<tr>
<td>LEDDRST</td>
<td>Input</td>
<td>Device level reset signal to reset all internal registers during the device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration. This port is not accessible to user signals.</td>
</tr>
<tr>
<td>PWMOUT0</td>
<td>Output</td>
<td>PWM output 0</td>
</tr>
<tr>
<td>PWMOUT1</td>
<td>Output</td>
<td>PWM output 1</td>
</tr>
<tr>
<td>PWMOUT2</td>
<td>Output</td>
<td>PWM output 2</td>
</tr>
<tr>
<td>EDDON</td>
<td>Output</td>
<td>Indicating the LED is on</td>
</tr>
</tbody>
</table>
Default Signal Values
The iCEcube2 software assigns the logic "0" value to all unconnected input ports.

Verilog Instantiation
```
SB_LEDDA_IP PWMgen_inst (  
  .LEDDCS(ledd_cs),  
  .LEDDCLK(led_clk),  
  .LEDDDAT7(led_ip_data[7]),  
  .LEDDDAT6(led_ip_data[6]),  
  .LEDDDAT5(led_ip_data[5]),  
  .LEDDDAT4(led_ip_data[4]),  
  .LEDDDAT3(led_ip_data[3]),  
  .LEDDDAT2(led_ip_data[2]),  
  .LEDDDAT1(led_ip_data[1]),  
  .LEDDDAT0(led_ip_data[0]),  
  .LEDDADDR3(led_ip_addr[3]),  
  .LEDDADDR2(led_ip_addr[2]),  
  .LEDDADDR1(led_ip_addr[1]),  
  .LEDDADDR0(led_ip_addr[0]),  
  .LEDDDEN(led_ip_den),  
  .LEDDEXE(led_ip_exe),  
  .LEDDRST(led_ip_rst),  
  .PWMOUT0(LED0),  
  .PWMOUT1(LED1),  
  .PWMOUT2(LED2),  
  .LEDDON(led_on)  
);  
```
Table 7.43. SB_IO_OD Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGEPIN</td>
<td>Bidirectional</td>
<td>Bidirectional I/O pin.</td>
</tr>
<tr>
<td>LATCHINPUTVALUE</td>
<td>Input</td>
<td>Latches/Holds the input data.</td>
</tr>
<tr>
<td>CLOCKENABLE</td>
<td>Input</td>
<td>Clock enable signal.</td>
</tr>
<tr>
<td>INPUTCLK</td>
<td>Input</td>
<td>Clock for the input registers.</td>
</tr>
<tr>
<td>OUTPUTCLK</td>
<td>Input</td>
<td>Clock for the output registers.</td>
</tr>
<tr>
<td>OUTPUTENABLE</td>
<td>Input</td>
<td>Enable Tristate output. Active high.</td>
</tr>
<tr>
<td>DOUT0</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DOUT1</td>
<td>Input</td>
<td>Data to package pin.</td>
</tr>
<tr>
<td>DIN0</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
<tr>
<td>DIN1</td>
<td>Output</td>
<td>Data from package pin.</td>
</tr>
</tbody>
</table>
Default Signal Values
The iCEcube2 software assigns the logic “0” value to all unconnected input ports except for CLOCKENABLE.

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

Parameter Values
Parameter PIN_TYPE = 6'b000000;
// See Input and Output Pin Function Tables in SB_IO.
// Default value of PIN_TYPE = 6'b000000
Parameter NEG_TRIGGER = 1'b0;
// Specify the polarity of all FFs in the I/O to be falling edge when
NEG_TRIGGER = 1. Default is 1'b0, rising edge.

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

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

defparam OpenDrainInst0.PIN_TYPE = 6'b000000;
defparam OpenDrainInst0.NEG_TRIGGER = 1'b0;
7.4.5.1. SB_I2C

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

![SB_I2C Diagram]

Table 7.44. SB_I2C Port Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCLKI</td>
<td>Input</td>
<td>System Clock input.</td>
</tr>
<tr>
<td>SBRWI</td>
<td>Input</td>
<td>System Read/Write Input.</td>
</tr>
<tr>
<td>SBSTBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>SBADRI0</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 0.</td>
</tr>
<tr>
<td>SBADRI1</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 1.</td>
</tr>
<tr>
<td>SBADRI2</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 2.</td>
</tr>
<tr>
<td>SBADRI3</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 3.</td>
</tr>
<tr>
<td>SBADRI5</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 5.</td>
</tr>
<tr>
<td>SBADRI6</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 6.</td>
</tr>
<tr>
<td>SBADRI7</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 7.</td>
</tr>
<tr>
<td>SBDATI0</td>
<td>Input</td>
<td>System Data Input. Bit 0.</td>
</tr>
<tr>
<td>SBDATI1</td>
<td>Input</td>
<td>System Data Input. Bit 1.</td>
</tr>
<tr>
<td>SBDATI2</td>
<td>Input</td>
<td>System Data Input. Bit 2.</td>
</tr>
<tr>
<td>SBDATI3</td>
<td>Input</td>
<td>System Data Input. Bit 3.</td>
</tr>
<tr>
<td>SBDATI5</td>
<td>Input</td>
<td>System Data Input. Bit 5.</td>
</tr>
<tr>
<td>SBDATI7</td>
<td>Input</td>
<td>System Data Input. Bit 7.</td>
</tr>
<tr>
<td>SBDATO0</td>
<td>Output</td>
<td>System Data Output. Bit 0.</td>
</tr>
<tr>
<td>SBDATO1</td>
<td>Output</td>
<td>System Data Output. Bit 1.</td>
</tr>
<tr>
<td>SBDATO2</td>
<td>Output</td>
<td>System Data Output. Bit 2.</td>
</tr>
</tbody>
</table>
**SDA_INPUT_DELAYED**

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

```c
/* synthesis SDA_INPUT_DELAYED= value */
Value:
0: No delay.
1: Add 50 ns delay. (Default value).
```
SDA_OUTPUT_DELAYED
SDA_OUTPUT_DELAYED attribute is used to add 50 ns additional delay to the SDAO signal.
/* synthesis SDA_OUTPUT_DELAYED= value */
Value:
0: No delay (Default value).
1: Add 50 ns delay.

SCL_INPUT_FILTERED
SCL_INPUT_FILTERED attribute is used to add 50 ns glitch filter to the SCLI signal.
/* synthesis SCL_INPUT_FILTERED= value */
Value:
0: No Filter delay (Default value).
1: Add 50 ns glitch filter delay.

7.4.6. SB_SPI
The iCE40 UltraPlus device supports two SPI hard IP primitives, located at lower left corner and lower right corner of the chip.
<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCLKI</td>
<td>Input</td>
<td>System Clock input.</td>
</tr>
<tr>
<td>SBRWI</td>
<td>Input</td>
<td>System Read/Write Input.</td>
</tr>
<tr>
<td>SBSTBI</td>
<td>Input</td>
<td>Strobe Signal</td>
</tr>
<tr>
<td>SBADRI0</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 0.</td>
</tr>
<tr>
<td>SBADRI1</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 1.</td>
</tr>
<tr>
<td>SBADRI2</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 2.</td>
</tr>
<tr>
<td>SBADRI3</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 3.</td>
</tr>
<tr>
<td>SBADRI5</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 5.</td>
</tr>
<tr>
<td>SBADRI6</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 6.</td>
</tr>
<tr>
<td>SBADRI7</td>
<td>Input</td>
<td>System Bus Control registers address. Bit 7.</td>
</tr>
<tr>
<td>SBDATI0</td>
<td>Input</td>
<td>System Data Input. Bit 0.</td>
</tr>
<tr>
<td>SBDATI1</td>
<td>Input</td>
<td>System Data Input. Bit 1.</td>
</tr>
<tr>
<td>SBDATI2</td>
<td>Input</td>
<td>System Data Input. Bit 2.</td>
</tr>
<tr>
<td>SBDATI3</td>
<td>Input</td>
<td>System Data Input. Bit 3.</td>
</tr>
<tr>
<td>SBDATI5</td>
<td>Input</td>
<td>System Data Input. Bit 5.</td>
</tr>
<tr>
<td>SBDATI7</td>
<td>Input</td>
<td>System Data Input. Bit 7.</td>
</tr>
<tr>
<td>SBDATO0</td>
<td>Input</td>
<td>System Data Output. Bit 0.</td>
</tr>
<tr>
<td>SBDATO1</td>
<td>Input</td>
<td>System Data Output. Bit 1.</td>
</tr>
<tr>
<td>SBDATO2</td>
<td>Input</td>
<td>System Data Output. Bit 2.</td>
</tr>
<tr>
<td>SBDATO3</td>
<td>Input</td>
<td>System Data Output. Bit 3.</td>
</tr>
<tr>
<td>SBDATO5</td>
<td>Input</td>
<td>System Data Output. Bit 5.</td>
</tr>
<tr>
<td>SBDATO7</td>
<td>Input</td>
<td>System Data Output. Bit 7.</td>
</tr>
<tr>
<td>SBACKO</td>
<td>Output</td>
<td>System Acknowledgement</td>
</tr>
<tr>
<td>SPIIRQ</td>
<td>Output</td>
<td>SPI Interrupt output.</td>
</tr>
<tr>
<td>SPIWKUP</td>
<td>Output</td>
<td>SPI Wake Up from Standby signal.</td>
</tr>
<tr>
<td>MI</td>
<td>Input</td>
<td>Master Input from PAD</td>
</tr>
<tr>
<td>SO</td>
<td>Output</td>
<td>Slave Output to PAD</td>
</tr>
<tr>
<td>SOE</td>
<td>Output</td>
<td>Slave Output Enable to PAD. Active High.</td>
</tr>
<tr>
<td>SI</td>
<td>Input</td>
<td>Slave Input from PAD</td>
</tr>
<tr>
<td>MO</td>
<td>Output</td>
<td>Master Output to PAD</td>
</tr>
<tr>
<td>MOE</td>
<td>Output</td>
<td>Master Output Enable to PAD. Active High.</td>
</tr>
<tr>
<td>SCKO</td>
<td>Output</td>
<td>Slave Clock Output to PAD</td>
</tr>
<tr>
<td>SCKOE</td>
<td>Output</td>
<td>Slave Clock Output Enable to PAD. Active High.</td>
</tr>
<tr>
<td>SCSNI</td>
<td>Input</td>
<td>Slave Chip Select Input From PAD</td>
</tr>
<tr>
<td>MCSNO0</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 0.</td>
</tr>
<tr>
<td>MCSNO1</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 1.</td>
</tr>
<tr>
<td>MCSNO2</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 2.</td>
</tr>
<tr>
<td>MCSNO3</td>
<td>Output</td>
<td>Master Chip Select Output to PAD. Line 3.</td>
</tr>
<tr>
<td>MCSNOE0</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 0.</td>
</tr>
<tr>
<td>MCSNOE1</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 1.</td>
</tr>
<tr>
<td>MCSNOE2</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 2.</td>
</tr>
<tr>
<td>MCSNOE3</td>
<td>Output</td>
<td>Master Chip Select Output Enable to PAD. Active High. Line 3.</td>
</tr>
</tbody>
</table>
Parameters

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

Table 7.47. SB_SPI Parameters

<table>
<thead>
<tr>
<th>SPI Location</th>
<th>Parameter</th>
<th>Parameter Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Left Corner</td>
<td>BUS_ADDR74</td>
<td>0b0000</td>
<td>Fixed value. SBADRI[7:4] bits also should match with this value to activate the IP.</td>
</tr>
<tr>
<td>Lower Right Corner</td>
<td>BUS_ADDR74</td>
<td>0b0010</td>
<td>Fixed value. SBADRI[7:4] bits also should match with this value to activate the IP.</td>
</tr>
</tbody>
</table>

Synthesis Attribute

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

/* synthesis SPI_CLK_DIVIDER=[Divide Range] */

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

7.4.7. SB_MAC16

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

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

![SB_MACH16 Diagram](image)

Figure 7.33. SB_MAC16

Port B (Generated Clock)
7.4.8. SB_SPRAM256KA

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

![Single Port RAM Primitive](image)

**Figure 7.34. SB_SPRAM256KA**

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

The following shows which MASKWREN bits enables the mask for the DATAIN nibbles.
- MASKWREN(3) enables mask for DATAIN(15:12)
- MASKWREN(2) enables mask for DATAIN (11:8)
- MASKWREN(1) enables mask for DATAIN (7:4)
- MASKWREN(0) enables mask for DATAIN(3:0)

### 7.4.9. **SB_IO_I3C**

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

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

![Figure 7.35. SB_IO_I3C](image-url)
Default Signal Values
The iCEcube2 software assigns the logic ‘0’ value to all unconnected input ports except for CLOCK_ENABLE.

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

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

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

Table 7.49. Input Pin Function

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

Table 7.50. Output Pin Function

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

Pull-Up Resistor Configuration.
You can configure the internal pull-up resistor strength of an I/O to a predefined resistor value through the parameter settings and synthesis attributes. The PULLUP parameter setting enables one of the resistor values as given in table 1 and WEAKPULLUP parameter setting enables the 100k pull-up resistor.
The specific resistor value associated with the PULLUP parameter needs to specify through the PULLUP_RESISTOR attribute. The PULLUP_RESISTOR value is effective only when PULLUP parameter is set to 1.
Both the PULLUP and WEAKPULLUP parameters are allowed to set simultaneously. When both parameters are set the effective pull-up resistor values is [100K | | {3P3K/6P8K/10K}].
Once the required parameters, attribute values are set, the active low PU_ENB, WEAK_PU_ENB signals allows you to control the pull-up resistor network dynamically.

Synthesis Attribute Syntax:
/* synthesis PULLUP_RESISTOR = "3P3K" */

Resistor Value
Table 7.51. Resistor Value

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

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

SB_IO_I3C I3CIO_INST
(
  .PACKAGE_PIN (Package_Pin), //User’s Pin signal name
  .LATCH_INPUT_VALUE (latch_input_value), //Latches/holds the Input value
  .CLOCK_ENABLE (clock_enable), //Clock Enable common to input and output clock
  .INPUT_CLK (input_clk), //Clock for the input registers
  .OUTPUT_CLK (output_clk), //Clock for the output registers
  .OUTPUT_ENABLE (output_enable), //Output Pin Tristate/Enable control
  .D_OUT_0 (d_out_0), //Data 0 - out to Pin/Rising clk edge
  .D_OUT_1 (d_out_1), //Data 1 - out to Pin/Falling clk edge
  .D_IN_0 (d_in_0), //Data 0 - Pin input/Rising clk edge
  .D_IN_1 (d_in_1), //Data 1 - Pin input/Falling clk edge
  .PU_ENB (pu_enb), // Active low Pull Up resistor Enable -Default to GND
  .WEAK_PU_ENB (weak_pu_enb) //Active low Weak Pull Up Enable for 100k resistor. Default to GND
)
/* synthesis PULLUP_RESISTOR "3P3K" */

defparam I3CIO_INST.PIN_TYPE = 6'b000000; // See Input and Output Pin Function Tables.
  // Default value of PIN_TYPE = 6’000000 i.e. an input pad, with the input signal registered.

defparam I3CIO_INST.PULLUP = 1'b0; // By default, the IO will have NO pull up.
defparam I3CIO_INST.WEAK_PULLUP = 1'b0; // By default, the IO will have NO Weak pull up.
defparam I3CIO_INST.NEG_TRIGGER = 1'b0;
defparam I3CIO_INST.IO_STANDARD = "SB_LVCMOS";
7.5. Device Configuration Primitives

7.5.1. SB_WARMBOOT

The iCE FPGA devices permit you to load a different configuration image during regular operation. Through the use of the Warm Boot Primitive, you can load one of four pre-defined configuration images into the iCE FPGA device. Note that this Warm Boot mode is different from the Cold Boot operation, which is executed during the initial device boot-up sequence.

![SB_WARMBOOT Diagram](image)

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

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

**Verilog Instantiation**

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

Figure 7.36. SB_WARMBOOT

---

© 2016-2021 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.

All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
Technical Support Assistance
Submit a technical support case through www.latticesemi.com/techsupport.
# Revision History

## Revision 3.2, November 2021

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Adjustments in formatting across the document.</td>
</tr>
<tr>
<td>Acronyms in This Document</td>
<td>Added this section.</td>
</tr>
<tr>
<td>I/O Primitives</td>
<td>Changed table name to Pull-up Resistor Value and K to k unit in Table 4.4.</td>
</tr>
<tr>
<td>Hard Macro Primitives</td>
<td>• Changed 16K x 16 mode to 16 kb x 16 mode in SB_SPRAM256KA.</td>
</tr>
<tr>
<td></td>
<td>• Updated Table 7.6 and Table 7.47 to change I^2C column to SPI Location and parameter value of Lower Right Corner to 0b0010.</td>
</tr>
<tr>
<td></td>
<td>• Changed K to k unit in Table 7.51.</td>
</tr>
</tbody>
</table>

## Revision 3.1, July 2021

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>• Assigned document number FPGA-TN-02026.</td>
</tr>
<tr>
<td></td>
<td>• Updated document template.</td>
</tr>
<tr>
<td>Disclaimers</td>
<td>Added this section.</td>
</tr>
<tr>
<td>Hard Macro Primitives</td>
<td>Updated Figure 7.15.</td>
</tr>
</tbody>
</table>

## Revision 3.0, August 2016

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Previous Lattice release.</td>
</tr>
</tbody>
</table>