



XAPP480 (v1.0) May 2, 2007

Using Suspend Mode in Spartan-3 Generation FPGAs

Summary

While some applications require the lowest possible system cost or highest performance, still other applications require the lowest possible standby power. Spartan™-3 Generation FPGAs offer low-power options that balance cost and performance trade-offs.

The Spartan-3A, Spartan-3AN, and Spartan-3A DSP FPGA families offer an advanced static power management feature called Suspend mode, which reduces FPGA power consumption while retaining the FPGA's configuration data and maintaining the application state. The device can quickly enter and exit Suspend mode as required in an application.

Suspend Features and Benefits

The following are the significant features and benefits of the Suspend mode.

- Quickly and easily puts the FPGA into a static condition, eliminating most active current.
- Quiescent current is reduced by 40% or more.
- FPGA configuration data and the state of the FPGA application during Suspend mode is retained.
- Fast, programmable FPGA wake-up time from Suspend mode, in less than 500 μ s.
- Each user-I/O pin has an individual control that defines how the pin behaves during Suspend mode.
- Suspend mode is externally activated by the system using a single dedicated control pin called SUSPEND.
- The FPGA's AWAKE pin indicates the present Suspend mode status.

Design Preparation for Suspend Mode

To use the Suspend feature, follow these steps:

- [Define the I/O Behavior During Suspend Mode](#) in the source design or in a user constraints file (UCF).
- Define the [AWAKE Pin Behavior when Suspend Feature is Enabled](#).
- Define the [SUSPEND Input Glitch Filter](#) setting.
- Define the [Suspend Mode Wake-Up Timing Controls](#).
- [Enable the Suspend Feature](#).
- Generate the FPGA bitstream.

If a specific application state must be maintained, see ["Design Requirements to Maintain Application Data," page 8](#).

Entering Suspend Mode

[Figure 1](#) provides a block diagram of how the FPGA enters Suspend mode. [Figure 2](#) provides example waveforms.

© 2007 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

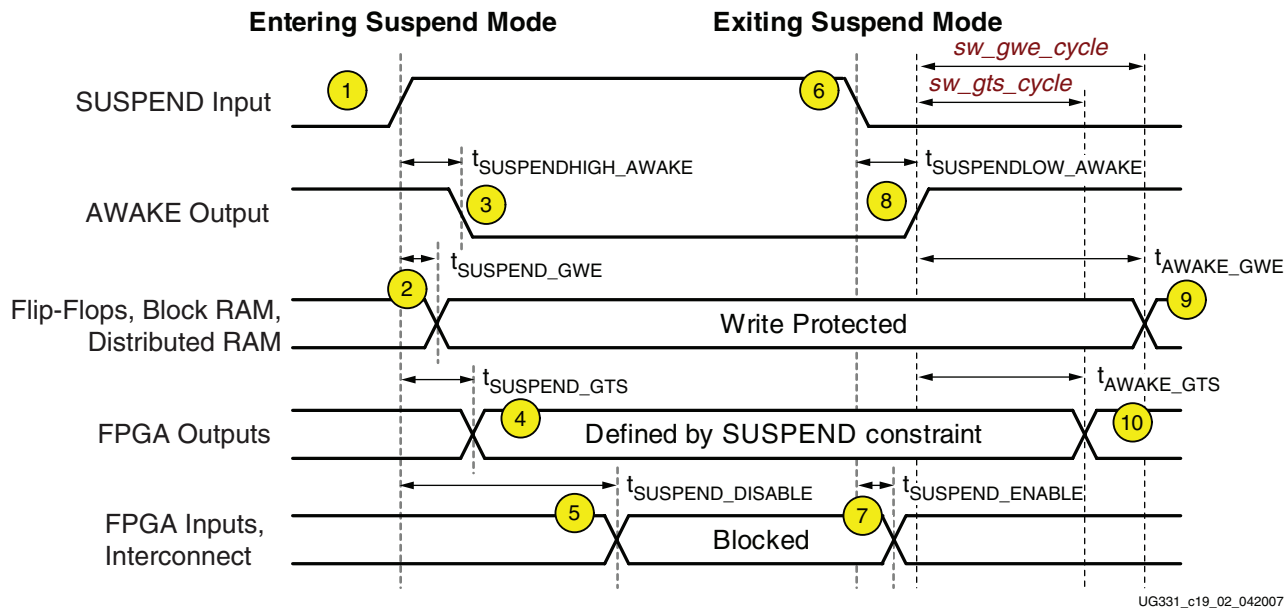


Figure 2: Suspend Mode Waveforms (Entering and Exiting) - See Timing in Table 1

Phases 1 through 5 in Figure 2 are described below. Phases 6 through 10 are described after Figure 3.

1. An external signal drives the FPGA's SUSPEND pin High, unconditionally forcing the FPGA into the power-saving Suspend mode. Data values are captured for I/O pins with a SUSPEND constraint set to DRIVE_LAST_VALUE; however, this value is not presented until Step 4.
2. In response to the SUSPEND input going High, and after a delay of $t_{\text{SUSPEND_GWE}}$, the FPGA write protects and preserves the states of all clocked elements. The states of all flip-flops, block RAM, distributed RAM (LUT RAM), shift registers (SRL16), and I/O latches are preserved during Suspend mode.
3. After a delay of $t_{\text{SUSPENDHIGH_AWAKE}}$, the FPGA drives the AWAKE output Low to indicate that it is entering SUSPEND mode.
4. After a delay of $t_{\text{SUSPEND_GTS}}$, the FPGA switches the normal behavior of all outputs over to the Suspend mode behavior defined by the SUSPEND constraint assigned to each I/O. See "Define the I/O Behavior During Suspend Mode," page 7.
5. After a delay of $t_{\text{SUSPEND_DISABLE}}$, FPGA inputs are blocked and the interconnects shut off to prevent any internal switching activity.

Exiting Suspend Mode

There are two possible ways to exit Suspend mode in a powered system:

- Drive the SUSPEND input Low, exiting Suspend mode normally.
- Pulse the PROG_B input Low, resetting the FPGA and causing the FPGA to reprogram.

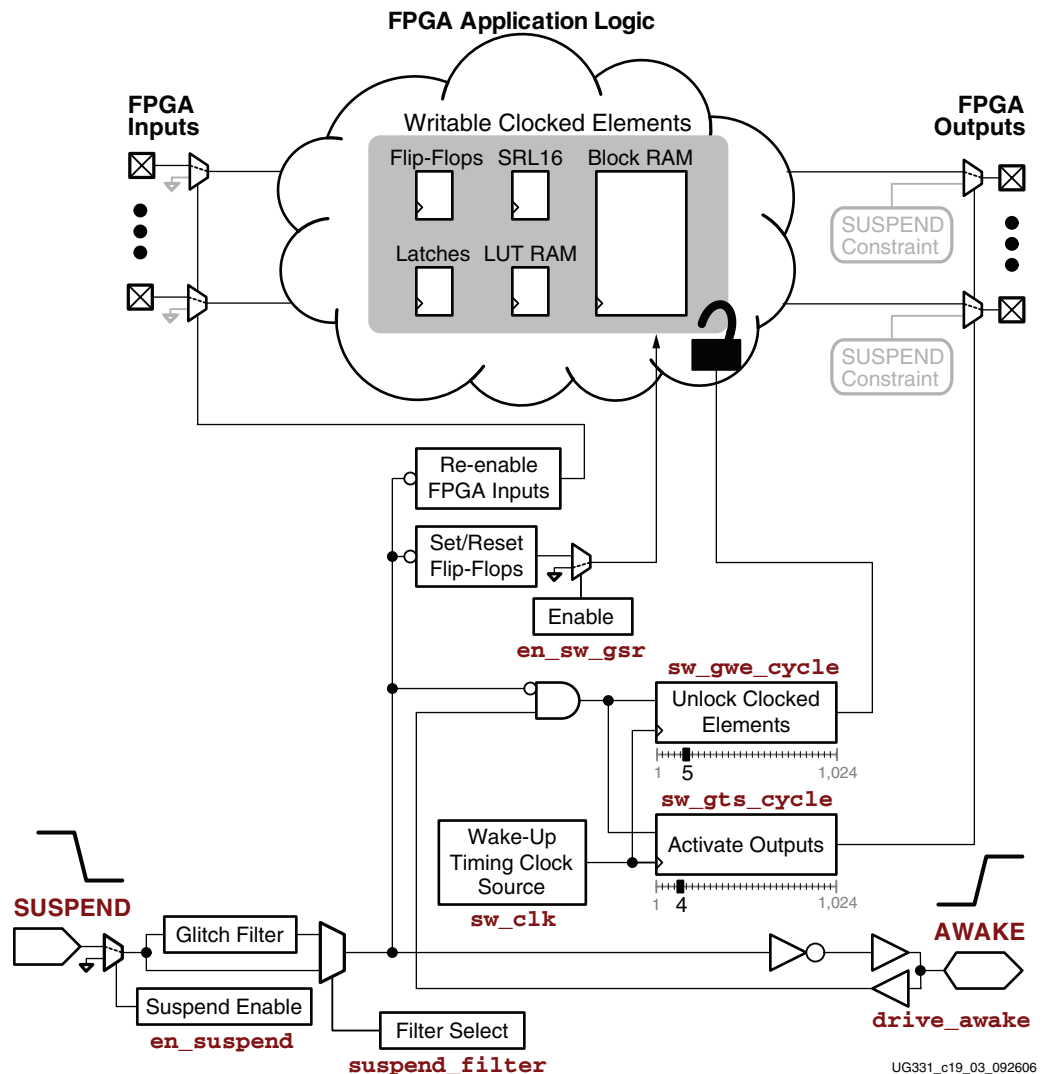


Figure 3: Exiting Suspend Mode

Figure 3 is a block diagram showing how to exit Suspend mode using the SUSPEND pin.

When SUSPEND goes Low, the FPGA automatically re-enables all inputs and interconnects after a delay of $t_{\text{SUSPEND_ENABLE}}$.

If enabled in the FPGA bitstream, all flip-flops are optionally globally set or reset according to the FPGA design description. By default, the flip-flops are not globally set or reset, which preserves the state of the FPGA application from the beginning of Suspend mode.

The remaining wake-up process depends on two user-programmable timers which define when FPGA outputs are re-enabled and when the write-protect lock is released from all writable clocked elements. These timers begin after the AWAKE pin is High. The wake-up timing clock source is also programmable.

Phases 6 through 10 correspond to the markers in Figure 2, page 3:

6. The system drives the FPGA's SUSPEND input Low, causing the FPGA to exit Suspend mode.
7. The FPGA releases the inputs and interconnect after a delay of $t_{\text{SUSPEND_ENABLE}}$, allowing signals to propagate internally. There is no danger of corrupting the internal state because all clocked elements are still write protected.

8. After a delay of $t_{\text{SUSPENDLOW_AWAKE}}$, the FPGA asserts the AWAKE signal with the bitstream option **drive_awake:yes**. If the option is **drive_awake:no**, then the FPGA releases AWAKE to become an open-drain output. In this case, an external pull-up resistor is required or an external signal must drive AWAKE High before the FPGA continues to awaken. All subsequent timing is measured from when the AWAKE output goes High. If multiple FPGAs are waking up and need to be synchronized, set **drive_awake:no** in each and then use an external pull-up resistor to synchronize the AWAKE pins. If other devices are waking up and the FPGA(s) need to wait, set **drive_awake:no** and use an external signal to control the AWAKE pin and drive it High once the rest of the system is ready.
9. After a delay of $t_{\text{AWAKE_GTS}}$, the FPGA switches output behavior from the specified **SUSPEND Constraint** to the function specified in the FPGA application. The timing of this switch-over is controlled by the Suspend/Wake **sw_gts_cycle** bitstream generation setting, which defines when the FPGA's internal Global Three-State (GTS) control is released. After the specified number of clock cycles, the outputs are active according to normal FPGA application. By default, the outputs switch over four clock cycles after AWAKE goes High. The outputs are generally released before the clocked elements to allow signals to propagate out of the FPGA.
10. After a delay of $t_{\text{AWAKE_GWE}}$, the writable, clocked elements are released according to the Suspend/Wake **sw_gwe_cycle** bitstream generator setting, which defines when the FPGA's internal Global Write Enable (GWE) control is asserted. After the specified cycle, it is again possible to write to flip-flops, block RAM, distributed RAM (LUT RAM), shift registers (SRL16), and I/O latches. By default, the clocked elements are released five clock cycles after AWAKE goes High. Generally, the write-protect lock should be held until after outputs are enabled.

PROG_B Programming Pin Always Overrides Suspend Mode

Pulsing the PROG_B programming pin Low always overrides Suspend mode and forces the FPGA to restart configuration. Likewise, power-cycling the FPGA also restarts configuration. If the Suspend input remains High the device will re-enter Suspend mode after finishing configuration.

Suspend Mode Timing Example

[Table 1](#) provides example, typical timing for the Suspend feature. Refer to the *Spartan-3A FPGA Family Data Sheet* ([DS529](#)), the *Spartan-3AN FPGA Family Data Sheet* ([DS557](#)), and the *Spartan-3A DSP FPGA Family Data Sheet* ([DS610](#)) as the official sources of these values.

Table 1: Suspend Mode Timing Parameters

Symbol	Description	Min	Typ	Max	Units
Entering Suspend Mode					
t _{SUSPENDHIGH_AWAKE}	Rising edge of SUSPEND pin to falling edge of AWAKE pin, without glitch filter (<i>suspend_filter:No</i>)	–	7	–	ns
t _{SUSPENDFILTER}	Adjustment to SUSPEND pin rising edge parameters when glitch filter enabled (<i>suspend_filter:Yes</i>)	+160	+300	+600	ns
t _{SUSPEND_GWE}	Rising edge of SUSPEND pin to write-protect lock on all writable clocked elements, without glitch filter (<i>suspend_filter:No</i>)	–	< 5	–	ns
t _{SUSPEND_GTS}	Rising edge of SUSPEND pin until FPGA output pins drive their defined SUSPEND constraint behavior, without glitch filter (<i>suspend_filter:No</i>)	–	10	–	ns
t _{SUSPEND_DISABLE}	Rising edge of the SUSPEND pin to FPGA input pins and interconnect disabled, without glitch filter (<i>suspend_filter:No</i>)	–	340	–	ns
Exiting Suspend Mode					
t _{SUSPENDLOW_AWAKE}	Falling edge of the SUSPEND pin to rising edge of the AWAKE pin. Does not include DCM lock time.	–	4 to 108	–	μs
t _{SUSPEND_ENABLE}	Falling edge of the SUSPEND pin to FPGA input pins and interconnect re-enabled	–	3.7 to 109	–	μs
t _{AWAKE_GWE1}	Rising edge of the AWAKE pin until write-protect lock released on all writable clocked elements, using <i>sw_clk:InternalClk</i> and <i>sw_gwe_cycle:1</i> .	–	67	–	ns
t _{AWAKE_GWE512}	Rising edge of the AWAKE pin until write-protect lock released on all writable clocked elements, using <i>sw_clk:InternalClk</i> and <i>sw_gwe_cycle:512</i> .	–	14	–	μs
t _{AWAKE_GTS1}	Rising edge of the AWAKE pin until outputs return to the behavior described in the FPGA application, using <i>sw_clk:InternalClk</i> and <i>sw_gts_cycle:1</i> .	–	57	–	ns
t _{AWAKE_GTS512}	Rising edge of the AWAKE pin until outputs return to the behavior described in the FPGA application, using <i>sw_clk:InternalClk</i> and <i>sw_gts_cycle:512</i> .	–	14	–	μs

Enable the Suspend Feature

The Suspend power-saving feature must first be enabled in the FPGA bitstream before it can be used. By default, the Suspend feature is disabled and SUSPEND has no effect.

Via User Constraints File (UCF)

Suspend mode is enabled and the **SUSPEND Input Glitch Filter** option is defined using a CONFIG statement in a user constraints file (UCF). Table 2 shows the available options. This is the recommended method for enabling Suspend mode as this constraint also automatically reserves the AWAKE pin.

```
CONFIG ENABLE_SUSPEND = "FILTERED" ;
```

Figure 4: UCF Constraint Defining Suspend Mode Behavior for an I/O pin

Table 2: Available Options for the ENABLE_SUSPEND Constraint

Option	Suspend Mode	SUSPEND Pin Filter	AWAKE Pin
NO	Suspend mode is disabled	Not applicable. Connect SUSPEND pin to GND.	Available as a user I/O pin in the FPGA application
FILTERED	Suspend mode is enabled	Glitch filter is enabled	AWAKE status indicator
UNFILTERED		Glitch filter is bypassed	

Via BitGen

CAUTION! Setting the [en_suspend](#) bitstream option is an alternate way to enable the Suspend mode. However, this method is not recommended because it does not automatically reserve the AWAKE pin in the application.

```
bitgen -g en_suspend:Yes
```

Define the I/O Behavior During Suspend Mode

Use a [SUSPEND Constraint](#) to define the behavior of each I/O and output pin during Suspend mode.

Single-Ended I/O Standards

Each output, open-drain output, or bidirectional I/O pin in the FPGA application that uses a single-ended I/O standard can be individually programmed for one of the Suspend mode behaviors shown in [Table 3](#). The default behavior is for the pin to be high impedance during Suspend mode although other options are available.

Table 3: Output Behavior Options during Suspend Mode

SUSPEND Attribute	Function
DRIVE_LAST_VALUE	The output continues to drive the level that was last stored in the output latch, according to the chosen standard. Requires V_{CCO} to remain at the recommended operating conditions for the bank.
3STATE (default)	The output is in the high-impedance state with no active internal pull-up or pull-down resistor. Results in the lowest possible I/O current draw.
3STATE_PULLUP	The output is in the high-impedance state with an internal pull-up resistor to the associated V_{CCO} supply. Requires V_{CCO} to remain at the recommended operating conditions for the bank.
3STATE_PULLDOWN	The output is in the high-impedance state with an internal pull-down resistor to GND.
3STATE_KEEPER	The output is high impedance. The internal bus keeper circuit is active. Requires V_{CCO} to remain at the recommended operating conditions for the bank.

Differential I/O Standards

The differential output drivers and input receivers consume static power when used in an FPGA application. In Suspend mode, differential inputs and outputs are disabled to save power.

The output drivers for the “true” differential I/O standards (LVDS, RSDS, mini-LVDS, PPDS, TMDS) are high impedance, using any of the [3STATE](#) attributes described in [Table 3](#). The [DRIVE_LAST_VALUE](#) attribute is not supported for differential output drivers.

Treat the pseudo-differential I/O standards, such as BLVDS, LVPECL, DIFF_HSTL, and DIFF_SSTL, as two single-ended I/O pins. All the attributes apply as for “[Single-Ended I/O](#)”

Standards although for any differential standard the settings must be set appropriately for both pins of the complementary pair.

When in the high-impedance state, the differential driver pair does not conduct current to the power or ground rails, or between adjacent pins.

Differential input receivers are disabled in Suspend mode. Differential input termination (DIFF_TERM) is disabled when in Suspend mode.

SUSPEND Constraint

The SUSPEND constraint allows each pin to have an individually defined behavior during Suspend mode. The available options are in [Table 3, page 7](#).

UCF Example

[Figure 5](#) shows an example UCF constraint that defines the Suspend mode behavior for a specific pin. The SUSPEND constraint can be included on the same UCF line as other constraints for a pin.

```
NET "<net_name>" SUSPEND = "<io_type>" ;
```

Figure 5: UCF Constraint Defining Suspend Mode Behavior for an I/O pin

Below is an example of UCF entries for a single-ended pin and a differential pair:

```
NET "TX<0>" IOSTANDARD = LVCMOS_33 ; | SUSPEND = DRIVE_LAST_VALUE ;
NET "TX_P<0>" IOSTANDARD = LVDS_33 ; | SUSPEND = 3STATE_PULLUP ;
NET "TX_N<0>" IOSTANDARD = LVDS_33 ; | SUSPEND = 3STATE_PULLDOWN ;
```

More Information

For additional information on the SUSPEND constraint, see the Constraints Guide.

- **Constraints Guide** for ISE™ 9.1i
<http://toolbox.xilinx.com/docsan/xilinx9/books/docs/cgd/cgd.pdf>

Application State Maintained during Suspend Mode

After entering Suspend mode, all writable clocked elements are write-protected after a delay of $t_{\text{SUSPEND_GWE}}$. The state of all clocked memory elements is maintained during Suspend mode.

- Logic block flip-flops
- I/O block latches and flip-flops
- Logic block distributed RAM (LUT RAM)
- Logic block shift registers (SRL16)
- Block RAM and registers

When exiting Suspend mode, all writable clocked elements are re-enabled, controlled by the [sw_gwe_cycle](#) setting.

An additional bitstream option called [en_sw_gsr](#) controls whether all clocked elements are globally set or reset when the FPGA awakens from Suspend mode. By default, [en_sw_gsr:No](#), which means that clocked elements are not set or reset when the FPGA awakens and all states are preserved.

Design Requirements to Maintain Application Data

During Suspend mode, the Global Write Enable (GWE) is removed, maintaining the state of all flip-flops and user RAM. The FPGA requires a delay of $t_{\text{SUSPEND_GWE}}$ between recognizing a High on the SUSPEND pin and disabling GWE internally. This is the first event after SUSPEND goes High, before AWAKE toggles and before the inputs are disabled. During this delay, additional user clocks to flip-flops or RAM may continue to update their contents. Since the

GWE signal may have some skew between locations on the device, some may be disabled while others remain enabled on the last clock edge before GWE takes full effect. Therefore, if application data must be preserved, it is recommended to stop the clock or remove clock enables on storage elements before entering Suspend mode. For example, a BUFGE buffer can be used to disable clock inputs before Suspend mode is activated. An external oscillator can usually be disabled easily, which can preserve the state of the entire system. Minimizing the number of clocks in the system will make it easier to control clocks for Suspend mode.

Make sure `en_sw_gsr:No` to avoid initializing the flip-flops when exiting Suspend mode. Exiting Suspend mode should be synchronized to a user clock in order to avoid race conditions corrupting the application data. Inputs are enabled first, allowing control signals to continue to hold off the toggling of storage elements. The assertion of GWE can be synchronized to a user clock to align it with a system clock edge.

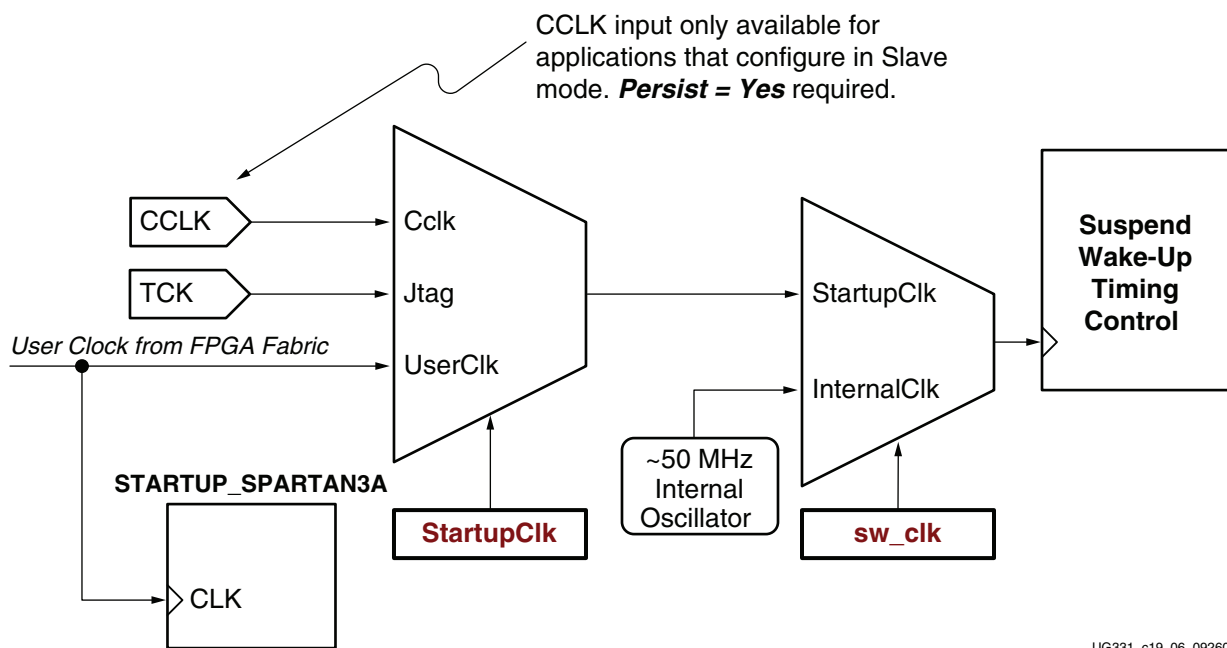
Suspend Mode Wake-Up Timing Controls

When exiting Suspend mode, the wake-up sequence for the FPGA is programmable and controlled by a single clock.

Wake-Up Timing Clock Source (sw_clk)

The wake-up timing when exiting Suspend mode is controlled by a selectable clock source as shown in Figure 6 and described in Table 4. The clock source is defined by one or two bitstream generator options, `sw_clk` and sometimes `StartupClk`.

The internal oscillator is disabled during Suspend mode to conserve power.



UG331_c19_06_092606

Figure 6: Suspend Mode Wake-Up Timing Control Clock Selection

- The `sw_clk` option is specific to the Suspend feature. By default, `sw_clk:InternalClk`.
- The `StartupClk` option is available on every application. The same option used to clock the start-up process at the end of configuration can be used to clock the wake-up process at the end of Suspend. By default `StartupClk:Cclk`, but using this for Suspend wake-up requires that the Slave configuration mode be persisted. When using `sw_clk:StartupClk` and `StartupClk:Cclk`, the CCLK pin becomes the clock source when exiting Suspend mode.

Table 4: Clock Sources to Wake-Up from Suspend Mode

sw_clk Setting	StartupClk Setting	Clock Source	Restriction
InternalClk	--	Internal Oscillator	The oscillator has an imprecise frequency of about 50 MHz.
StartupClk	Cclk	CCLK pin on FPGA	This option is only available for FPGAs using Slave configuration mode. The bitstream option Persist: Yes must be set. This option is not available for FPGAs using the Master configuration mode; use InternalClk instead.
	JtagClk	TCK pin on FPGA	The JTAG interface must be active to exit Suspend mode.
	UserClk	CLK input on the STARTUP_SPARTAN3A design primitive	The clock input to the STARTUP design primitive can originate from any nonclocked signal in the FPGA. It cannot originate from a flip-flop source because all clocked elements are write-protected while in Suspend mode. The inputs and interconnect are enabled before wake-up begins, a delay of $t_{\text{SUSPEND_ENABLE}}$ after Suspend goes Low.

Switch Outputs from Suspend to Normal Behavior (sw_gts_cycle)

The Suspend/Wake [sw_gts_cycle](#) bitstream option controls when I/O pins are released from their SUSPEND constraint settings and returned to normal operation. The timing is controlled by the “Wake-Up Timing Clock Source (sw_clk)” described earlier. The default [sw_gts_cycle](#) setting is 4 cycles, but this control can be set for any value between 1 and 1,024 clock cycles.

This control becomes active after the AWAKE pin goes High. After the specified number of clock cycles, all output, open-drain output, and bidirectional I/O pins transition from their Suspend behavior, either the default [3STATE](#) or individually specified using a [SUSPEND Constraint](#), back to the normal behavior specified in the original FPGA application.

It is best to release the outputs before releasing the write-protect lock on all clocked elements.

Release Write Protect on Clocked Elements (sw_gwe_cycle)

The Suspend/Wake [sw_gwe_cycle](#) bitstream option controls when the write-protect lock is released on all clocked elements.

The timing is controlled by the [Wake-Up Timing Clock Source \(sw_clk\)](#) described above. The default [sw_gwe_cycle](#) setting is 5 cycles, but this control can be set for any value between 1 and 1,024 clock cycles.

This control becomes active after the AWAKE pin goes High. After the specified number of clock cycles, the write-protect lock is released from all writable, clocked elements such as flip-flops, block RAM, etc.

If the [en_sw_gsr:yes](#) option was set, then the clocked elements are already globally set or reset to the value specified in the original FPGA design before the write-protect lock is released. If [en_sw_gsr:no](#), then the state of the FPGA after entering Suspend mode is preserved.

It is best to release the outputs before releasing the write-protect lock on all clocked elements.

Dedicated Configuration Pins Unaffected During Suspend Mode

The following dedicated configuration pins are unaffected when the FPGA is in Suspend mode:

- JTAG pins TDI, TMS, TCK, and TDO
- DONE pin
- PROG_B pin

SUSPEND Pin

When the Suspend feature is enabled (see “[Enable the Suspend Feature](#),” page 6), the SUSPEND pin controls when the FPGA enters Suspend mode. During normal FPGA operation, the SUSPEND pin must be Low. When High, the SUSPEND pin forces the FPGA into the low-power Suspend mode after the delays shown in [Table 1](#), page 6. [Table 5](#) describes the functionality of the SUSPEND pin.

If the Suspend feature is not enabled for an application (the application never enters low-power mode), then connect the SUSPEND pin to GND.

Table 5: SUSPEND Pin Functionality

en_suspend Setting	SUSPEND Pin	Function
no (default) Suspend mode disabled	X	Active mode. The Suspend feature is disabled. The SUSPEND pin is unused and ignored. Connect the SUSPEND pin to GND.
yes Suspend mode enabled	0	Active mode. The FPGA performs the application described in the bitstream loaded into the FPGA during configuration. When the SUSPEND pin changes from High to Low, wake the FPGA from Suspend mode.
	1	Suspend mode. When the SUSPEND pin changes from Low to High, force the FPGA to enter power-saving Suspend mode.

Characteristics

The SUSPEND pin is an LVCMOS/LVTTL receiver, and power to the input buffer is supplied by the V_{CCAUX} power rail. The SUSPEND pin has no pull-up resistors during configuration, and the PUDC_B control has no affect on the SUSPEND pin.

SUSPEND Input Glitch Filter

The SUSPEND pin has a programmable glitch filter to guard against short pulses, which could cause the FPGA to spuriously enter Suspend mode. Turning off the filter allows the FPGA to enter or exit SUSPEND mode more quickly, but the application must guard against spurious pulses. The difference in delay is the $t_{SUSPENDFILTER}$ value in [Table 1](#), page 6.

Via User Constraints File (UCF)

The SUSPEND filter is set as part of the ENABLE_SUSPEND constraint, as described in “[Via User Constraints File \(UCF\)](#),” page 6.

Bitstream Generator (BitGen) Option

The filter can also be enabled via a bitstream generator option:

```
bitgen -g suspend_filter:Yes
```

Effect on FPGA Configuration

Suspend mode is activated by an FPGA configuration bitstream option. Consequently, the SUSPEND pin has no effect on configuration.

If Suspend mode is enabled in the bitstream and the SUSPEND pin is High, the FPGA successfully configures and then immediately enters Suspend mode. The FPGA's DONE pin will be High, but the AWAKE pin will be Low.

Tie SUSPEND to GND if not Using Suspend Mode

If not using Suspend mode, connect the SUSPEND pin to GND. Do not leave the pin floating.

AWAKE Pin

The AWAKE pin optionally provides status on the Suspend power-savings mode.

General Behavior (Suspend Feature Disabled)

Unless the Suspend feature is enabled, the AWAKE pin is a general-purpose user-I/O pin.

AWAKE Pin Behavior when Suspend Feature is Enabled

If the Suspend feature is enabled, then the AWAKE pin indicates the present state of the FPGA, as summarized in Table 6. The AWAKE pin cannot be used by the FPGA application as a general-purpose I/O pin.

Table 6: AWAKE Pin Status

AWAKE Pin	Indication
0	The FPGA is presently in the low-power Suspend mode.
1	The FPGA is active.

The AWAKE pin can further be configured as an open-drain output (the default) or a full-swing output driver, as shown in Figure 7. This behavior is controlled by a bitstream generator (BitGen) option:

```
bitgen -g drive_aware:no
```

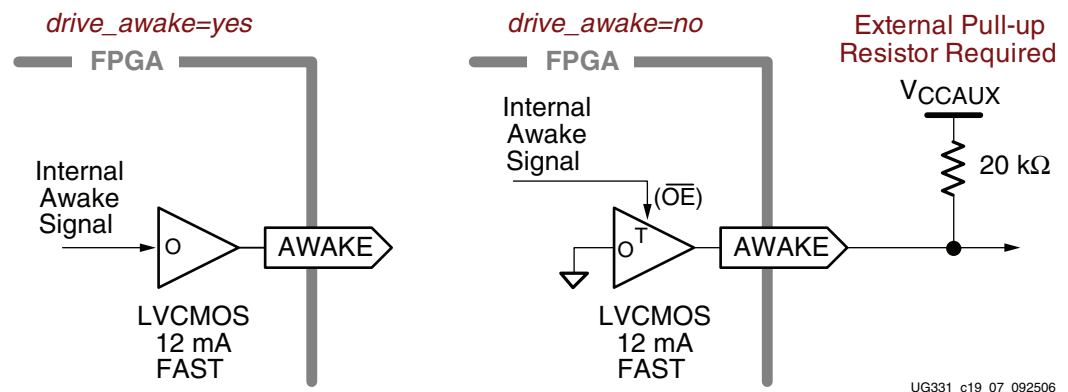


Figure 7: AWAKE Output Drive Options if Suspend Mode Enabled

The AWAKE output pin is supplied by the V_{CCAUX} power rail.

When `drive_aware:yes`, the AWAKE pin is an active output driver, equivalent to a user I/O configured as LVC MOS, with 12 mA output drive and a Fast slew rate.

Controlling Wake-Up from an External Source

By default `drive_aware:no`. When `drive_aware:no`, the AWAKE pin is an open-drain output capable of sinking 12 mA. In this case, an external pull-up resistor is required to exit Suspend mode. The resistor value should be high to minimize the amount of current flow during Suspend mode. The resistor needs to be strong enough to overcome the I/O pin leakage. A large resistor value also equates to a longer AWAKE rise time. The FPGA does not exit Suspend mode and begin the wake-up process until AWAKE goes High.

Synchronizing Wake-Up

The wake-up process can be synchronized across multiple FPGAs or between the FPGAs and the system by using one SUSPEND signal to control multiple devices. The AWAKE pin can also be used to synchronize multiple devices. Multiple FPGAs can have their AWAKE pins tied together to a single pull-up resistor in order to start their wake-up processes at the same time. If [sw_clk:StartupClk](#) and [StartupClk:UserClk](#), the wake-up counters can also be synchronized.

Holding the AWAKE pin Low delays the transition from Suspend mode to Active mode by holding off the [sw_gwe_cycle](#) and [sw_gts_cycle](#) counters, and allows an external controller to decide when to begin the wake-up process in the FPGA.

JTAG Operations Allowed During Suspend Mode

[Table 7](#) shows the JTAG operations permitted when the FPGA is in Suspend mode. Executing these JTAG operations increases the FPGA's power consumption while in Suspend mode.

Table 7: JTAG Operations Allowed during Suspend Mode

Boundary Scan Command	Description
IDCODE	Read the JTAG ID code that describes the FPGA array type in the JTAG chain. This value is different from the Device DNA identifier, which is unique to every device.
BYPASS	Enables BYPASS.
USERCODE	Read the user-defined code embedded in the FPGA bitstream.

Do not use any other JTAG instructions when in Suspend mode or while transitioning into and out of Suspend mode. Furthermore, do not enter Suspend mode when performing a Readback operation.

Post-Configuration CRC Limitations When Using Suspend Mode

If an application uses the post-configuration CRC feature and an error occurs, do not enter Suspend mode. The FPGA will not wake from Suspend mode without reprogramming, such as asserting PROG_B or power-cycling the FPGA.

Several design options are possible:

1. Do not use the post-configuration CRC feature when the Suspend mode feature is enabled and *vice versa*.
2. Always reprogram the device when a CRC error occurs.

For more information, see the "Configuration CRC" chapter in [UG332: Spartan-3 Generation Configuration User Guide](#).

Suspend Mode Bitstream Generator Options

Table 8 summarizes the various bitstream options associated with Suspend mode.

Table 8: Suspend Mode Bitstream Generator Options

Suspend Mode Bitstream Options	Options (default)	Description
en_suspend	No	Suspend mode is not used in this application. Connect the SUSPEND pin to GND.
	Yes	Enables the power-saving Suspend feature, controlled by the SUSPEND pin.
drive_aware	No	If Suspend mode is enabled, indicates the present status on AWAKE using an open-drain output. An external pull-up resistor or High signal is required to exit SUSPEND mode.
	Yes	If Suspend mode is enabled, indicates the present status by actively driving the AWAKE output.
suspend_filter	Yes	Enables the glitch filter on the SUSPEND pin.
	No	Disables the glitch filter on the SUSPEND pin.
en_sw_gsr	No	The state of all clocked elements in the FPGA is preserved.
	Yes	Pulses the GSR signal during wake-up, setting or resetting all clocked elements, as originally specified in the FPGA application. The GSR pulse occurs before the AWAKE pin goes High and before the sw_gwe_cycle and sw_gts_cycle settings are active.
sw_clk	StartupClk	Uses the clock defined by the StartupClk bitstream generator setting to control the Suspend wake-up timing.
	InternalClk	Uses the internally generated 50 MHz oscillator to control the Suspend wake-up timing.
sw_gwe_cycle	1,..., <u>5</u> ,...,1024	After the AWAKE pin is High, indicates the number of clock cycles as defined by the sw_clk setting, when the global write-protect lock is released for writable clocked elements (flip-flops, block RAM, etc.). The default value is five clock cycles after the AWAKE pin goes High. Generally, this value is equal to or greater than the sw_gts_cycle setting.
sw_gts_cycle	1,..., <u>4</u> ,...,1024	After the AWAKE pin is High, indicates the number of clock cycles as defined by the sw_clk setting, when the I/O pins switch from their SUSPEND Constraint settings back to their normal functions. The default value is four clock cycles after the AWAKE pin goes High. Generally, this value is equal to or less than the sw_gwe_cycle setting.

FPGA Voltage Requirements During Suspend Mode

During Suspend mode, the V_{CCINT} and V_{CCAUX} rails must remain powered at their specified data sheet levels. However, the V_{CCO} supply to each of the four I/O banks can be potentially turned off to conserve additional power, depending on system requirements. Optionally, V_{CCO} can be reduced to 1.0V during Suspend mode, but this also affects the voltage levels for any output pin with a `SUSPEND=DRIVE_LAST_VALUE` constraint.

The FPGA's power-on reset (POR) circuit continues to monitor the V_{CCINT} and V_{CCAUX} supplies. The POR circuit does not monitor the V_{CCO} supplies after configuration. By default, if the V_{CCINT} or V_{CCAUX} supply dips below the minimum specified data sheet voltage limit, then the FPGA restarts configuration.

In very power-sensitive applications, it is possible to reduce the V_{CCINT} and V_{CCAUX} voltage levels even further to conserve additional power. However, the POR circuit must be disabled in the FPGA bitstream using the `en_porb:No` bitstream option. With the POR circuit disabled, V_{CCINT} can be reduced to 1.0V and V_{CCAUX} can be reduced to 2.0V while the FPGA is in Suspend mode.

Typical Supply Requirements During Suspend Mode

Power supply requirements are design dependent but the following provides some general expectations and examples. When entering Suspend mode, the FPGA exhibits the following typical characteristics on the V_{CCINT} and V_{CCAUX} power rails:

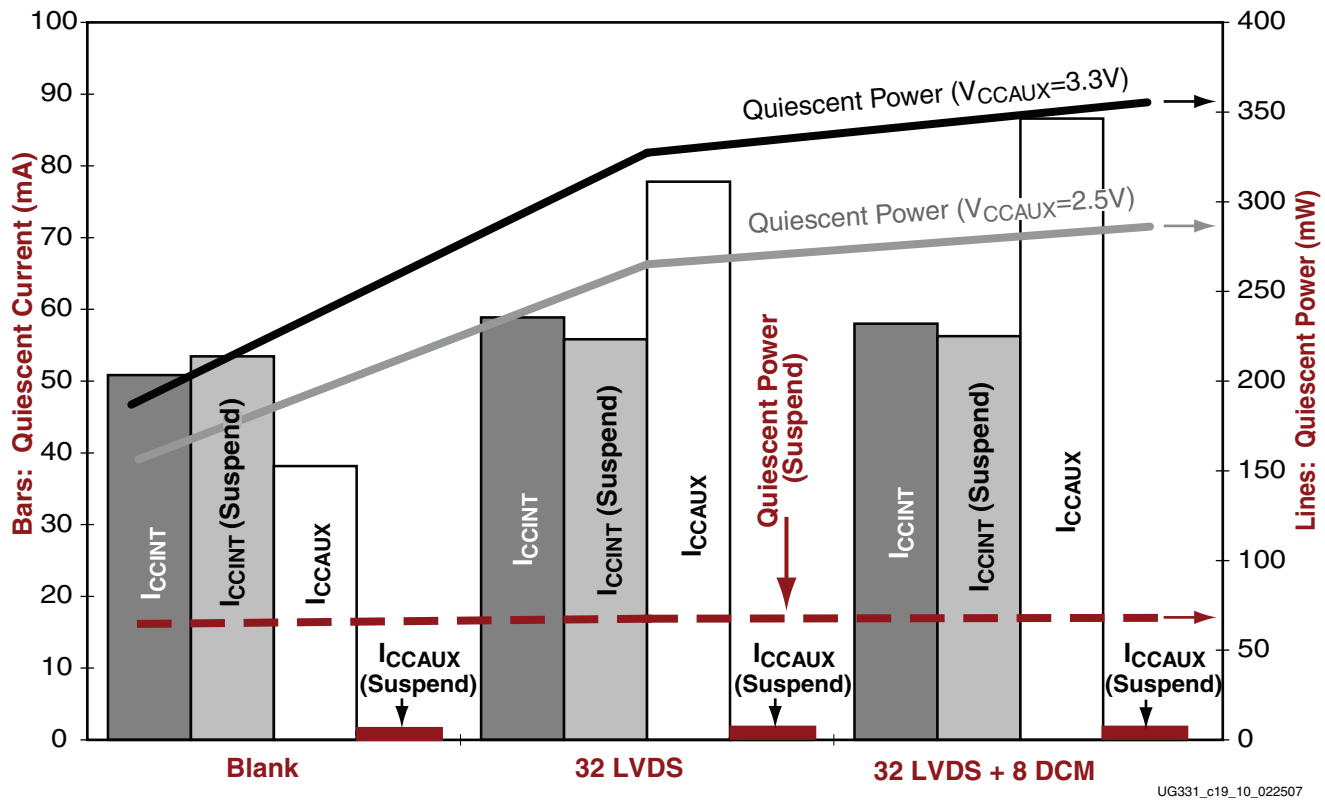
- The current required on the V_{CCAUX} supply drops significantly as the internal FPGA circuits powered by V_{CCAUX} are internally switched over to the V_{CCINT} supply during Suspend mode.
- The current required on the V_{CCINT} supply may increase slightly from its quiescent current level.

Design Examples

Figure 8 graphically demonstrates the effect that Suspend mode has on some example, representative designs measured on a typical XC3S1400A FPGA. The results for other array sizes roughly scale with device density. The Suspend mode primarily affects current consumption on the V_{CCINT} and V_{CCAUX} power rails; there are also power savings for the V_{CCO} rail, depending on the how the user-programmable `SUSPEND` constraints are defined in the application (see “[Define the I/O Behavior During Suspend Mode](#),” page 7).

Figure 8 includes three example designs that highlight the Suspend mode behavior:

- **Blank:** A blank FPGA design. No logic is used in this application. A blank design provides the lowest quiescent current and establishes the baseline power consumption.
- **32LVDS:** A design that includes 32 LVDS differential input channels (64 pins) connected to 32 LVDS differential output channels (64 pins). The differential I/O buffers are powered by the V_{CCAUX} voltage supply.
- **32LVDS+8DCM:** A design that includes the circuitry described for 32LVDS plus eight Digital Clock Managers (DCMs). The differential I/O buffers and DCMs are powered by the V_{CCAUX} voltage supply.



UG331_c19_10_022507

Figure 8: Effects of Spartan-3A Suspend Mode on Example Designs Measured on Typical XC3S1400A

Figure 8 also shows four bars, indicating the typical quiescent current on the V_{CCINT} and V_{CCAUX} supplies under normal quiescent conditions with all clocks stopped, and adjacent bars for current during Suspend mode. The associated current measurement, in mA, appears along the left-side vertical axis. Note that the current on V_{CCAUX} during Suspend mode is near the base of the chart, highlighted in burgundy.

Furthermore, Figure 8 shows the total quiescent power (current multiplied by the voltage applied to each power rail). The associated resulting power measurement, in mW, appears on the right-side vertical axis. V_{CCINT} is 1.2V nominally. On Spartan-3A and Spartan-3A DSP FPGAs, V_{CCAUX} can be either 2.5V or 3.3V nominally. By physics, the quiescent power is lower when $V_{CCAUX} = 2.5V$. Note the significant reduction in total power consumption when the FPGA is in Suspend mode. Although the total power savings is design dependent, Suspend mode typically reduces power consumption by 40% or more, with a minimum power savings of about 20%.

During Suspend mode, some of the circuitry powered by the V_{CCAUX} supply is switched over to the V_{CCINT} supply. Note the **Blank** design example in Figure 8. The current on the V_{CCINT} supply actually *increases* while the current on the V_{CCAUX} supply drops significantly! However, the total V_{CCINT} current during Suspend remains below that used in an active FPGA application. Furthermore, despite the increased V_{CCINT} current, the overall system power is reduced because current is being switched from the 2.5V or 3.3V V_{CCAUX} supply to the 1.2V V_{CCINT} supply.

The power savings are more pronounced in the **32LVDS** and **32LVDS+8DCM** examples and both designs use circuitry that consumes current on the V_{CCAUX} supply.

Suspend Mode Effect on In-System Flash in Spartan-3AN Platform FPGAs

The Suspend pin has no direct effect on the In-System Flash memory in the Spartan-3AN platform FPGAs. However, whenever the FPGA is in a quiescent state, as in Suspend mode, the flash memory will also be in a quiescent state. The quiescent power consumption is extremely low. See the [Spartan-3AN FPGA Data Sheet](#) for details.

References

1. [UG331](#), *Spartan-3 Generation FPGA User Guide*
2. [DS529](#), *Spartan-3A FPGA Data Sheet*
3. [DS557](#), *Spartan-3AN FPGA Data Sheet*
4. [DS610](#), *Spartan-3A DSP FPGA Data Sheet*

For additional information on the SUSPEND constraint, see the Constraints Guide.

- **Constraints Guide** for ISE™ 9.1i
<http://toolbox.xilinx.com/docsan/xilinx9/books/docs/cgd/cgd.pdf>

Conclusion

While some applications require the lowest possible system cost or highest performance, still other applications require the lowest possible standby power. Spartan-3 Generation FPGAs offer low-power options that balance cost and performance trade-offs.

The Spartan-3A, Spartan-3AN, and Spartan-3A DSP FPGA platforms offer an advanced power management feature called Suspend mode, which reduces FPGA power consumption while retaining the FPGA's configuration data and application state. Suspend mode retains all data and offers fast wake-up times.

Appendix - Hibernate Mode

Another method to reduce power is to use the Hibernate mode. Hibernate mode provides the maximum possible power savings for applications that can be turned off for long periods of time and that can afford to lose the present application state.

Hibernate mode provides further power savings by switching off power rails. In the Spartan-3A, Spartan-3AN, and Spartan-3A DSP platforms, all power rails can be turned off. Power FETs with low “on” resistance are recommended to perform the switching action. Configuration data is lost upon entering Hibernate mode; therefore, reconfiguration is required after exiting the mode.

Hibernate mode is also supported in the Spartan-3 and Spartan-3E FPGA platforms, but these devices require different treatment of the V_{CCO} supply. For the Spartan-3 and Spartan-3E platforms, it is safest to maintain V_{CCO} power for all banks throughout the Hibernation period to keep the power diodes inside the I/O block turned off when signals are applied to the pins. In Hibernate mode, the powered V_{CCO} rails account for little current because the I/Os are in a high-impedance state. Under certain conditions, it is also possible to switch off the V_{CCO} rail to a particular bank.

For more detail on the Hibernate mode, see the Power Management chapter of [UG331](#), Spartan-3 Generation FPGA User Guide. [Table 9](#) compares the Suspend and Hibernate options for the Spartan-3 Generation platforms.

Table 9: Spartan-3 Generation Power-Saving Options

	Spartan-3A/3AN/3A DSP Suspend Mode	Spartan-3A/3AN/3A DSP Hibernate Mode	Spartan-3/3E Hibernate Mode
Configuration data retained	Yes	No	
Application state retained (flip-flops, block RAM, distributed RAM, SRL16)	Yes	No	
Time to exit from power-saving mode	Approximately 500 μ s + DCM lock time + programmable timing	FPGA configuration time (tens of milliseconds)	
Power consumption while in power-saving mode	Low	Lowest	
Power supplies	Maintained or scaled down to save additional power	Removed	Removed except for V_{CCO}

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/02/07	1.0	Initial Xilinx release.