FPGA States of Operation

Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration, and start-up. Figure 49 outlines these three FPGA states.

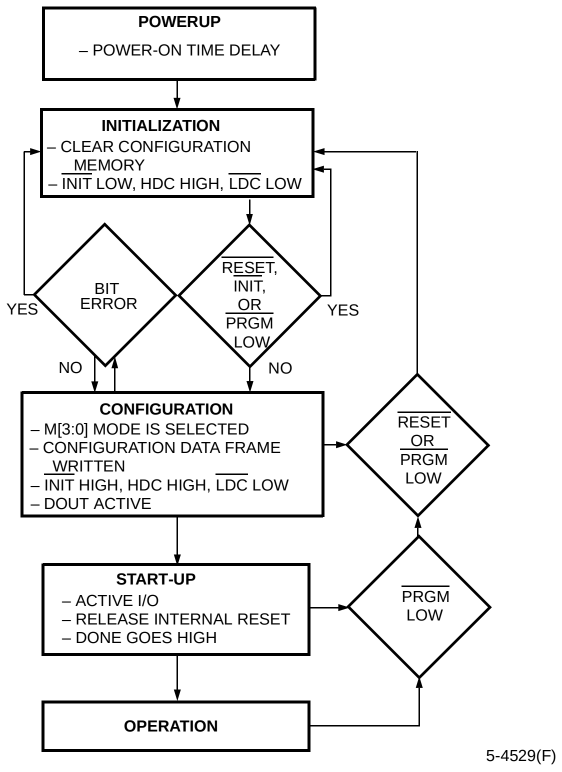

POWERUP
– POWER-ON TIME DELAY
INITIALIZATION
– CLEAR CONFIGURATION MEMORY
– $\overline{\text{INIT}}$ LOW, HDC HIGH, $\overline{\text{LDC}}$ LOW
BIT ERROR
YES
NO
$\overline{\text{RESET}}$, $\overline{\text{INIT}}$, OR $\overline{\text{PRGM}}$ LOW
YES
NO
CONFIGURATION
– M[3:0] MODE IS SELECTED
– CONFIGURATION DATA FRAME WRITTEN
– $\overline{\text{INIT}}$ HIGH, HDC HIGH, $\overline{\text{LDC}}$ LOW
– DOUT ACTIVE
$\overline{\text{RESET}}$ OR $\overline{\text{PRGM}}$ LOW
START-UP
– ACTIVE I/O
– RELEASE INTERNAL RESET
– DONE GOES HIGH
$\overline{\text{PRGM}}$ LOW
OPERATION
5-4529(F)

Figure 49. FPGA States of Operation

Initialization

Upon powerup, the device goes through an initialization process. First, an internal power-on-reset circuit is triggered when power is applied. When V_{DD} reaches the voltage at which portions of the FPGA begin to operate (2.5 V to 3 V for the OR3Cxx, 2.2 V to 2.7 V for the OR3Txxx), the I/Os are configured based on the configuration mode, as determined by the mode select inputs M[2:0]. A time-out delay is initiated when V_{DD} reaches between 3.0 V and 4.0 V (OR3Cxx) or 2.7 V to 3.0 V (OR3Txxx) to allow the power supply voltage to stabilize. The $\overline{\text{INIT}}$ and DONE outputs are low. At powerup, if V_{DD} does not rise from 2.0 V to V_{DD} in less than 25 ms, the user should delay configuration by inputting a low into $\overline{\text{INIT}}$, $\overline{\text{PRGM}}$, or $\overline{\text{RESET}}$ until V_{DD} is greater than the recommended minimum operating voltage (4.75 V for OR3Cxx commercial devices and 3.0 V for OR3Txxx devices).

At the end of initialization, the default configuration option is that the configuration RAM is written to a low state. This prevents shorts prior to configuration. As a configuration option, after the first configuration (i.e., at reconfiguration), the user can reconfigure without clearing the internal configuration RAM first. The active-low, open-drain initialization signal $\overline{\text{INIT}}$ is released and must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more $\overline{\text{INIT}}$ pins should be wire-ANDed. If $\overline{\text{INIT}}$ is held low by one or more FPGAs or an external device, the FPGA remains in the initialization state. $\overline{\text{INIT}}$ can be used to signal that the FPGAs are not yet initialized. After $\overline{\text{INIT}}$ goes high for two internal clock cycles, the mode lines (M[3:0]) are sampled, and the FPGA enters the configuration state.

The high during configuration (HDC), low during configuration ($\overline{\text{LDC}}$), and DONE signals are active outputs in the FPGA's initialization and configuration states. HDC, $\overline{\text{LDC}}$, and DONE can be used to provide control of external logic signals such as reset, bus enable, or PROM enable during configuration. For parallel master configuration modes, these signals provide PROM enable control and allow the data pins to be shared with user logic signals.

FPGA States of Operation (continued)

If configuration has begun, an assertion of $\overline{\text{RESET}}$ or $\overline{\text{PRGM}}$ initiates an abort, returning the FPGA to the initialization state. The $\overline{\text{PRGM}}$ and $\overline{\text{RESET}}$ pins must be pulled back high before the FPGA will enter the configuration state. During the start-up and operating states, only the assertion of $\overline{\text{PRGM}}$ causes a reconfiguration.

In the master configuration modes, the FPGA is the source of configuration clock (CCLK). In this mode, the initialization state is extended to ensure that, in daisy-chain operation, all daisy-chained slave devices are ready. Independent of differences in clock rates, master mode devices remain in the initialization state an additional six internal clock cycles after $\overline{\text{INIT}}$ goes high.

When configuration is initiated, a counter in the FPGA is set to 0 and begins to count configuration clock cycles applied to the FPGA. As each configuration data frame is supplied to the FPGA, it is internally assembled into data words. Each data word is loaded into the internal configuration memory. The configuration loading process is complete when the internal length count equals the loaded length count in the length count field, and the required end of configuration frame is written.

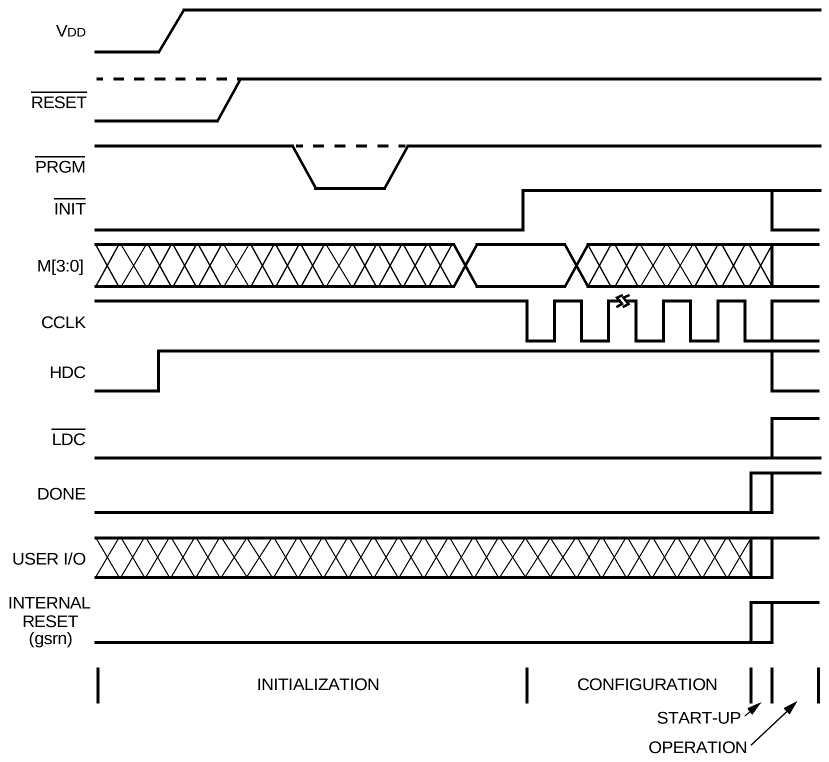
All OR3Cxx I/Os operate as TTL inputs during configuration (OR3Txxx I/Os are CMOS-only). All I/Os that are not used during the configuration process are 3-stated with internal pull-ups.

Warning: During configuration, all OR3Txxx inputs have internal pull-ups enabled. If these inputs are driven to 5V, they will draw substantial current (≅ 5 ma). This is due to the fact that the inputs are pulled up to 3V.

During configuration, the PIC and PLC latches/FFs are held set/reset and the internal BIDI buffers are 3-stated. The combinatorial logic begins to function as the FPGA is configured. Figure 50 shows the general waveform of the initialization, configuration, and start-up states.

Configuration

The *ORCA* Series FPGA functionality is determined by the state of internal configuration RAM. This configuration RAM can be loaded in a number of different modes. In these configuration modes, the FPGA can act as a master or a slave of other devices in the system. The decision as to which configuration mode to use is a system design issue. Configuration is discussed in detail, including the configuration data format and the configuration modes used to load the configuration data in the FPGA, following a description of the start-up state.



5-4482(F)

Figure 50. Initialization/Configuration/Start-Up Waveforms

FPGA States of Operation (continued)

Start-Up

After configuration, the FPGA enters the start-up phase. This phase is the transition between the configuration and operational states and begins when the number of CCLKs received after $\overline{\text{INIT}}$ goes high is equal to the value of the length count field in the configuration frame and when the end of configuration frame has been written. The system design issue in the start-up phase is to ensure the user I/Os become active without inadvertently activating devices in the system or causing bus contention. A second system design concern is the timing of the release of global set/reset of the PLC latches/FFs.

There are configuration options that control the relative timing of three events: DONE going high, release of the set/reset of internal FFs, and user I/Os becoming active. Figure 51 shows the start-up timing for *ORCA* FPGAs. The system designer determines the relative timing of the I/Os becoming active, DONE going high, and the release of the set/reset of internal FFs. In the *ORCA* Series FPGA, the three events can occur in any arbitrary sequence. This means that they can occur before or after each other, or they can occur simultaneously.

There are four main start-up modes: CCLK_NOSYNC, CCLK_SYNC, UCLK_NOSYNC, and UCLK_SYNC. The only difference between the modes starting with CCLK and those starting with UCLK is that for the UCLK modes, a user clock must be supplied to the start-up logic. The timing of start-up events is then based upon this user clock, rather than CCLK. The difference between the SYNC and NOSYNC modes is that for SYNC mode, the timing of two of the start-up events, release of the set/reset of internal FFs, and the I/Os becoming active is triggered by the rise of the external DONE pin followed by a variable number of rising clock edges (either CCLK or UCLK). For the NOSYNC mode, the timing of these two events is based only on either CCLK or UCLK.

DONE is an open-drain bidirectional pin that may include an optional (enabled by default) pull-up resistor to accommodate wired ANDing. The open-drain DONE signals from multiple FPGAs can be tied together (ANDed) with a pull-up (internal or external) and used as an active-high ready signal, an active-low PROM enable, or a reset to other portions of the system. When used in SYNC mode, these ANDed DONE pins can be used to synchronize the other two start-up events, since they can all be synchronized to the same external signal. This signal will not rise until all FPGAs release their DONE pins, allowing the signal to be pulled high.

The default for *ORCA* is the CCLK_SYNC synchronized start-up mode where DONE is released on the first CCLK rising edge, C1 (see Figure 51). Since this is a synchronized start-up mode, the open-drain DONE signal can be held low externally to stop the occurrence of the other two start-up events. Once the DONE pin has been released and pulled up to a high level, the other two start-up events can be programmed individually to either happen immediately or after up to four rising edges of CCLK (Di, Di + 1, Di + 2, Di + 3, Di + 4). The default is for both events to happen immediately after DONE is released and pulled high.

A commonly used design technique is to release DONE one or more clock cycles before allowing the I/O to become active. This allows other configuration devices, such as PROMs, to be disconnected using the DONE signal so that there is no bus contention when the I/Os become active. In addition to controlling the FPGA during start-up, other start-up techniques that avoid contention include using isolation devices between the FPGA and other circuits in the system, reassigning I/O locations, and maintaining I/Os as 3-stated outputs until contentions are resolved.

Each of these start-up options can be selected during bit stream generation in *ORCA* Foundry, using Advanced Options. For more information, please see the *ORCA* Foundry documentation.

FPGA States of Operation (continued)

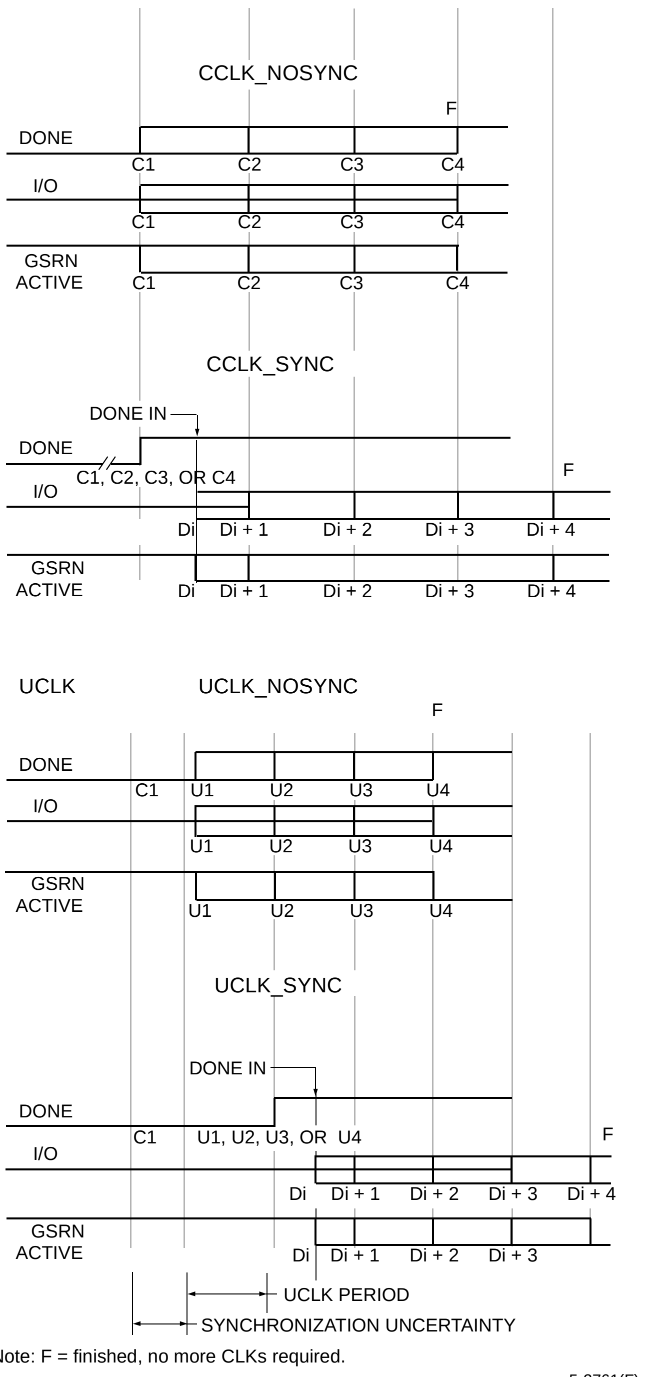


Figure 51. Start-Up Waveforms

Reconfiguration

To reconfigure the FPGA when the device is operating in the system, a low pulse is input into $\overline{\text{PRGM}}$. The configuration data in the FPGA is cleared, and the I/Os not used for configuration are 3-stated. The FPGA then samples the mode select inputs and begins reconfiguration. When reconfiguration is complete, DONE is released, allowing it to be pulled high.

Partial Reconfiguration

All *ORCA* device families have been designed to allow a partial reconfiguration of the FPGA at any time. This is done by setting a bit stream option in the previous configuration sequence that tells the FPGA to not reset all of the configuration RAM during a reconfiguration. Then only the configuration frames that are to be modified need to be rewritten, thereby reducing the configuration time.

Other bit stream options are also available that allow one portion of the FPGA to remain in operation while a partial reconfiguration is being done. If this is done, the user must be careful to not cause contention between the two configurations (the bit stream resident in the FPGA and the partial reconfiguration bit stream) as the second reconfiguration bit stream is being loaded.

Other Configuration Options

There are many other configuration options available to the user that can be set during bit stream generation in *ORCA* Foundry. These include options to enable boundary scan and/or the microprocessor interface (MPI) and/or the programmable clock manager (PCM), readback options, and options to control and use the internal oscillator after configuration.

Other useful options that affect the next configuration (not the current configuration process) include options to disable the global set/reset during configuration, disable the 3-state of I/Os during configuration, and disable the reset of internal RAMs during configuration to allow for partial configurations (see above). For more information on how to set these and other configuration options, please see the *ORCA* Foundry documentation.

Configuration Data Format

The *ORCA* Foundry Development System interfaces with front-end design entry tools and provides tools to produce a fully configured FPGA. This section discusses using the *ORCA* Foundry Development System to generate configuration RAM data and then provides the details of the configuration frame format.

The *ORCA* OR3Cxx and OR3Txxx Series FPGAs are bit stream compatible.

Using *ORCA* Foundry to Generate Configuration RAM Data

The configuration data bit stream defines the I/O functionality, logic, and interconnections within the FPGA. The bit stream is generated by the development system. The bit stream created by the bit stream generation tool is a series of 1s and 0s used to write the FPGA configuration RAM. It can be loaded into the FPGA using one of the configuration modes discussed later.

In the bit stream generator, the designer selects options that affect the FPGA's functionality. Using the output of the bit stream generator, **circuit_name.bit**, the development system's download tool can load the configuration data into the *ORCA* series FPGA evaluation board from a PC or workstation.

Alternatively, a user can program a PROM (such as a Serial ROM or a standard EPROM) and load the FPGA from the PROM. The development system's PROM programming tool produces a file in .mks or .exo format.

Configuration Data Frame

Configuration data can be presented to the FPGA in two frame formats: autoincrement and explicit. A detailed description of the frame formats is shown in Figure 52, Figure 53, and Table 32. The two modes are similar except that autoincrement mode uses assumed address incrementation to reduce the bit stream size, and explicit mode requires an address for each data frame. In both cases, the header frame begins with a series of 1s and a preamble of 0010, followed by a 24-bit length count field representing the total number of configuration clocks needed to complete the loading of the FPGAs.

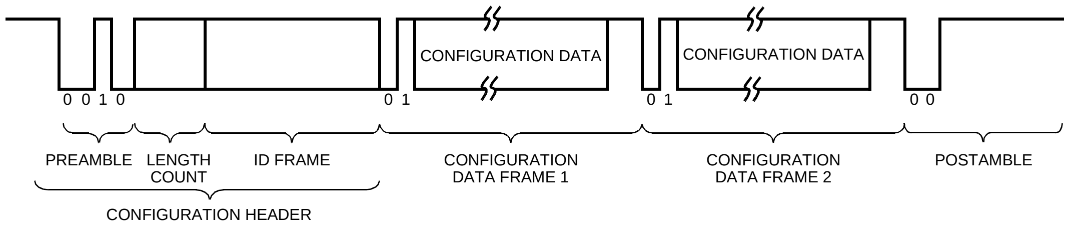
Following the header frame is a mandatory ID frame. (Note that the ID frame was optional in the *ORCA* 2C and 2C/TxxA Series.)

The ID frame contains data used to determine if the bit stream is being loaded to the correct type of *ORCA* FPGA (i.e., a bit stream generated for an OR3C55 is being sent to an OR3C55). Error checking is always enabled for Series 3 devices, through the use of an 8-bit checksum. One bit in the ID frame also selects between the autoincrement and explicit address modes for this load of the configuration data.

A configuration data frame follows the ID frame. A data frame starts with a 01-start bit pair and ends with enough 1-stop bits to reach a byte boundary. If using autoincrement configuration mode, subsequent data frames can follow. If using explicit mode, one or more address frames must follow each data frame, telling the FPGA at what addresses the preceding data frame is to be stored (each data frame can be sent to multiple addresses).

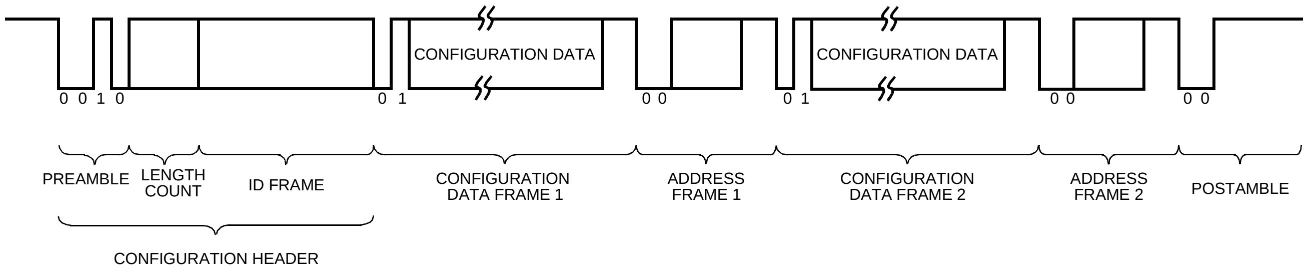
Following all data and address frames is the postamble. The format of the postamble is the same as an address frame with the highest possible address value with the checksum set to all ones.

Configuration Data Format (continued)



5-5759(F)

Figure 52. Serial Configuration Data Format—Autoincrement Mode



5-5760(F)

Figure 53. Serial Configuration Data Format—Explicit Mode

Table 32. Configuration Frame Format and Contents

Header	11110010	Preamble
	24-bit Length Count	Configuration frame length.
	11111111	Trailing header—8 bits.
ID Frame	0101 1111 1111 1111	ID frame header.
	Configuration Mode	00 = autoincrement, 01 = explicit.
	Reserved [41:0]	Reserved bits set to 0.
	ID	20-bit part ID.
	Checksum	8-bit checksum.
	11111111	Eight stop bits (high) to separate frames.
Configuration Data Frame (repeated for each data frame)	01	Data frame header.
	Data Bits	Number of data bits depends upon device.
	Alignment Bits = 0	String of 0 bits added to bit stream to make frame header, plus data bits reach a byte boundary.
	Checksum	8-bit checksum.
	11111111	Eight stop bits (high) to separate frames.
Configuration Address Frame	00	Address frame header.
	14 Address Bits	14-bit address of location to start data storage.
	Checksum	8-bit checksum.
	11111111	Eight stop bits (high) to separate frames.
Postamble	00	Postamble header.
	11111111 111111	Dummy address.
	1111111111111111	16 stop bits.*

* In MPI configuration mode, the number of stop bits = 32.

Note: For slave parallel mode, the byte containing the preamble must be 11110010. The number of leading header dummy bits must be (n * 8) + 4, where n is any nonnegative integer and the number of trailing dummy bits must be (n * 8), where n is any positive integer. The number of stop bits/frame for slave parallel mode must be (x * 8), where x is a positive integer. Note also that the bit stream generator tool supplies a bit stream that is compatible with all configuration modes, including slave parallel mode.

Configuration Data Format (continued)

The length and number of data frames and information on the PROM size for the Series 3 FPGAs are given in Table 33.

Table 33. Configuration Frame Size

Devices	OR3T20	OR3T30	OR3C/T55	OR3C/T80	OR3T125
# of Frames	856	984	1240	1496	1880
Data Bits/Frame	202	232	292	352	442
Configuration Data (# of frames x # of data bits/frame)	172,912	228,288	362,080	526,592	830,960
Maximum Total # Bits/Frame (align bits, 01 frame start, 8-bit checksum, 8 stop bits)	224	256	312	376	464
Maximum Configuration Data (# bits/frame x # of frames)	191,744	251,904	386,880	562,496	872,320
Maximum PROM Size (bits) (add configuration header and postamble)	191,912	252,072	387,048	562,664	872,488

Bit Stream Error Checking

There are three different types of bit stream error checking performed in the *ORCA* Series 3 FPGAs:
ID frame, frame alignment, and CRC checking.

The ID data frame is sent to a dedicated location in the FPGA. This ID frame contains a unique code for the device for which it was generated. This device code is compared to the internal code of the FPGA. Any differences are flagged as an ID error. This frame is automatically created by the bit stream generation program in *ORCA* Foundry.

Each data and address frame in the FPGA begins with a frame start pair of bits and ends with eight stop bits set to 1. If any of the previous stop bits were a 0 when a frame start pair is encountered, it is flagged as a frame alignment error.

Error checking is also done on the FPGA for each frame by means of a checksum byte. If an error is found on evaluation of the checksum byte, then a checksum/parity error is flagged. The checksum is the XOR of all the data bytes, from the start of frame up to and including the bytes before the checksum. It applies to the ID, address, and data frames.

When any of the three possible errors occur, the FPGA is forced into an idle state, forcing $\overline{\text{INIT}}$ low. The FPGA will remain in this state until either the $\overline{\text{RESET}}$ or $\overline{\text{PRGM}}$ pins are asserted.

If using either of the MPI modes to configure the FPGA, the specific type of bit stream error is written to one of the MPI registers by the FPGA configuration logic. The $\overline{\text{PGRM}}$ bit of the MPI control register can also be used to reset out of the error condition and restart configuration.

FPGA Configuration Modes

There are eight methods for configuring the FPGA. Seven of the configuration modes are selected on the M0, M1, and M2 inputs. The eighth configuration mode is accessed through the boundary-scan interface. A fourth input, M3, is used to select the frequency of the internal oscillator, which is the source for CCLK in some configuration modes. The nominal frequencies of the internal oscillator are 1.25 MHz and 10 MHz. The 1.25 MHz frequency is selected when the M3 input is unconnected or driven to a high state.

There are three basic FPGA configuration modes: master, slave, and peripheral. The configuration data can be transmitted to the FPGA serially or in parallel bytes. As a master, the FPGA provides the control signals out to strobe data in. As a slave device, a clock is generated externally and provided into the CCLK input. In the three peripheral modes, the FPGA acts as a microprocessor peripheral. Table 34 lists the functions of the configuration mode pins. Note that two configuration modes previously available on the OR2Cxx and OR2C/TxxA devices (master parallel down and synchronous peripheral) have been removed for Series 3 devices.

Table 34. Configuration Modes

M2	M1	M0	CCLK	Configuration Mode	Data
0	0	0	Output	Master Serial	Serial
0	0	1	Input	Slave Parallel	Parallel
0	1	0	Output	Microprocessor: *Motorola** *PowerPC*	Parallel
0	1	1	Output	Microprocessor: *Intel i960*	Parallel
1	0	0	Output	Master Parallel	Parallel
1	0	1	Output	Async Peripheral	Parallel
1	1	0	Reserved		
1	1	1	Input	Slave Serial	Serial

* *Motorola* is a registered trademark of Motorola, Inc.

Master Parallel Mode

The master parallel configuration mode is generally used to interface to industry-standard, byte-wide memory, such as the 2764 and larger EPROMs. Figure 54 provides the connections for master parallel mode. The FPGA outputs an 18-bit address on A[17:0] to memory and reads 1 byte of configuration data on the rising edge of RCLK. The parallel bytes are internally serialized starting with the least significant bit, D0. D[7:0] of the FPGA can be connected to D[7:0] of the microprocessor only if a standard prom file format is used. If a .bit or .rbt file is used from *ORCA* Foundry, then the user must mirror the bytes in the .bit or .rbt file OR leave the .bit or .rbt file unchanged and connect D[7:0] of the FPGA to D[0:7] of the microprocessor.

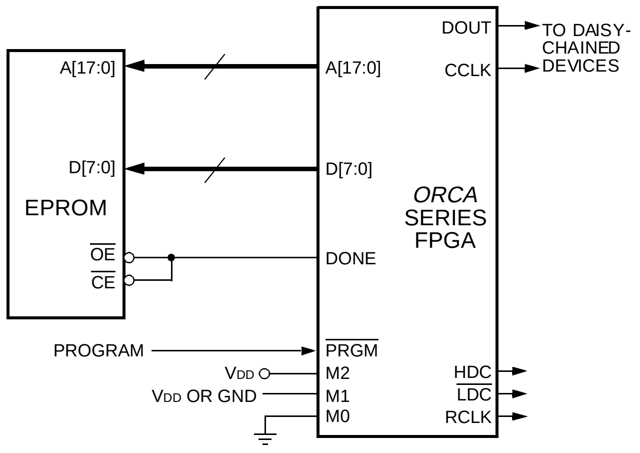


Figure 54. Master Parallel Configuration Schematic

In master parallel mode, the starting memory address is 00000 Hex, and the FPGA increments the address for each byte loaded.

One master mode FPGA can interface to the memory and provide configuration data on DOUT to additional FPGAs in a daisy-chain. The configuration data on DOUT is provided synchronously with the falling edge of CCLK. The frequency of the CCLK output is eight times that of RCLK.

FPGA Configuration Modes (continued)

Master Serial Mode

In the master serial mode, the FPGA loads the configuration data from an external serial ROM. The configuration data is either loaded automatically at start-up or on a $\overline{\text{PRGM}}$ command to reconfigure. The ATT1700A Series Serial PROMs can be used to configure the FPGA in the master serial mode. This provides a simple 4-pin interface in a compact package.

Configuration in the master serial mode can be done at powerup and/or upon a configure command. The system or the FPGA must activate the serial ROM's $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ inputs. At powerup, the FPGA and serial ROM each contain internal power-on reset circuitry that allows the FPGA to be configured without the system providing an external signal. The power-on reset circuitry causes the serial ROM's internal address pointer to be reset. After powerup, the FPGA automatically enters its initialization phase.

The serial ROM/FPGA interface used depends on such factors as the availability of a system reset pulse, availability of an intelligent host to generate a configure command, whether a single serial ROM is used or multiple serial ROMs are cascaded, whether the serial ROM contains a single or multiple configuration programs, etc. Because of differing system requirements and capabilities, a single FPGA/serial ROM interface is generally not appropriate for all applications.

Data is read in the FPGA sequentially from the serial ROM. The DATA output from the serial ROM is connected directly into the DIN input of the FPGA. The CCLK output from the FPGA is connected to the CLK input of the serial ROM. During the configuration process, CCLK clocks one data bit on each rising edge.

Since the data and clock are direct connects, the FPGA/serial ROM design task is to use the system or FPGA to enable the $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ of the serial ROM(s). There are several methods for enabling the serial ROM's $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ inputs. The serial ROM's $\overline{\text{RESET}}$/OE is programmable to function with RESET active-high and $\overline{\text{OE}}$ active-low or $\overline{\text{RESET}}$ active-low and OE active-high.

In Figure 55, serial ROMs are cascaded to configure multiple daisy-chained FPGAs. The host generates a 500 ns low pulse into the FPGA's $\overline{\text{PRGM}}$ input. The FPGA's $\overline{\text{INIT}}$ input is connected to the serial ROMs' $\overline{\text{RESET}}$/OE input, which has been programmed to function with $\overline{\text{RESET}}$ active-low and OE active-high. The FPGA DONE is routed to the $\overline{\text{CE}}$ pin. The low on DONE enables the serial ROMs. At the completion of configuration, the high on the FPGA's DONE disables the serial ROM.

Serial ROMs can also be cascaded to support the configuration of multiple FPGAs or to load a single FPGA when configuration data requirements exceed the capacity of a single serial ROM. After the last bit from the first serial ROM is read, the serial ROM outputs $\overline{\text{CEO}}$ low and 3-states the DATA output. The next serial ROM recognizes the low on $\overline{\text{CE}}$ input and outputs configuration data on the DATA output. After configuration is complete, the FPGA's DONE output into $\overline{\text{CE}}$ disables the serial ROMs.

This FPGA/serial ROM interface is not used in applications in which a serial ROM stores multiple configuration programs. In these applications, the next configuration program to be loaded is stored at the ROM location that follows the last address for the previous configuration program. The reason the interface in Figure 55 will not work in this application is that the low output on the $\overline{\text{INIT}}$ signal would reset the serial ROM address pointer, causing the first configuration to be reloaded.

In some applications, there can be contention on the FPGA's DIN pin. During configuration, DIN receives configuration data, and after configuration, it is a user I/O. If there is contention, an early DONE at start-up (selected in *ORCA* Foundry) may correct the problem. An alternative is to use $\overline{\text{LDC}}$ to drive the serial ROM's $\overline{\text{CE}}$ pin. In order to reduce noise, it is generally better to run the master serial configuration at 1.25 MHz (M3 pin tied high), rather than 10 MHz, if possible.

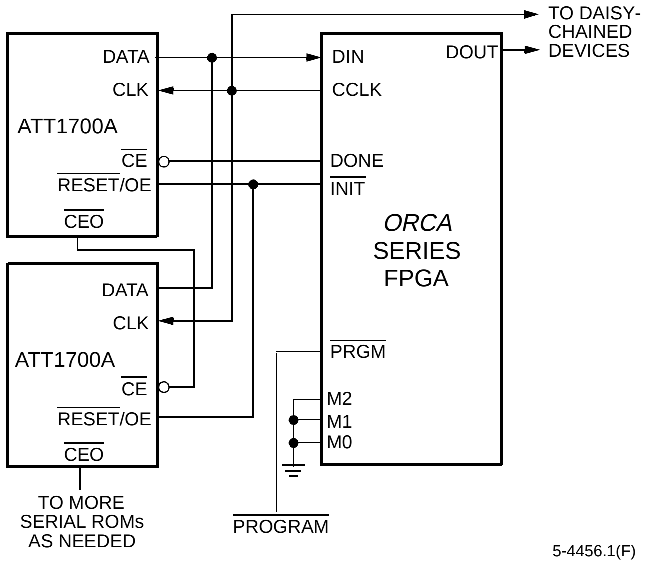


Figure 55. Master Serial Configuration Schematic

FPGA Configuration Modes (continued)

Asynchronous Peripheral Mode

Figure 56 shows the connections needed for the asynchronous peripheral mode. In this mode, the FPGA system interface is similar to that of a microprocessor-peripheral interface. The microprocessor generates the control signals to write an 8-bit byte into the FPGA. The FPGA control inputs include active-low $\overline{\text{CS0}}$ and active-high CS1 chip selects and $\overline{\text{WR}}$ and $\overline{\text{RD}}$ inputs. The chip selects can be cycled or maintained at a static level during the configuration cycle. Each byte of data is written into the FPGA's D[7:0] input pins. D[7:0] of the FPGA can be connected to D[7:0] of the microprocessor only if a standard prom file format is used. If a .bit or .rbt file is used from *ORCA* Foundry, then the user must mirror the bytes in the .bit or .rbt file OR leave the .bit or .rbt file unchanged and connect D[7:0] of the FPGA to D[0:7] of the microprocessor.

The FPGA provides an RDY/$\overline{\text{BUSY}}$ status output to indicate that another byte can be loaded. A low on RDY/$\overline{\text{BUSY}}$ indicates that the double-buffered hold/shift registers are not ready to receive data, and this pin must be monitored to go high before another byte of data can be written. The shortest time RDY/$\overline{\text{BUSY}}$ is low occurs when a byte is loaded into the hold register and the shift register is empty, in which case the byte is immediately transferred to the shift register. The longest time for RDY/$\overline{\text{BUSY}}$ to remain low occurs when a byte is loaded into the holding register and the shift register has just started shifting configuration data into configuration RAM.

The RDY/$\overline{\text{BUSY}}$ status is also available on the D7 pin by enabling the chip selects, setting $\overline{\text{WR}}$ high, and applying $\overline{\text{RD}}$ low, where the $\overline{\text{RD}}$ input provides an output enable for the D7 pin when $\overline{\text{RD}}$ is low. The D[6:0] pins are not enabled to drive when $\overline{\text{RD}}$ is low and, therefore, only act as input pins in asynchronous peripheral mode. Optionally, the user can ignore the RDY/$\overline{\text{BUSY}}$ status and simply wait until the maximum time it would take for the RDY/$\overline{\text{BUSY}}$ line to go high, indicating the FPGA is ready for more data, before writing the next data byte.

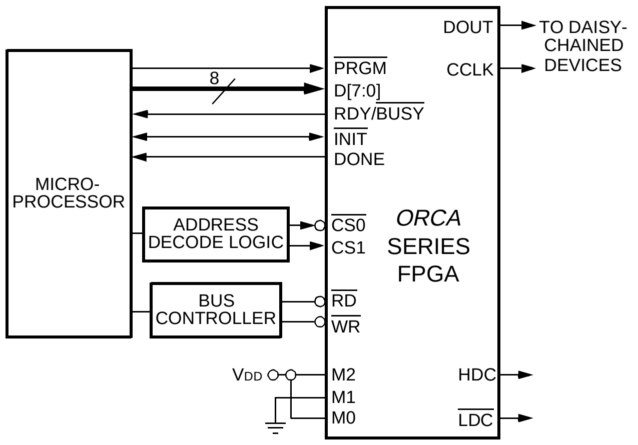


Figure 56. Asynchronous Peripheral Configuration

Microprocessor Interface (MPI) Mode

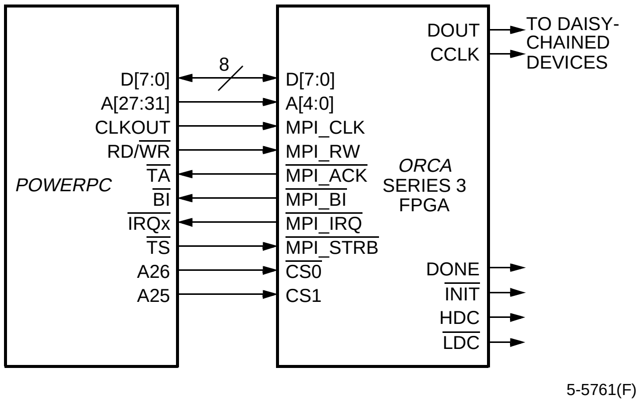
The built-in MPI in Series 3 FPGAs is designed for use in configuring the FPGA. Figure 57 and Figure 58 show the glueless interface for FPGA configuration and readback from the *PowerPC* and *i960* processors, respectively. When enabled by the mode pins, the MPI handles all configuration/readback control and handshaking with the host processor. For single FPGA configuration, the host sets the configuration control register $\overline{\text{PRGM}}$ bit to zero then back to a one and, after reading that the $\overline{\text{INIT}}$ signal is high in the MPI status register, transfers data 8 bits at a time to the FPGA's D[7:0] input pins.

If configuring multiple FPGAs through daisy-chain operation is desired, the MP_DAISY bit must be set in the configuration control register of the MPI. Because of the latency involved in a daisy-chain configuration, the MP_HOLD_BUS bit may be set to zero rather than one for daisy-chain operation. This allows the MPI to acknowledge the data transfer before the configuration information has been serialized and transferred on the FPGA daisy-chain. The early acknowledgment frees the host processor to perform other system tasks. Configuring with the MP_HOLD_BUS bit at zero requires that the host microprocessor poll the RDY/$\overline{\text{BUSY}}$ bit of the MPI status register and/or use the MPI interrupt capability to confirm the readiness of the MPI for more configuration data.

FPGA Configuration Modes (continued)

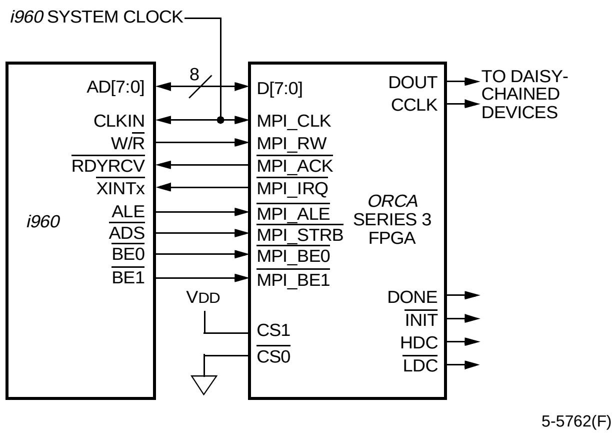
There are two options for using the host interrupt request in configuration mode. The configuration control register offers control bits to enable the interrupt on either a bit stream error or to notify the host processor when the FPGA is ready for more configuration data. The MPI status register may be used in conjunction with, or in place of, the interrupt request options. The status register contains a 2-bit field to indicate the bit stream error status. As previously mentioned, there is also a bit to indicate the MPI's readiness to receive another byte of configuration data. A flow chart of the MPI configuration process is shown in Figure 59. The MPI status and configuration register bit maps can be found in the Special Function Blocks section and MPI configuration timing information is available in the Timing Characteristics section of this data sheet.

Configuration readback can also be performed via the MPI when it is in user mode. The MPI is enabled in user mode by setting the MP_USER bit to 1 in the configuration control register prior to the start of configuration or through a configuration option. To perform readback, the host processor writes the 14-bit readback start address to the readback address registers and sets the $\overline{\text{RD\_CFG}}$ bit to 0 in the configuration control register. Readback data is returned 8 bits at a time to the readback data register and is valid when the DATA_RDY bit of the status register is 1. There is no error checking during readback. A flow chart of the MPI readback operation is shown in Figure 60. The RD_DATA pin used for dedicated FPGA readback is invalid during MPI readback.



Note: FPGA shown as a memory-mapped peripheral using $\overline{\text{CS0}}$ and CS1. Other decoding schemes are possible using $\overline{\text{CS0}}$ and/or CS1.

Figure 57. *PowerPC*/MPI Configuration Schematic



Note: FPGA shown as only system peripheral with fixed chip select signals. For multiperipheral systems, address decoding and/or latching can be used to implement chip selects.

Figure 58. *i960*/MPI Configuration Schematic

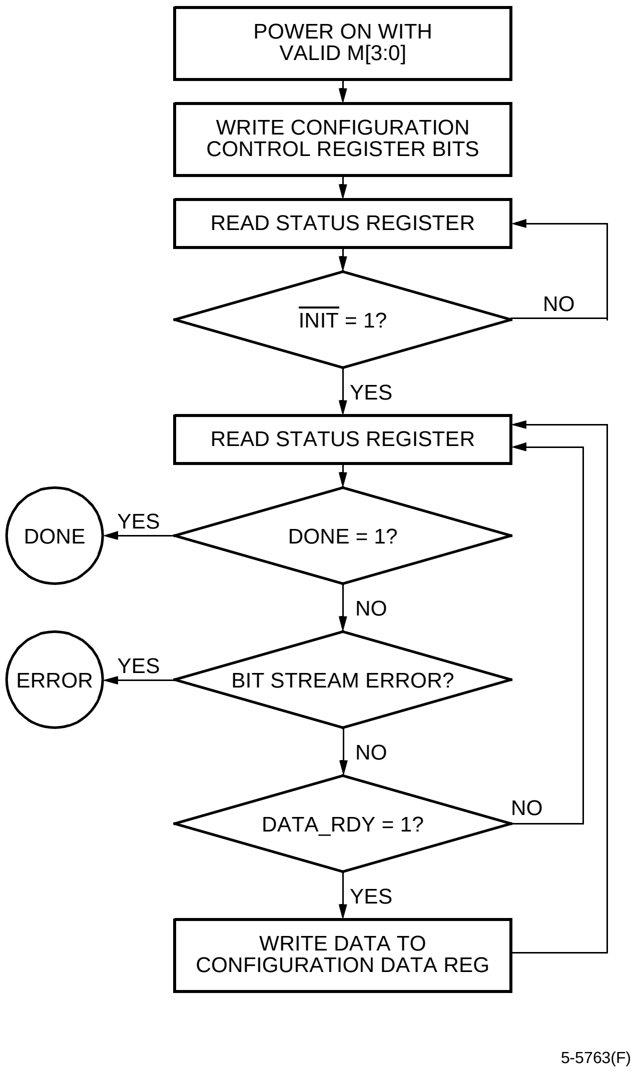
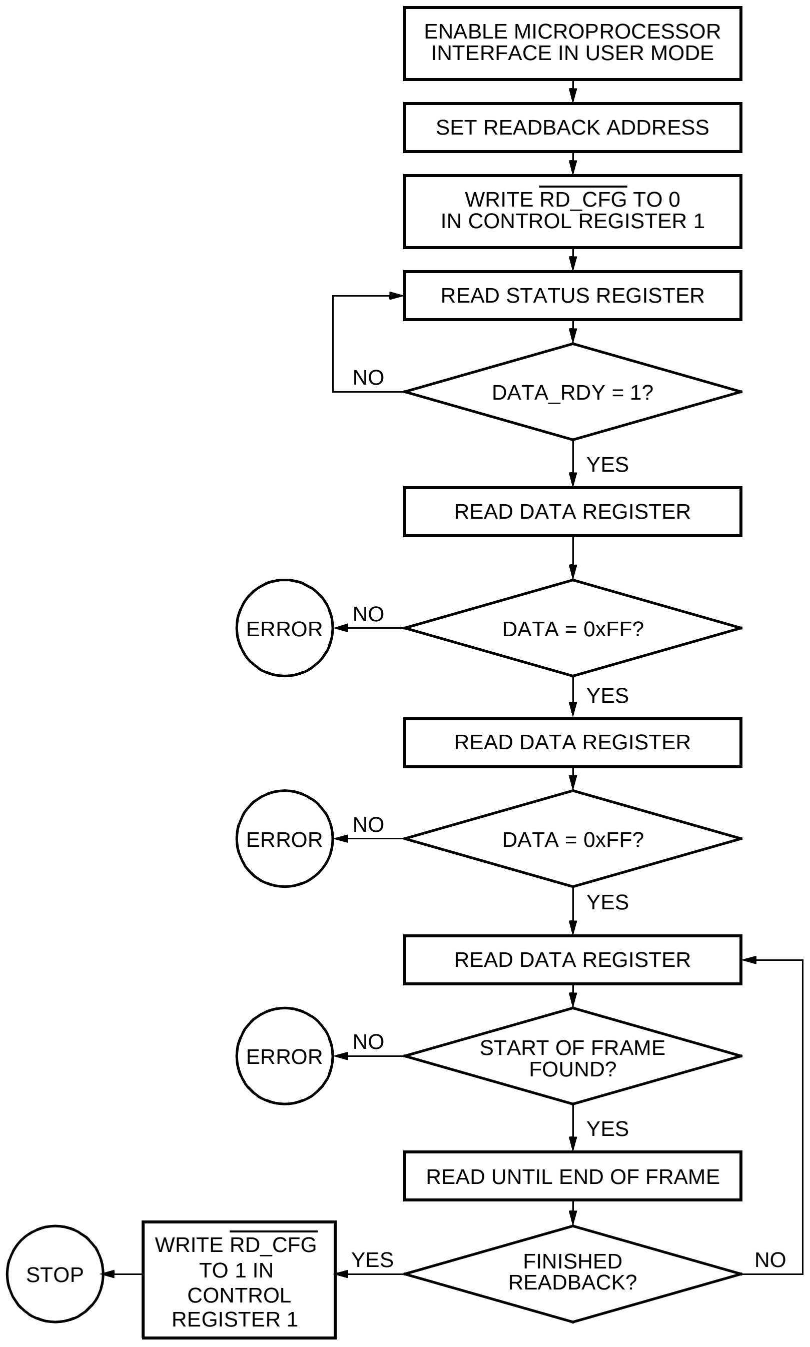


Figure 59. Configuration Through MPI

FPGA Configuration Modes (continued)



5-5764(F)

Figure 60. Readback Through MPI

FPGA Configuration Modes (continued)

Slave Serial Mode

The slave serial mode is primarily used when multiple FPGAs are configured in a daisy-chain (see the Daisy-Chaining section). It is also used on the FPGA evaluation board that interfaces to the download cable. A device in the slave serial mode can be used as the lead device in a daisy-chain. Figure 61 shows the connections for the slave serial configuration mode.

The configuration data is provided into the FPGA's DIN input synchronous with the configuration clock CCLK input. After the FPGA has loaded its configuration data, it retransmits the incoming configuration data on DOUT. CCLK is routed into all slave serial mode devices in parallel.

Multiple slave FPGAs can be loaded with identical configurations simultaneously. This is done by loading the configuration data into the DIN inputs in parallel.

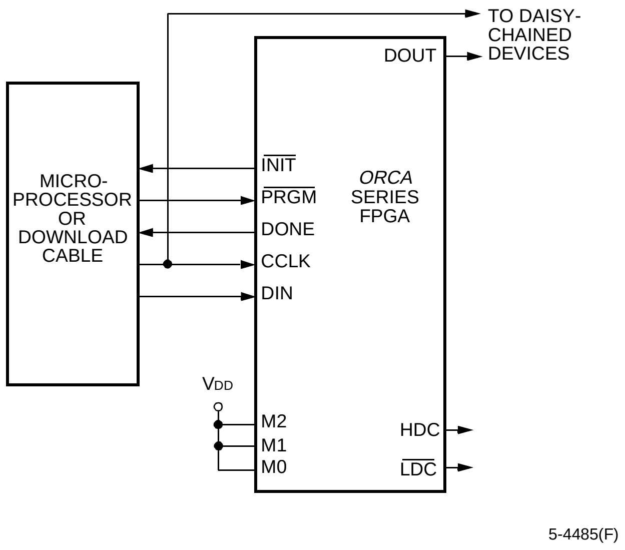


Figure 61. Slave Serial Configuration Schematic

Slave Parallel Mode

The slave parallel mode is essentially the same as the slave serial mode except that 8 bits of data are input on pins D[7:0] for each CCLK cycle. Due to 8 bits of data being input per CCLK cycle, the DOUT pin does not contain a valid bit stream for slave parallel mode. As a result, the lead device cannot be used in the slave parallel mode in a daisy-chain configuration.

Figure 62 is a schematic of the connections for the slave parallel configuration mode. $\overline{\text{WR}}$ and $\overline{\text{CS0}}$ are active-low chip select signals, and CS1 is an active-high chip select signal. These chip selects allow the user to configure multiple FPGAs in slave parallel mode using an 8-bit data bus common to all of the FPGAs. These chip selects can then be used to select the FPGA(s) to be configured with a given bit stream. The chip selects must be active for each valid CCLK cycle until the device has been completely programmed. They can be inactive between cycles but must meet the setup and hold times for each valid positive CCLK. D[7:0] of the FPGA can be connected to D[7:0] of the microprocessor only if a standard prom file format is used. If a .bit or .rbt file is used from *ORCA* Foundry, then the user must mirror the bytes in the .bit or .rbt file OR leave the .bit or .rbt file unchanged and connect D[7:0] of the FPGA to D[0:7] of the microprocessor.

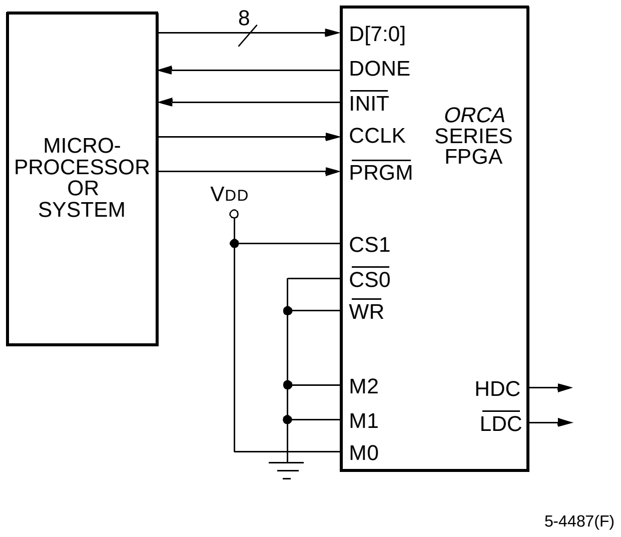


Figure 62. Slave Parallel Configuration Schematic

FPGA Configuration Modes (continued)

Daisy-Chaining

Multiple FPGAs can be configured by using a daisy-chain of the FPGAs. Daisy-chaining uses a lead FPGA and one or more FPGAs configured in slave serial mode. The lead FPGA can be configured in any mode except slave parallel mode. (Daisy-chaining is available with the boundary-scan ram_w instruction discussed later.)

All daisy-chained FPGAs are connected in series. Each FPGA reads and shifts the preamble and length count in on positive CCLK and out on negative CCLK edges.

An upstream FPGA that has received the preamble and length count outputs a high on DOUT until it has received the appropriate number of data frames so that downstream FPGAs do not receive frame start bit pairs. After loading and retransmitting the preamble and length count to a daisy-chain of slave devices, the lead device loads its configuration data frames.

The loading of configuration data continues after the lead device has received its configuration data if its internal frame bit counter has not reached the length count. When the configuration RAM is full and the number of bits received is less than the length count field, the FPGA shifts any additional data out on DOUT.

The configuration data is read into DIN of slave devices on the positive edge of CCLK, and shifted out DOUT on the negative edge of CCLK. Figure 63 shows the connections for loading multiple FPGAs in a daisy-chain configuration.

The generation of CCLK for the daisy-chained devices that are in slave serial mode differs depending on the configuration mode of the lead device. A master parallel mode device uses its internal timing generator to produce an internal CCLK at eight times its memory address rate (RCLK). The asynchronous peripheral mode device outputs eight CCLKs for each write cycle. If the lead device is configured in slave mode, CCLK must be routed to the lead device and to all of the daisy-chained devices.

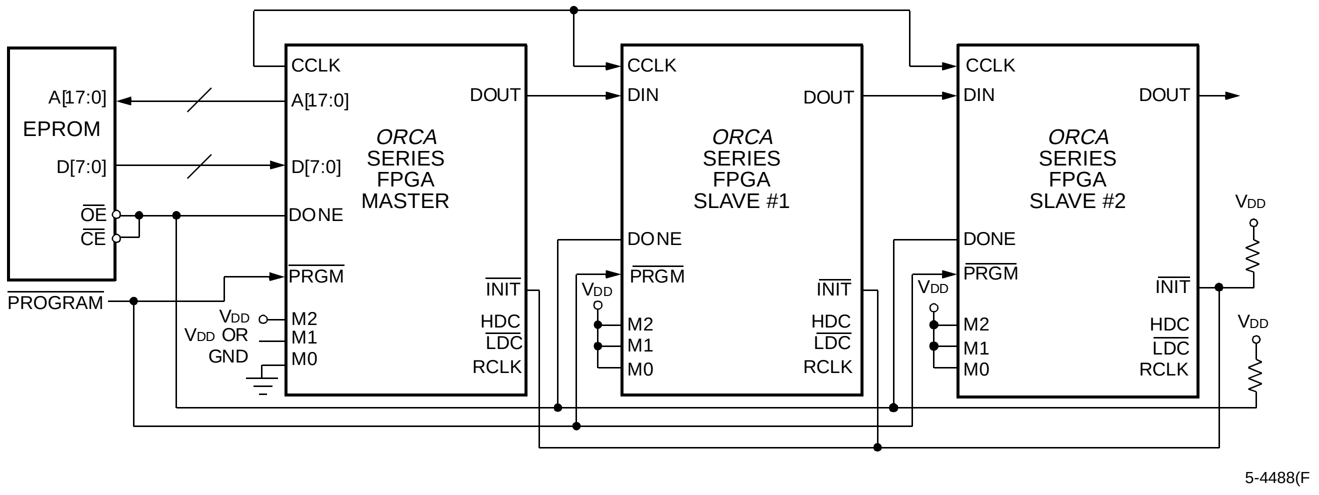


Figure 63. Daisy-Chain Configuration Schematic

As seen in Figure 63, the $\overline{\text{INIT}}$ pins for all of the FPGAs are connected together. This is required to guarantee that powerup and initialization will work correctly. In general, the DONE pins for all of the FPGAs are also connected together as shown to guarantee that all of the FPGAs enter the start-up state simultaneously. This may not be required, depending upon the start-up sequence desired.

FPGA Configuration Modes (continued)

Daisy-Chaining with Boundary Scan

Multiple FPGAs can be configured through the JTAG ports by using a daisy-chain of the FPGAs. This daisy-chaining operation is available upon initial configuration after powerup, after a power-on reset, after pulling the program pin to reset the chip, or during a reconfiguration if the EN_JTAG RAM has been set.

All daisy-chained FPGAs are connected in series. Each FPGA reads and shifts the preamble and length count in on the positive TCK and out on the negative TCK edges.

An upstream FPGA that has received the preamble and length count outputs a high on TDO until it has received the appropriate number of data frames so that downstream FPGAs do not receive frame start bit pairs. After loading and retransmitting the preamble and length count to a daisy-chain of downstream devices, the lead device loads its configuration data frames.

The loading of configuration data continues after the lead device had received its configuration read into TDI of downstream devices on the positive edge of TCK, and shifted out TDO on the negative edge of TCK. Figure 63 shows the connections for loading multiple FPGAs in a JTAG daisy-chain configuration.