XILINX®

XAPP452 (v1.1) June 25, 2008

Spartan-3 FPGA Family Advanced Configuration Architecture

Summary

This application note provides a detailed description of the Spartan®-3 FPGA family configuration architecture. It explains the composition of the bitstream file and how this bitstream is interpreted by the configuration logic to program the part. Additionally, a methodology is presented that will guide the user through the readback process. Although the other members of the Spartan-3 generation families are similar, this information is specific to the original Spartan-3 family.

Introduction

CLBs, IOBs, and Configuration Architecture

Spartan-3 devices, like all FPGAs, have both non-programmable and programmable areas. The non-programmable areas include the configuration logic, the Boundary-Scan logic, and other components. The programmable areas include portions of the input/output blocks (IOBs), digital clock managers (DCMs), configurable logic blocks (CLBs), the initial content of the block RAMs, and the interconnect routing between clock resources, logic resources, and I/Os.

Since Spartan-3 device configuration is based on CMOS Configuration Latches, it is volatile and must be configured upon power-up. Configuration is required to define the LUT equations, signal routing, flip-flop reset polarity, IOB voltage standards, and all other aspects of the user design. This configuration information, along with the instructions for the configuration logic, is combined to form a bitstream.

Configuration Modes

Spartan-3 devices can be configured through the SelectMAP (Slave Parallel) interface, serial interface, or through the Boundary-Scan (JTAG) interface. The configuration mode must be specified by setting the appropriate logic levels on the Mode (M2, M1, M0) pins.

While the mechanics of the various configuration modes differ, the actual configuration method is transparent to the Spartan-3 FPGA configuration logic. The IOBs, CLBs, and all other user-configurable logic are configured in exactly the same way regardless of configuration mode. A bitstream for the SelectMAP interface can look exactly the same as a bitstream for the Serial interface.

The mechanics for delivering a bitstream to a Spartan-3 device in the different modes is explained in Module 2 of the Spartan-3 FPGA data sheet (DS099).

Reading Configuration Bits from a Spartan-3 Device (Readback)

After configuring a Spartan-3 device, it is possible to read back the configuration logic to verify that the configuration remains as the user intended, or to read design information or data from an operating device.

Configuration Readback may be performed through either SelectMAP mode or Boundary-Scan mode. Readback and all other Boundary-Scan operations are available regardless of which configuration mode the device is in. However, readback is not possible through the Serial interfaces.

Device Architecture

Overview

The purpose of device configuration is to configure the internal hardware and interconnects as specified by the user. The configuration memory, which specifies the function of the FPGA's logic resources, can be visualized as a rectangular array of bits. The information provided by the user is specified in a bitstream, which also includes instructions to the device detailing how to interpret the user information. The information provided in the bitstream is more than just initialization data for the internal memory cells – it contains instructions on how the information configures the device.

Spartan-3 Device Configuration Registers

Spartan-3 devices have a number of internal registers that control configuration and readback (see Table 1). This section describes each of those registers in detail.

Table 1: **Device Registers**

Name	Mnemonic	Read/Write	Binary Address
Cyclic Redundancy Check	CRC	R/W	00000
Frame Address Register	FAR	R/W	00001
Frame Data Input Register	FDRI	W	00010
Frame Data Output Register	FDRO	R	00011
Command Register	CMD	R/W	00100
Control Register	CTL	R/W	00101
Mask Register	MASK	R/W	00110
Status Register	STAT	R	00111
Legacy Output Register	LOUT	W	01000
Configuration Options Register	COR	R/W	01001
Multiple Frame Write Register	MFWR	W	01010
Frame Length Register	FLR	R/W	01011
(Reserved)	–	–	01100
(Reserved)	–	–	01101
Product IDCODE Register	IDCODE	R/W	01110

Cyclic Redundancy Check Register

Data input integrity is provided through the Cyclic Redundancy Check (CRC) register. When data is written to any configuration register except the LOUT register, a 16-bit CRC value is calculated using the register data and address. The resulting value is saved in the CRC register. At the end of the Frame Data Input (FDRI) Register, the last 32-bit word is automatically interpreted as a CRC value (known as *Auto CRC*). That value is checked against the current value of the CRC register. A CRC check may also be done explicitly by writing a precalculated value into the CRC register. If the result is non-zero, an error is indicated. The CRC_ERROR bit is accessible through the Status Register. If a CRC error is detected during configuration, the configuration logic is put in the ERROR mode (signaled by the assertion of INIT_B).

CRC checking can be disabled through the BitGen `-g CRC:Disable` option or by setting the COR register's CRC_BYPASS bit. If CRC is disabled, the default CRC value of 0x0000DEFC must be written to the CRC register in place of the calculated CRC value.

The Spartan-3 FPGA configuration uses a standard 16-bit CRC checksum algorithm. The 16-bit CRC polynomial is shown below:

$$\text{CRC-16} = X^{16} + X^{15} + X^{2} + 1$$

The algorithm is implemented by shifting the data stream into a 16-bit shift register, shown in Figure 1. Register Bit 0 receives an XOR of the incoming data with the output of Bit 15. Bit 2 receives an XOR of the input to Bit 0 with the output of Bit 1. Bit 15 receives an XOR of the input to Bit 0 with the output of Bit 14.

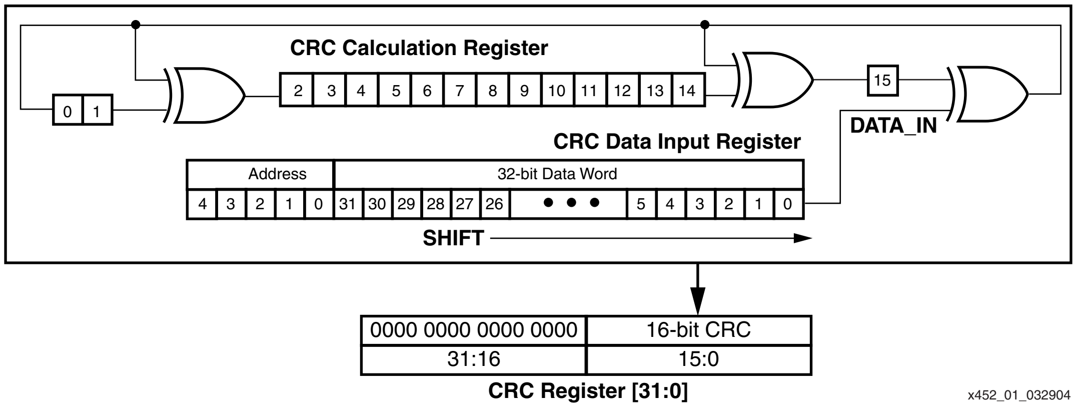

Figure 1: **Serial 16-bit CRC**

Frame Address Register

The Frame Address (FAR) Register holds the address of the current frame. As shown in Figure 2, the frame address consists of three parts: the Column Address, the Major Address, and the Minor Address. The command in the Command (CMD) Register is executed each time the FAR register is loaded with a new value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Column Address			Major Address (Column Position)								Minor Address (Frame Address)								FRM_BYTE(1)								
0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0

1. FRM_BYTE is not used.

Figure 2: **Frame Address Register**

Frame Data Input Register

The Frame Data Input (FDRI) Register loads configuration frame data into Spartan-3 devices. The FDRI register is a shift register into which data is loaded prior to transfer to the configuration memory. To write configuration data to the device, the CMD register is loaded with the Write Configuration Data (WCFG) command and then the FDRI register is loaded with at least two frames of 32-bit words.

The write operation is pipelined such that the first frame of data is written to the configuration memory while the second frame is being shifted in. The last configuration frame is always a pad frame. This pad frame is dummy data, which is not actually written to the configuration memory. Each frame write must include enough 32-bit data words to load the entire frame.

Frame Data Output Register

The Frame Data Output (FDRO) Register is used for reading or capturing configuration data from the Spartan-3 device via readback. CRC is not calculated based on FDRO data. CRC data also is not read back. To perform Readback, the CMD register is loaded with the Read Configuration Data (RCFG) command, then the correct number of words is read out of the FDRO register.

Command Register

The configuration state machine interprets the contents of the Command (CMD) Register. Configuration commands control the operation of the configuration state machine, the Frame Data Registers (FDRI and FDRO) and some of the global signals. The command in the CMD register is executed each time the FAR register is loaded with a new value. Table 2 lists all valid commands.

Table 2: **CMD Codes**

Command	Code	Description
NULL	0000	No Operation
WCFG	0001	Write Configuration Data – Used prior to writing configuration data to the FDRI register.
MFWR	0010	Multiple Frame Write – Performs a write of a single frame data to multiple frame addresses.
DGHIGH/LFRM	0011	Last Frame Write – GHIGH_B is deasserted during this time. This command also can be used for shutdown reconfiguration.
RCFG	0100	Read Configuration Data – Used prior to reading configuration data from the FDRO register.
START	0101	Begin Startup Sequence – Starts the startup sequence, which completes configuration when finished. The startup sequence begins after a successful CRC check and a DESYNC command is performed.
RCAP	0110	Reset Capture – Used when performing capture in single-shot mode. This command must be used to reset the capture signal if signal-shot capture has been selected.
RCRC	0111	Reset CRC – Used to reset the CRC register.
AGHIGH	1000	Assert GHIGH_B Signal – Used prior to reconfiguration to prevent contention while writing new configuration data. This command is only used in non-active reconfiguration.
SWITCH	1001	Switch CCLK Frequency – Changes the frequency of the master CCLK. The frequencies are listed in Table 6.
GRESTORE	1010	Pulse the GRESTORE Signal – Used to set/reset the internal IOB and CLB flip-flops.
SHUTDOWN	1011	Begin Shutdown Sequence – Starts the shutdown sequence and disables the device when finished. Shutdown is performed on the next successful CRC check or RCRC instruction.
GCAPTURE	1100	Pulse GCAPTURE – Causes the capture cells within the device to be loaded.
DESYNCH	1101	Reset DALIGN Signal – Used at the end of configuration to desynchronize the device.

Control Register

The Control (CTL) Register selects aspects of the configuration circuitry and logic. Figure 3 shows its fields, and Table 3 defines them.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																										SBITS		PERSIST			GTS_USR_B
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	0	0	0	1	0	x

Figure 3: **Control Register**

Table 3: **Control Register Bits**

Name	Bit Indices	Description
SBITS	5:4	Security Level: 00: Read/Write OK (default) 01: Readback disabled 1x: Readback disabled, writing disabled except to CRC, CMD, and TEST registers.
Persist	3	Determines if the configuration interface defined by M2:M0 remains after configuration 0: No (default) 1: Yes
GTS_USER_B	0	Active-Low global 3-state of I/Os. Turns off pull-up resistors if GTS_CFG_B is also asserted. 0: Yes 1: No (Default)

Mask Register

The Mask (MASK) Register controls write permission of the CTL register. A 1 in bit N of the MASK register allows the bit position to be written in the CTL register. The default value of the MASK register is 0.

Status Register

The Status (STAT) Register is loaded with current values of several control or status signals. This register can be read back to provide the current configuration state of the device. Figure 4 shows the fields in the STAT register, and Table 4 defines them.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																		ID_ERROR	DONE	INIT		MODE		GHIGH_B	GWE	GTS_CFG	IN_ERROR	DCI_MATCH	DCM_LOCK	(RESERVED)	CRC_ERROR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4: **STAT Register**

Table 4: **STAT Values**

Name	Bit Indices	Description
ID_ERROR	13	IDCODE not validated while trying to write the FDRI register
DONE	12	Input from the DONE pin
INIT	11	Input from the INIT pin
MODE	10:8	Input from the MODE pins (M2:M0)
GHIGH_B	7	Status of GHIGH_B (0 = asserted)
GWE	6	Status of GWE (0 = all FFs and Block RAMs are write-disabled)
GTS_CFG	5	Status of GTS_CFG_B (0 = all I/Os are 3-stated)
IN_ERROR	4	Legacy input error. This error occurs when serial data is loaded too fast.
DCI_MATCH	3	DCI is matched. This bit is a logical AND function of all the MATCH signals (one per bank). If no DCI I/Os are in a particular bank, then a 1 is used.
DCM_LOCK	2	DCMs are locked. This bit is a logical AND function of all the LOCKED signals. If DCM is not used, then a 1 is used.
CRC_ERROR	0	CRC error

Legacy Output Register

The Legacy Output (LOUT) Register is used for daisy chaining the configuration bitstream to other Xilinx devices. Data written to the LOUT register is serialized and appears on the DOUT pin. The LOUT command should not be used for SelectMAP or JTAG modes.

Configuration Options Register

The Configuration Options (COR) Register selects configuration options for the device. Figure 5 shows these options, and Table 5 defines them.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		CRC_BYPASS				DONE_PIPE	DRIVE_DONE	SINGLE	OSCFSEL						SSCLKSRC		DONE_CYCLE			MATCH_CYCLE			LOCK_CYCLE			GTS_CYCLE			GWE_CYCLE		
0	0	x	0	0	x	x	x	x	x	x	x	x	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 5: **Configuration Options Register**

Table 5: **COR Values**

Name	Bit Indices	Description
CRC_BYPASS	29	0: CRC used (Default) 1: Does not check against updated CRC value
DONE_PIPE	25	0: No pipeline stage for DONEIN (Default) 1: Add pipeline stage for DONEIN. The FPGA waits for DONE, which is delayed by one StartupClk cycle. Use this option when StartupClk is running at high speeds
DRIVE_DONE	24	0: DONE pin is open drain (Default) 1: DONE is actively driven High
SINGLE	23	0: Readback is not one-shot. Newly captured values are loaded on each successive CAP assertion on the CAPTURE_SPARTAN3 primitive. Capture can also be performed with the GCAPTURE instruction in the CMD register. (Default) 1: Readback is one-shot. The RCAP instruction must be loaded into the CMD register between successive readbacks.
OSCFSEL	22:19	Select CCLK frequency in Master configuration modes (see Table 6)
SSCLKSRC	16:15	Startup sequence clock source: 00: CCLK (Default) 01: UserClk (connection on the STARTUP_SPARTAN3 block) 1x: JTAGClk
DONE_CYCLE	14:12	Startup phase in which the DONE pin is released
MATCH_CYCLE	11:9	Stall in this startup phase until the DCI is matched
LOCK_CYCLE	8:6	Stall in this startup phase until DCMs are locked
GTS_CYCLE	5:3	Startup phase in which the Global 3-State (GTS) is deasserted
GWE_CYCLE	2:0	Startup phase in which the Global Write Enable (GWE) is asserted

Table 6: **OSCFSEL-Specified Master CCLK Settings**

CCLK (MHz)	OSCFSEL
3	X10x
6	X000
12	X001
25	X010
50	X011
100	X11x

Multiple Frame Write Register

The Multiple Frame Write (MFWR) Register is used with the BitGen `-g Compress` option. If more than one frame has identical data, it is possible to load that frame into the configuration logic, and instruct the logic to load the frame into multiple address locations. Depending on the utilization of the device, this may decrease the size of the bitstream considerably. This feature is only supported upon initial configuration. Therefore, to reconfigure the device with this feature, the part must be power-cycled or reset with PROG_B.

To write multiple frames with the same data, the following steps need to be performed:

1. Write the WCFG command to the CMD register.
2. Write a desired frame to the FDRI register.
3. Write the FAR register with the first desired address.
4. Write the MFWR command to the CMD register.
5. Write two dummy words to the MFWR register.
6. Write the FAR register with the second desired address.
7. Write two dummy words to the MFW register.
8. Repeat steps 6 and 7 until the last desired address is reached.

Frame Length Register

At the beginning of the configuration bitstream, the Frame Length (FLR) Register is written with the length of a frame, as measured in 32-bit words. This length counter provides sequencing information for the configuration read and write operations. The FLR register must be written before any FDR operation can be performed. It is not necessary to set the FLR register more than once. If the number of bits in a frame is not a multiple of 32, then the length counter of the frame must be rounded up to the next integer. Table 7 lists the values for the FLR register for all Spartan-3 devices.

Table 7: **Spartan-3 FPGA Configuration Data Frames and Programming Times**

Device	# of Frames	Frame Length in Bits	Configuration Bits	Total # of Bits (Including overhead)	Approx. SelectMap Download Time (50 MHz) in ms	Approx. Serial Download Time (50 MHz) in ms	Approx. JTAG Download Time (33 MHz) in ms
XC3S50	368	1,184	435,712	439,264	1.10	8.79	13.31
XC3S200	615	1,696	1,043,040	1,047,616	2.62	20.95	31.75
XC3S400	767	2,208	1,693,536	1,699,136	4.25	33.98	51.49
XC3S1000	995	3,232	3,215,840	3,223,488	8.06	64.67	97.68
XC3S1500	1223	4,256	5,205,088	5,214,784	13.04	104.30	158.02
XC3S2000	1451	5,280	7,661,280	7,673,024	19.18	153.46	232.52
XC3S4000	1793	6,304	11,303,072	11,316,864	28.29	226.34	342.94
XC3S5000	1945	6,816	13,257,120	13,271,936	33.18	265.44	402.18

When writing to the FLR register, the data is always specified in words. The actual value written to the FLR register is "Actual Frame Length – 1" because the device frame length starts with 0. For the XC3S50 part, the actual frame length in words is 37. However, when writing to the FLR register, 36 is specified to account for the frame length starting from 0.

IDCODE Register

The Product IDCODE (IDCODE) Register contains information to identify the device being accessed (see Table 8). When the IDCODE is written to the IDCODE register, the configuration logic checks the written data with an internal constant. If the values do not match, ID_ERROR is asserted. An IDCODE write is required before any frame data is written, providing a means to prevent a bitstream from mistargeting a part or identifying the part.

Table 8: **Spartan-3 Device IDCODEs**

Device	IDCODE
XC3S50	0x0140D093
XC3S200	0x01414093
XC3S400	0x0141C093
XC3S1000	0x11428093
XC3S1500	0x01434093
XC3S2000	0x01440093
XC3S4000	0x01448093
XC3S5000	0x01450093

Bitstream Composition

Packets

A Spartan-3 FPGA bitstream consists of a specific sequence of writes to the configuration registers. After synchronization, all data, register writes, and frame data are encapsulated in packets. There are two kinds of packets: Type 1 and Type 2. Type 1 packets are used for register writes. A combination of Type 1 and Type 2 packets is used for frame data writes.

A Type 1 packet consists of two parts: a header and the data. The header (see Figure 6) describes which register is being accessed, whether it is a read or write operation, and the size of the data to follow. The data portion, always immediately following the header, is the number of 32-bit words specified in the header.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type			Op		Register Address														RSVD		Word Count										
0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	x	x	x	x	x	x	x	x	x	x	x

Figure 6: **Type 1 Header**

As shown in Figure 6, only $2^{11} – 1$ 32-bit words (65,504 bits) can follow a single Type 1 header. In larger devices, there is a significant amount of overhead in a bitstream because a new packet needs to be sent for every 65,504 bits of data. To cut down on this overhead, Type 2 headers are used for large data writes. A Type 2 header (see Figure 7) is also a 32-bit word, but no destination is specified and it has a much larger word count field. A Type 2 header must immediately follow a Type 1 header with Word Count = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type			Op		Word Count																										
0	1	0	x	x	x	x	X	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 7: **Type 2 Header**

With a Type 1 – Type 2 combination, $2^{27} - 1$ 32-bit words (4,294,967,264 bits) can be sent in a single packet. This size is significantly larger than the largest Spartan-3 FPGA bitstream. Therefore all of the frame data can be sent in a single FDRI packet, allowing for very large serial daisy chains, because all daisy chain data must be contained in a single LOUT packet.

Below is an example of two packets, with a write to the CMD register Figure 8) followed by a large write to the FDRI register (Figure 9 and Figure 10).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type			Op		Register Address														RSVD		Word Count										
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 8: **CMD Header: 0x30008001**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type			Op		Register Address														RSVD		Word Count										
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9: **FDRI Type 1 Header: 0x30004000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type			Op		Word Count																										
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 10: **FDRI Type 2 Header (Write 2048 Dwords): 0x50000800**

Command Words

There are three main parts of a bitstream:

1. First, a Synchronization Word (0xAA995566) is loaded. The Spartan-3 FPGA configuration logic processes everything on 32-bit boundaries, and the Sync Word effectively indicates where those boundaries lie.
2. After the Sync Word, packets are processed to set up the hardware to begin writes to the FDRI register. The majority of a bitstream consists of a number of writes to the FAR and FDRI registers.
3. After all the frame data is loaded, subsequent writes to registers are required to perform error checking and to begin the startup sequence.

Table 9 lists the bitstream start and configuration options. Table 10 indicates the bitstream data frames. Table 11 provides the bitstream final commands and startup.

Table 9: **Bitstream Start and Configuration Options for XC3S400**

Data Description	Data Field
Dummy Word	0xFFFFFFFF(1)
Synchronization Word	0xAA995566
CMD Write Packet Header	0x30008001
CMD Write Packet Data (Reset CRC)	0x00000007
FLR Write Packet Header	0x30016001
FLR Write Packet Data	0x00000044(2)
COR Write Packet Header	0x30012001
COR Write Packet Data	0x00003FE5(2)
IDCODE Write Packet Header	0x3001C001
IDCODE Write Packet Data (3S400)	0x0141C093(2)
MASK Write Packet Header	0x3000C001
MASK Write Packet Data	0x00000000(2)
CMD Write Packet Header	0x30008001
CMD Write Packet Data (Switch CCLK)	0x00000009
FAR Write Packet Header	0x30002001
FAR Write Packet Data	0x00000000
CMD Write Packet Header	0x30008001
CMD Write Packet Data (WCFG)	0x00000001

Notes:

1. This value may be different for different software releases.
2. This value may be different based on device/design options.

Table 10: **Bitstream Data Frames**

Data Description	Data Field
FDRI Write Packet Header (Type 1)	0x30004000
FDRI Write Packet Header (Type 2)	0x5000CF00(1)
FDRI Write Packet Data (52992 Dwords)(2)	0x--------(1)
CMD Write Packet Header	0x30008001
CMD Write Packet Data (GRESTORE)	0x0000000A
CMD Write Packet Header	0x30008001
CMD Write Packet Data (DGHIGH/LFRM)	0x00000003
No Op (one frame's worth)	0x20000000

Notes:

1. This value may be different based on device/design options.
2. Includes one Auto CRC word.

Table 11: **Bitstream Final Commands and Start Up**

Data Description	Data Field
CMD Write Packet Header	0x30008001
CMD Write Packet Data (START)	0x00000005
CTL Write Packet Header	0x3000A001
CTL Write Packet Data	0x00000000(1)
CRC Write Packet Header	0x30000001
CRC Write Packet Data	0x--------(1)
CMD Write Packet Header	0x30008001
CMD Write Packet Data (DESYNC)	0x0000000D
No Op (4 words)	0x20000000

Notes:

1. This value may be different based on device/design options.

The following four operations must be performed successfully in order for the device to enter the startup sequence:

1. Write to the COR register to program the desired startup sequence.
2. Write the START command to the CMD register.
3. Write a CRC checksum to the CRC register.
4. Write the DESYNCH command to the CMD register.

Configuration Columns and Frames

The Spartan-3 FPGA configuration memory can be visualized as a rectangular array of bits. The bits are grouped into vertical frames that are one-bit wide and extend from the top of the array to the bottom. A frame is the atomic unit of configuration. It is the smallest portion of the configuration memory that can be written to or read from.

Frames are grouped into larger units called columns. Spartan-3 devices have different types of columns. The physical hardware that is configured by each column type is not strictly limited to the hardware implied by the column name. For example, certain IOBs are configured in a CLB column, along with the CLBs.

Frames do not directly map to any single piece of hardware. For instance, a single frame does not configure a single CLB or IOB, but actually configures a part of several logical resources, as well as some routing.

Table 12: **Spartan-3 FPGA Bitstream Column Types**

Column Type	# of Frames per Column	Number of Columns per Device	Column Address
TERM(L/R)	2	2	00
IOI (L/R)	19	2	00
CLB	19	# CLB columns	00
Block RAM	76	# Block RAM columns	01
Block RAM Interconnect	19	# Block RAM columns	10
GCLK	3	1	00

TERM Columns

The TERM columns provide configuration data pertaining to the I/O electrical standard. The TERM columns contain only those IOBs that are on the left and right edges of the device. The IOBs on the top and bottom edges of the device are configured along with the corresponding CLB Column.

IOI Columns

The IOI columns provide configuration information pertaining to the IOB registers, muxes, and 3-state buffers in the IOBs. As with the TERM column type, the IOI columns only contain configuration data for those IOBs on the left and right edges of the device. The IOBs on the top and bottom edges of the device are configured along with the corresponding CLB Column.

CLB Columns

The CLB column type includes the configurable logic blocks, all routing and interconnects (other than the global clock trees), and all IOBs on the top and bottom edges of the device.

Block RAM Columns

The Block RAM column type contains the Block RAM initialization data. The Block RAM initialization data consists of the initial values for the Block RAM as well as information on width, depth, and read/write enables.

Block RAM Interconnect Columns

The Block RAM interconnect column type includes only the Block RAM routing information.

GCLK Columns

The GCLK column contains DCM attributes and BUFG configuration.

Configuration Frame Addressing

Each frame has its own unique address indicating which part of the device it configures. The configuration commands that specifically load frame data includes the frame address as well as the data. For large frame writes, internal counters automatically increment the frame address starting with the Minor address, the Major address, and lastly the Block address.

Three different address pieces make up a frame's address: the Column Address, the Major Address, and the Minor Address. The Column Address indicates in broad strokes what kind of data is being loaded (see Table 13). Notice that all IOB and CLB frames share the same Column Address. Block RAM and Block RAM interconnect frames have their own Column Addresses.

The second piece of the address, the Major Address, indicates where (vertically) in the device the frame lies. For instance, SliceX0Y0 and Slice X8Y0 have different Major Addresses, but Slices X0Y0 and X0Y8 have the same Major Address. Each Column Address type has its own Major Address numbering scheme. For instance, Column Address 0 (CLB and IOB Type) Major Address 0 configures the center of the device controlling the global routing resources. Column Address 1 (Block RAM type) Major Address 0 configures the left-most Block RAM column.

The third piece of the address, the Minor Address, indicates where within the Major Address the frame lies. For instance, Column Address 1 (Block RAM type) Major Address 0, Minor Address 0 configures a specific piece of the left-most Block RAM column. A single frame corresponds to each Column Type-Major Address-Minor Address combination.

Table 13: **Frame Address Scheme**

Column	TERM_L	IOI_L	CLB	BRAM_INIT	BRAM	CLB	GCLK_L	CLB	CENTER	CLB	GCLK_R	CLB	BRAM_INIT	BRAM	CLB	IOI_R	TERM_R
Block	0	0	0	2	1	0	0	0	0	0	0	0	2	1	0	0	0
Major	1	2	3	0	0	4	0	5	0	6	0	7	1	1	8	9	10
Minor	0~1	0~18	0~18	0~18	0~75	0~18	0	0~18	2	0~18	1	0~18	0~18	0~75	0~18	0~18	0~1

Readback

Readback is the process of reading out all the data in the internal configuration memory. Readback can be used to verify the current configuration data, and read the current state of all internal CLB and IOB registers as well as the current LUT RAM and Block RAM values. Readback is only available through the SelectMAP and Boundary-Scan interfaces. This application note only demonstrates the use of the SelectMAP interface for performing readback.

Readback Verification and Capture

Readback verification verifies the validity of the stored configuration data. It is most commonly used in space-based applications, where exposure to radiation might alter the data stored in the configuration memory cells.

Readback capture lists the states of all the internal flip-flops, which can be used for hardware debugging and functional verification. When Capture is initiated, the internal register states are loaded into the configuration memory, replacing the initial register startup value. This data may be extracted after readback of the configuration memory.

Readback capture is not required for LUT RAM, Block RAM, or SRL16. The current contents of these memory elements are always read back.

While both Verify and Capture can be performed in one readback, each requires slightly different preparation and post-processing.

Preparing for Readback in Design Entry

If only readback verification is to be performed, there are no additional steps at the time of design entry. However, if readback capture is to be used, the library primitive called CAPTURE_SPARTAN3 must be instantiated in the user design as shown in Figure 11.

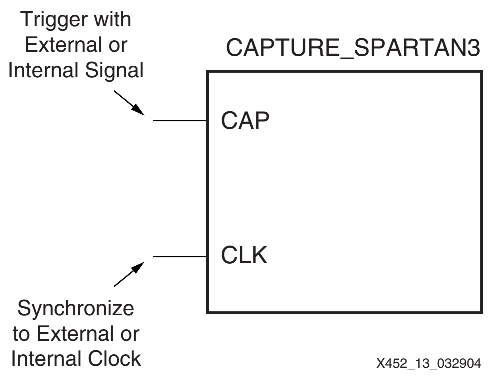
Trigger with External or Internal Signal
CAPTURE_SPARTAN3
CAP
CLK
Synchronize to External or Internal Clock
X452_13_032904

Figure 11: **Readback Capture Library Primitive**

The CAPTURE_SPARTAN3 component is used in the FPGA design to control when the logic states of all registers are captured into configuration memory. The CLK pin can be driven by any clock source that synchronizes Capture to the changing logic states of the registers. The CAP pin is an enable control. When CAP is asserted, the register states are captured in memory on the CLK rising edge.

Enabling Readback in the Software

If readback is to be performed through the SelectMAP interface after configuration, the configuration ports must stay active. Additionally, a readback bit file, which contains the commands to execute a readback and a bitmap for data verification, may be optionally generated by setting the readback option in BitGen. An example of the BitGen command line is shown below:

```
bitgen -w -l -m -b -g readback -g persist:yes -g security:none ...
```

Table 14 defines the options used in the example BitGen command line.

Table 14: **BitGen Options**

Option	Description
-w	Overwrites existing output.
-l	Generates a Logic Allocation file, shown in Figure 12.
-m	Generates a Mask file.
-b	Generates optional ASCII format files for all its binary counterparts.
-g readback	Generates the readback bit file.
-g persist:yes	Keeps the SelectMAP interface active after configuration.
-g security:none	Set by default and should be kept at default if readback is desired. At security level 1 or 2, readback is disabled.

Table 15 lists all the associated BitGen files used for readback.

Table 15: **Bitgen Files used in Readback**

File Extension	File Type	File Description
.rbb	Binary	Binary command sets and verification bitmap
.rba	ASCII	ASCII command sets and verification bitmap
.msk	Binary	Binary command sets and verification data mask
.ll	ASCII	ASCII bit number and location of registers, SRL16, LUT RAM, Block RAM

For more information about BitGen options, refer to the BitGen chapter in the *Development System Reference Guide*.

Creating Readback Commands

Different readback commands are required depending on the information desired, but the basic steps of a readback command set are the same, as indicated below:

1. Issue a Synchronization word (0xAA995566).
2. Issue a Shutdown command (Optional. For readback on designs containing Block RAM, LUT RAM, or SRL16, it is strongly recommended to shutdown the device during readback to prevent the readback process from corrupting the memory contents.)
3. Specify the frame length in the FLR register.
4. Issue an RCRC command.
5. Issue an RCFG command.
6. Specify the frame address in the FAR register.
7. Load the FDRO register.
8. Flush the command pipe with 32 bits of zeros.
9. Change pertinent SelectMAP signals to transition from a write of the data bus to a read.
10. Read the data.
11. Repeat steps 3 through 10 as needed.
12. Issue a Start command. (Optional. Only needed if SHUTDOWN is performed.)
13. Issue a DESYNC command (Optional).

Table 16 provides the commands for a readback example.

Table 16: **Sample Readback Commands for XC3S400**

Data Description	Data Field
Synchronization Word	0xAA995566
CMD Write Packet Header(1)	0x30008001
CMD Write Packet Data (SHUTDOWN)(1)	0x0000000B
FLR Write Packet Header	0x30016001
FLR Write Packet Data	0x00000044(2)
CMD Write Packet Header	0x30008001
CMD Write Packet Data (RCRC)	0x00000007
CMD Write Packet Header	0x30008001
CMD Write Packet Data (RCFG)	0x00000004
FAR Write Packet Header	0x30002001
FAR Write Packet Data	0x00000000(2)
FDRO Read Packet Header (Type 1)	0x28006000
FDRO Read Packet Header (Type 2)	0x4800CF00(2)
No Op (one word)	0x20000000

Notes:

1. Optional.
2. This value may be different based on device/design options.

The shutdown sequence is strongly recommended because LUT RAM, SRL16, and Block RAM have shared hardware between the configuration logic and the user logic. If both try to access the memory bits at the same time, the memory contents can be corrupted. If the Block RAM is configured to be read-only, then it is safe to read back the contents without corrupting them. However, when performing readback on Block RAMs, read operations should be halted because configuration logic takes control over Block RAMs during readback. LUT RAM and SRL16, however, should only be read back after a shutdown sequence to avoid possible memory corruption. A shutdown sequence de-asserts the GWE (Global Write Enable) signal, which disables user access to the memory bits and allows the configuration logic to read or write the memory contents without conflict.

If active readback is desired for designs containing LUT RAM, Block RAM, or SRL16, then it is best to perform multiple readbacks and skip the frames that contain these memory elements. To best achieve this, align memory elements in the same columns when designing to ease the readback process.

When skipping CLB frames, it is critical to stop readback one frame prior to the frame containing the memory elements. If the frame prior to the frame containing the memory elements is read back, we still risk corrupting configuration logic. For example, if the CLB frame with Minor Address 0 contains memory elements, readback must be stopped after the frame with Minor Address 17 of the previous CLB column. This process must be done carefully with the exact readback words specified in the FDRO. Otherwise, if insufficient words are read out other than those specified in the FDRO, a readback interrupt is considered to have occurred. In this case, the subsequent readback must include a readback FAR clearance instruction, as noted in Table 20.

It is also possible that the DCM might lose its lock during shutdown, if it has an external feedback path. The SHUTDOWN command causes all FPGA outputs to be 3-stated. Thus if the DCM has external feedback generated by itself, the lock might be lost. In this case, the DCM must be reset either through the user design.

The simplest readback to perform is a read of a single configuration register. No design information is required, and all the necessary information has been presented in the "Spartan-3 Device Configuration Registers" and "Bitstream Composition" sections. A useful example is to read back the STATUS configuration register. Using the information above, a packet can be constructed that reads one 32-bit word from that register. Table 17 shows the packet header.

Table 17: **Sample Status Register Readback Commands**

Data Description	Data Field
Dummy word	0xFFFFFFFF
Sync Word	0xAA995566
Status Register Read	0x2800E002
Flush pipe (2 words)	0x00000000
CMD Write Packet Header	0x30008001
CMD Write Packet Data (RCRC)(1)	0x00000007
Flush pipe (2 words)	0x00000000

Notes:

1. Because the read status activity causes the CRC_ERROR status to be asserted, it is important to clear the CRC_ERROR status bit to ensure normal device operation. Writing the precalculated CRC value to the CRC register or writing an RCRC command can do this.

If certain design information is desired, then a FAR – FDRO pair is required. To find the proper frame address to read back, the .ll (Logic Allocation) file is used. Figure 12 lists an example file.

The first part of the file explains all the fields in the .ll file. There are bit position columns and logic information. The OFFSET field starts counting with bit position 0 of the readback data, which includes the pad frame. The Frame Address specifies the FAR value if specific frame readback is desired. The Frame Offset specifies the bit offset within the specified frame address. In Figure 12, there are IOB status bits (Latch = I), CLB flip-flop status bits (Latch = YQ), CLB LUT bits (Ram = M:18), and Block RAM bits (Ram:B:BIT847). In the case where a net name is applicable, that name is included for design correlation purposes.

```
Revision 3
; Created by bitgen G.26 at Tue Jan 06 11:30:43 2004
; Bit lines have the following form:
; <offset> <frame address> <frame offset> <information>
; <information> may be zero or more <kw>=<value> pairs
; Block=<blockname     specifies the block associated with this
;                      memory cell.
;
; Latch=<name>         specifies the latch associated with this memory
cell.
;
; Net=<netname>        specifies the user net associated with this
;                      memory cell.
;
; COMPARE=[YES | NO]   specifies whether or not it is appropriate
;                      to compare this bit position between a
;                      "program" and a "readback" bitstream.
;                      If not present the default is NO.
;
; Ram=<ram id>:<bit>   This is used in cases where a CLB function
; Rom=<ram id>:<bit>   generator is used as RAM (or ROM).  <Ram id>
;                      is either 'F', 'G', or 'M', indicating that
;                      it is part of a single F or G function
;                      generator used as RAM, or as a single RAM
;                      (or ROM) built from both F and G.  <Bit> is
;                      a decimal number.
;
; Info lines have the following form:
; Info <name>=<value>  specifies a bit associated with the LCA
;                      configuration options, and the value of
;                      that bit.  The names of these bits may have
;                     special meaning to software reading the .ll file.
;
Info STARTSEL0=1
Bit    15824 0x00040200    368 Block=D3 Latch=O2 Net=xn_index_0_OBUF
Bit    21039 0x00040600   1167 Block=J2 Latch=I Net=xn_re_6_IBUF
Bit   309402 0x00120200    282 Block=SLICE_X12Y57 Latch=YQ Net=U0/N752

Bit   392413 0x00160000   1597 Block=SLICE_X16Y16 Ram=M:18
Bit   392414 0x00160000   1598 Block=SLICE_X16Y16 Ram=M:17
Bit   392415 0x00160000   1599 Block=SLICE_X16Y16 Ram=M:16

Bit  1281757 0x02000400   1117 Block=RAMB16_X0Y3 Ram=B:BIT847
Bit  1281758 0x02000400   1118 Block=RAMB16_X0Y3 Ram=B:BIT94
Bit  1281759 0x02000400   1119 Block=RAMB16_X0Y3 Ram=B:BIT222
```

Figure 12: **Sample .ll File**

Changing from a Write to a Read

After the configuration logic is sent data indicating a readback is desired, the $\overline{\text{CS}}$ and $\overline{\text{RDWR}}$ signals must change as defined below in order to switch from a write operation to a read. It is critical to follow the following sequence exactly, otherwise an ABORT sequence might be triggered.

1. Deassert $\overline{\text{CS}}$.
2. Deassert $\overline{\text{RDWR}}$.
3. Assert $\overline{\text{CS}}$.
4. On the following rising CCLK edge, data is driven out on the SelectMAP data bus. (Note: readback data is valid only when the BUSY pin is Low.)

Table 18 shows the truth table for the SelectMAP data bus. The effect of CS_B on D[0:7] bus is asynchronous to CCLK.

Table 18: **SelectMAP Bus Truth Table**

PROG_B	CS_B	BUSY	RDWR_B	D[0:7]
0	X	X	X	High-Z
1	0	0	0	Input
1	0	0	1	Active Output
1	0	1	0	Input; data not registered
1	0	1	1	Active Output; data not valid
1	1	High-Z	X	High-Z

Figure 13 shows the timing when a write operation is switched to a read.

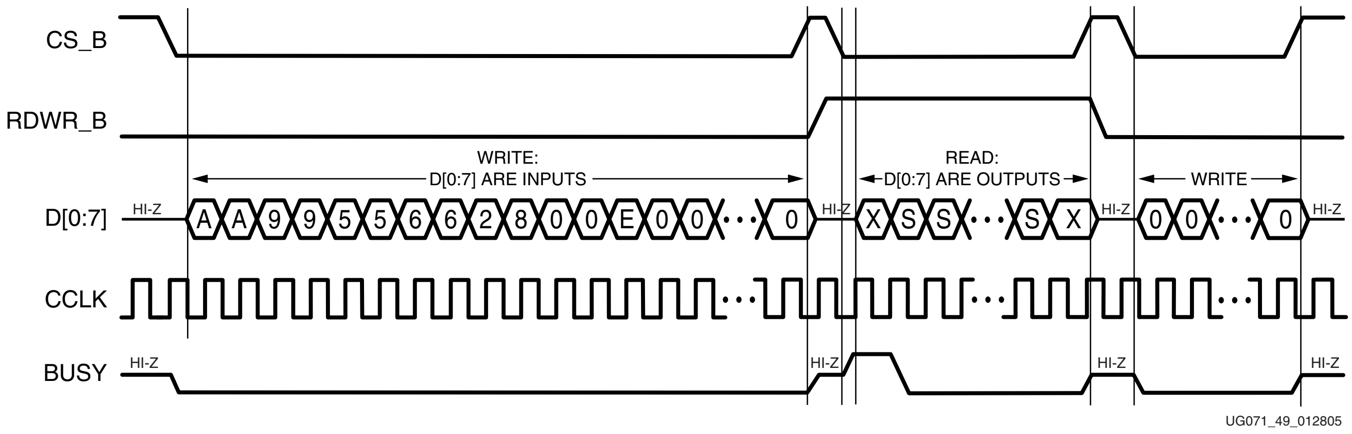

Figure 13: **Timing Diagram for a Write to Read Operation Change**

Configuration and Readback Interrupt

SelectMAP ABORT

An ABORT is an interruption in the SelectMAP configuration or readback sequence that occurs when the state of RDWR_B changes while CS_B is asserted. ABORT in the Spartan-3 devices is an asynchronous event, different from Spartan-II and Spartan-IIE devices. During a configuration ABORT, an 8-bit status word is driven onto the D[0:7] pins over the next four CCLK cycles. After the ABORT sequence finishes, the user may re-synchronize the configuration logic and resume configuration.

Configuration ABORT Sequence Description

An ABORT is signaled during configuration as follows:

1. The Configuration sequence begins normally.
2. The RDWR_B pin is pulled High while the device is selected (CS_B is asserted Low).
3. BUSY goes High if CS_B remains asserted (Low). The FPGA drives the status word onto the data pins if RDWR_B remains High for read control.
4. The ABORT ends when CS_B is deasserted.

Figure 14 shows the timing for an ABORT sequence.

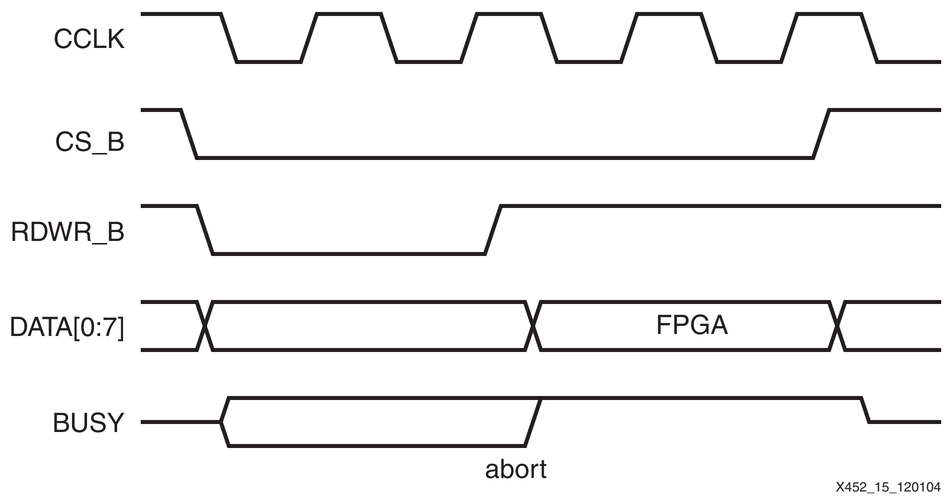

Figure 14: **ABORT Sequence Timing Diagram**

ABORT Status Word

During the configuration ABORT sequence, the device drives a status word onto the D[0:7] pins. Table 19 lists the key for that status word.

Table 19: **ABORT Status Word**

Bit Number	Status Bit Name	Meaning
D7	CFGERR_B	Configuration error (active Low) 0 = A configuration error has occurred 1 = No configuration error
D6	DALIGN	Sync word received (active High) 0 = No sync word has been received 1 = Sync word received by interface logic
D5	RIP	Readback in progress (active High) 0 = No readback is in progress 1 = A readback is in progress
D4	IN_ABORT_B	ABORT in progress (active Low) 0 = ABORT is in progress 1 = No ABORT is in progress
D3 - D0	1111	Fixed value

The ABORT sequence lasts four CCLK cycles. During those cycles, the status word changes to reflect data alignment and ABORT status. A typical sequence is:

```
11011111 => DALIGN = 1, IN_ABORT_B = 1
11001111 => DALIGN = 1, IN_ABORT_B = 0
10001111 => DALIGN = 0, IN_ABORT_B = 0
10011111 => DALIGN = 0, IN_ABORT_B = 1
```

After the last cycle, the synchronization word can be reloaded to establish data alignment.

Readback ABORT Sequence Description

An ABORT is signaled during readback as follows:

1. The Readback sequence begins normally.
2. The RDWR_B pin is pulled Low while the device is selected (CS_B is asserted Low).
3. BUSY goes High if CS_B remains asserted (Low).
4. The ABORT ends when CS_B is deasserted.

Figure 15 shows the timing for a readback ABORT.

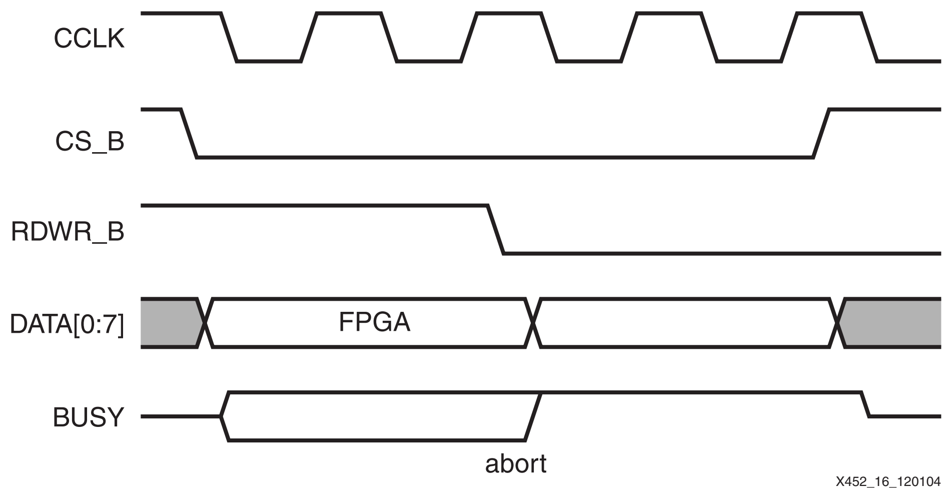

Figure 15: **Timing Diagram for Readback ABORT**

Note that ABORTs during readback are not followed by a status word, because RDWR_B is set for write control (FPGA D[0:7] pins are inputs).

Resuming Configuration or Readback After an ABORT

There are two ways to resume configuration or readback after an ABORT:

- The device can be resynchronized after the ABORT completes.
- The device can be reset by pulsing PROG_B Low at any time.

To resynchronize the device, CS_B must be deasserted, then reasserted. The configuration synchronization word can then be sent. Configuration or readback can be resumed by sending the last configuration/readback packet that was in progress when the ABORT occurred. Alternatively, configuration/readback can be restarted from the beginning.

When resynchronizing the device to continue readback, caution must be taken. If subsequent readback is started while the previous readback was interrupted, a race between the two control signals can be created. This race can cause two columns of configuration frames to be overwritten with the contents of a third frame. The two frames corrupted are the frames after the last frame read and frame 0.

To avoid configuration logic corruption, the instructions listed in Table 20 must be sent to the device prior to reinitializing the readback process or the configuration process.

Table 20: **Readback FAR Clearance Instructions for XC3S400**

Data Description	Data Field
Synchronization Word	0xAA995566
FLR Write Packet Header	0x30016001
FLR Write Packet Data	0x00000044(1)
FAR Write Packet Header	0x30002001
FAR Write Packet Data	0xFFFFFFFF
CMD Write Packet Header	0x30008001
CMD Write Packet Data (RCFG)	0x00000004
FDRO Write Packet Header (Type 1)	0x28006000
FDRO Write Packet Header (Type 2)	0x48000044(1)
CMD Write Packet Header	0x30008001
CMD Write Packet Data (START)	0x00000005
No Op (4 words)	0x20000000
DATA Flush(2)	–

Notes:

1. This value may be different based on device FLR. The FDRO word count should be one frame of words.
2. Toggle RDWR_D to read mode and clock out one frame of data.

Readback Data Format

The readback data stream contains the information contained within the configuration memory map (data frames) plus additional pad data produced by the pipelining process of reading the data. The readback stream does not contain any commands, options, or packet information found in the configuration stream, nor does it contain any CRC values, since this information is stored in internal configuration registers, not the configuration memory. Additionally, no CRC calculation is performed during readback.

Unlike the previous Spartan family parts (Spartan-II and Spartan-IIE devices), the Spartan-3 FPGA readback stream contains only one pad frame prior to the desired readback frame specified by the FAR register. Furthermore, there are no pad words between the readback frames.

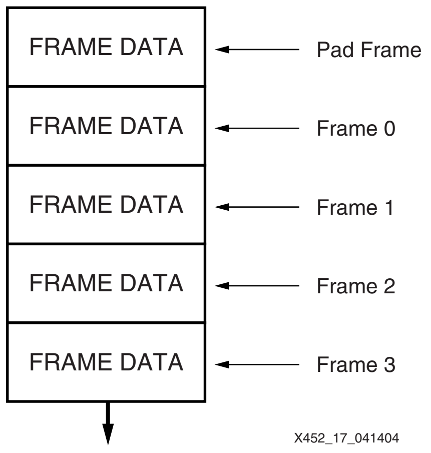

Figure 16: **Spartan-3 FPGA Readback Data Stream**

The second major difference between the Spartan-II, Spartan-IIE, and Spartan-3 devices is that DFF values are readback inverted as shown in Table 21.

Table 21: **Configuration Component Readback Truth Table**

	Spartan-II/Spartan-IIE Devices	**Spartan-3 Devices**
LUT RAM	Readback Value Inverted	Readback Value Inverted
SRL16	Readback Value Inverted	Readback Value Inverted
DFF	Readback Value in True Sense	Readback Value Inverted
Block RAM	Readback Value in True Sense	Readback Value in True Sense

Verifying Configuration Data

Readback verification is a process of making a bit per bit comparison of the readback data frames to the bitmap in the `<design>.rbb` readback file. However, not all of the readback data should be used for verification. There are three types of data bits that cannot be verified against the bitmap: pad data, RAM bits, and Capture bits. The pad data includes the pad frame, the readback commands, CRC bits, and any bit for which the device does not have a memory cell at the given location. RAM bits are configuration memory cells that hold the contents of LUT RAMs, SRL16, and Block RAMs. These values are dynamically changing per the user design. The Capture bits are the memory locations reserved for capturing internal register states. They are masked out only if the capture block is instantiated in the design. In addition, readback data is only valid when the BUSY pin is Low.

While the pad frame is a separate data frame that can be easily ignored by any system performing readback, the RAM and Capture bits are sprinkled throughout the data frames and must be masked out. However, masking out the bit information does not prevent possible LUT RAM, SRL16, or Block RAM memory corruption. To prevent memory corruption of these elements, either place the device in SHUTDOWN mode or skip the frames that contain memory elements. The `<design>.msk` mask file is used to mask out the RAM and Capture bits.

The declarations portion is throw-away data. The mask file includes command sets and the synchronization word, which can be omitted if an ABORT has not been executed.

Figure 17 shows a sample mask file.

```
0009 0FF0 0FF0 0FF0 0FF0 0000 0161 0008 746F File Declaration and header;
702E 6E63 6400 6200 0B33 7334 3030 6674 3235 Design Name, target device Date,
3600 6300 0B32 3030 342F 3031 2F30 3600 6400 etc. Size is dynamic
0931 313A 3330 3A35 3800 6500 033D A8FF FFFF
FFAA 9955 6630 0080 0100 0000 0730 0160 0100 Sync word
0000 4430 0120 0100 003F E530 01C0 0101 41C0 Readback command sets
9330 00C0 0100 0000 0030 0080 0100 0000 0930
0020 0100 0000 0030 0080 0100 0000 0130 0040
0050 00CF 00FF FFFF FFFF FFFF FFFF FFFF FFFF Masking Data
FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
FFFF FFFF FFFF ---- ---- ---- ---- ---- ---- 0000 0000
0000 0000 0000 0000 0000 0000 0000 ---- ---- ----
```

Figure 17: **Sample Mask File**

The masking data determines which data frame bits are configuration bits and should be verified against the bitmap in the `<design>.rbb` readback file, and which bits are either RAM or Capture bits and thus should be skipped. The MSK file masks out the 32 bits following each frame, but does not mask out the first 32-bit portion of the readback stream, the first frame pad data, or the following 32-bit pipeline data portion. See Figure 18 for the readback data stream alignment. The equation for this file follows:

RBB[i] = MSK[i] x DATA[i]

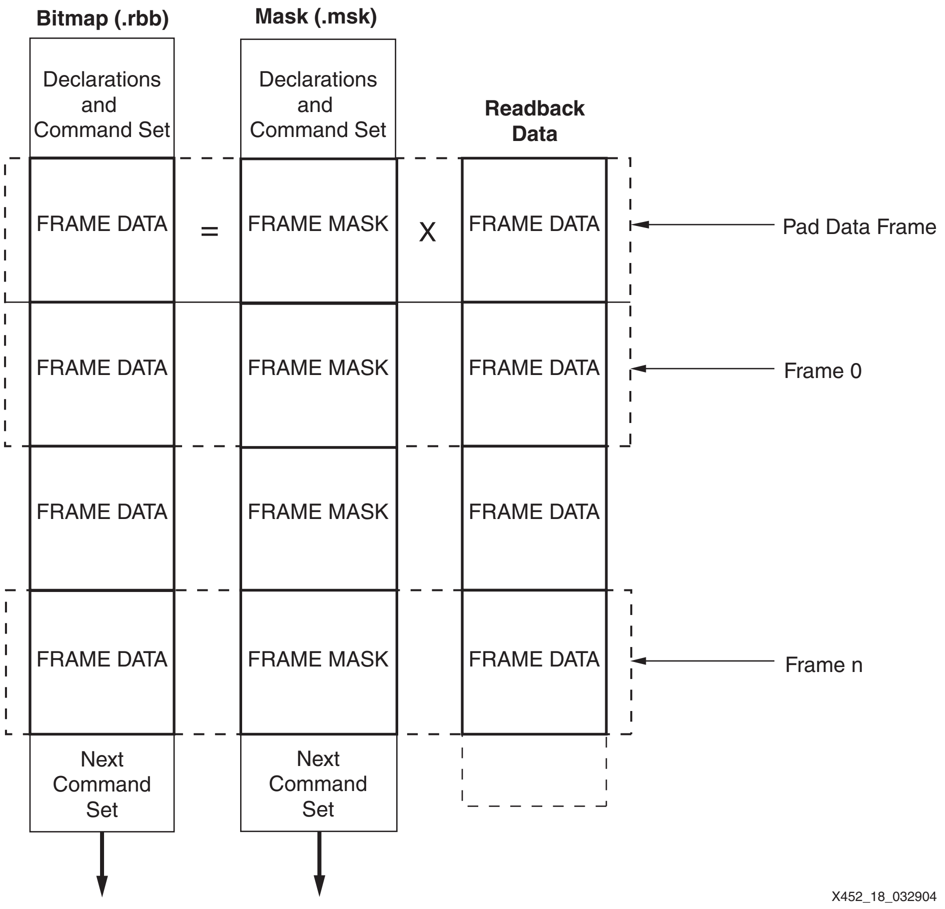

Figure 18: **Readback Data Stream Alignment**

Each bit position of the masking data corresponds to the bit position of the readback data. Therefore, the first masking data bit specifies whether to verify the first bit of the first valid frame against the bitmap `<design>.rbb` file. If the mask bit is a 0, the frame bit is verified. If the mask bit is a 1, the frame bit is not verified.

Figure 19 shows the flow for readback data verification.

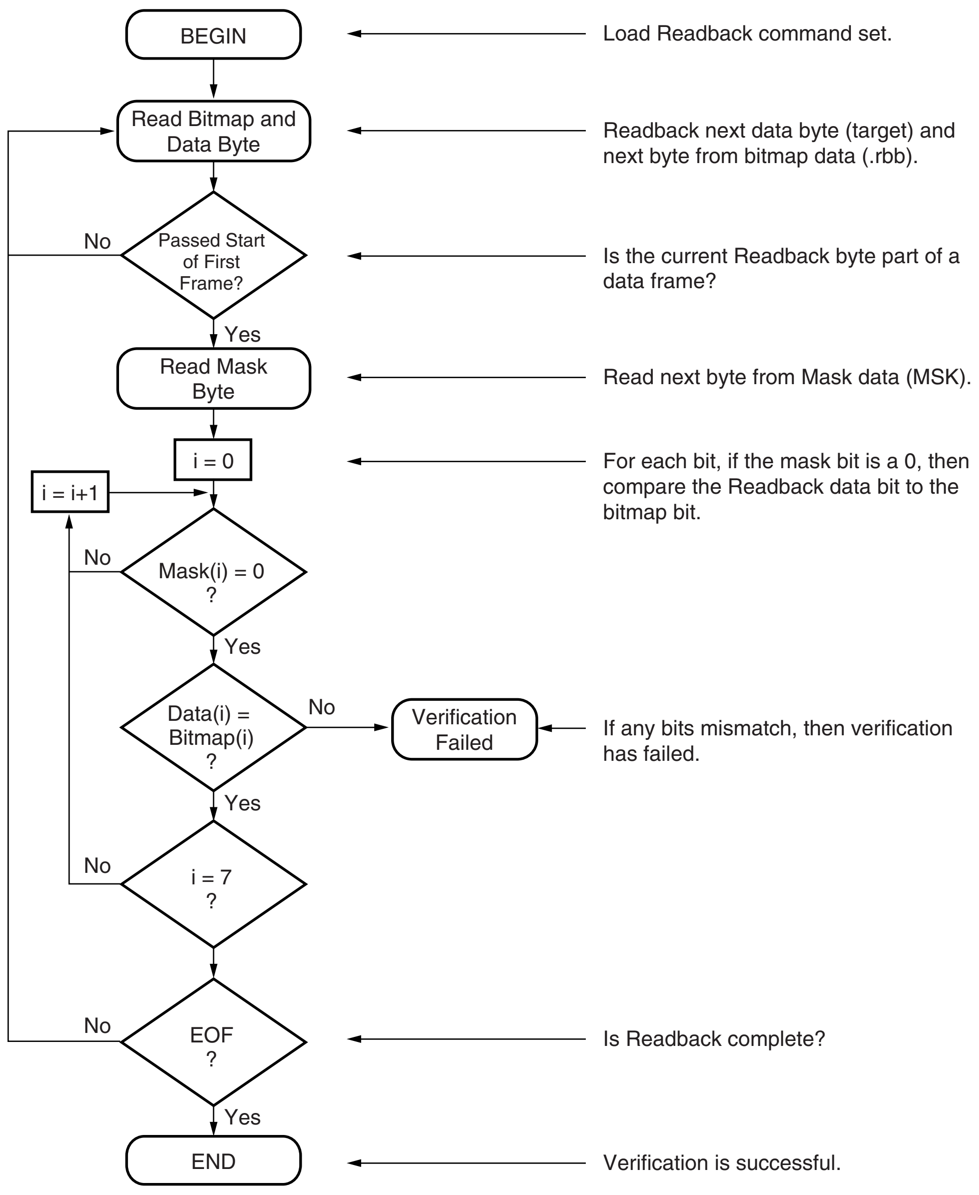
BEGIN
Load Readback command set.
Read Bitmap and Data Byte
Readback next data byte (target) and next byte from bitmap data (.rbb).
Passed Start of First Frame?
No
Yes
Is the current Readback byte part of a data frame?
Read Mask Byte
Read next byte from Mask data (MSK).
i = 0
i = i+1
For each bit, if the mask bit is a 0, then compare the Readback data bit to the bitmap bit.
Mask(i) = 0 ?
No
Yes
Data(i) = Bitmap(i) ?
No
Verification Failed
If any bits mismatch, then verification has failed.
Yes
i = 7 ?
No
EOF ?
No
Is Readback complete?
Yes
END
Verification is successful.
X452_19_032904

Figure 19: **Readback Data Verification Flow**

Although the use of the mask and the .rbb file ensures a complete verification on the FPGA, this method requires a significant amount of memory space to hold the mask and the .rbb file. It is also possible to verify the device configuration without the use of the mask or the .rbb file. An alternative method is to read back the device and calculate a frame- or column-based golden checksum on the readback data.

Partial Reconfiguration

Partial reconfiguration is not recommended or supported in the Spartan-3 family or other members of the Spartan-3 generation (Spartan-3E, Spartan-3A, Spartan-3AN, Spartan-3A DSP). The Spartan-3 generation devices require a full reconfiguration of the entire device. For partial reconfiguration applications, consider the Virtex® FPGAs, which allow configuration on a frame basis (see XAPP290), as opposed to the column-based configuration of the Spartan-3 generation. Virtex devices also provide an ICAP component for configuration.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/03/04	1.0	Initial Xilinx release.
06/25/08	1.1	Corrected XC3S1500 frame length in Table 7. Added more detail to Figure 13. Updated "Partial Reconfiguration" to note that it is not recommended or supported in the Spartan-3 generation.