Virtex™ Configuration Architecture Advanced Users' Guide

XAPP151 September 30,1999 (Version 1.2)

Application Note by Steve Kelem

Summary

The Virtex architecture supports powerful new configuration modes, including partial reconfiguration. These mechanisms are designed to give advanced applications access to and manipulation of on-chip data through the configuration interfaces.

This document is an overview of the Virtex architecture, emphasizing data bit location in the configuration bitstream. Knowing bit locations is the basis for accessing and altering on-chip data. FPGA applications can be built that change or examine the functionality of the operating circuit without stopping the circuit loaded in the chip. A glossary is included to explain some of the terminology used in this application note.

Xilinx Family:

Virtex Series

Introduction

CLBs, IOBs, and Configurations

Each Virtex chip contains configurable logic blocks (CLBs), Input-Output blocks (IOBs), Block SelectRAMs, clock resources, programmable routing, and configuration circuitry (Figure 1). These logic functions are configurable through the configuration bitstream. Configuration bitstreams contain a mix of commands and data. Configuration bitstreams can be read and written through one of the configuration ports on the chip.

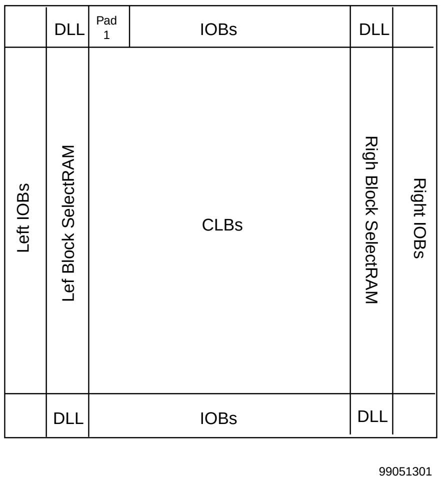

Figure 1: Virtex Architecture Overview

Configuring Virtex Devices

Virtex devices can be configured through the SelectMAP port, Master/Slave Serial port, or the Boundary Scan port. The collection of configuration bits is called a Configuration Bitstream. The bitstream is a series of configuration commands and configuration data, as shown in Figure 2. Bitstream architecture is discussed in "Configuration Logic Basics" on page 6.

sync	CMD 1	CMD 2	CMD 3	data	CMD 3	data

Figure 2: Bitstream Architecture

Writing all or some of a configuration is done by issuing configuration commands to the desired interface followed by the configuration data.

The SelectMAP port is an 8-bit port on the chip, whose pins are labeled D[7:0]. The configuration bitstream can be written eight bits at a time. Virtex chips can be configured to retain the eight SelectMAP, BUSY/DOUT, $\overline{\text{INIT}}$, $\overline{\text{WRITE}}$, and $\overline{\text{CS}}$ pins, allowing further reconfiguration through those pins. If further reconfiguration is not required, those pins can be reconfigured as User I/O.

When the Master/Slave Serial or Boundary-Scan port is used for configuration or reconfiguration, the configuration bitstream is written one bit at a time.

Timing relationships for the configuration ports are discussed in the Virtex Data Sheet located at http://www.xilinx.com/products/virtex.htm.

Reading Configuration Bits From A Virtex Device

Configuration data can be read from the SelectMAP interface or the Boundary Scan interface. Reading all or some of a configuration is done by issuing configuration read commands to the desired interface, then reading the data from the appropriate port. Configuration commands and data formats are discussed in "Configuration Logic Basics" on page 6.

The SelectMAP port has an 8-bit data port. To read the SelectMAP interface, all 12 SelectMAP pins must remain as SelectMAP as opposed to User I/O after configuration.

The Boundary Scan (JTAG) port allows bit-at-a-time access to the configuration. The Master/Slave Serial port does not allow a read operation.

CLBs & IOBs

Virtex CLBs are organized as a rectangular array. The configuration data is organized into CLB columns. Each CLB contains two slices, each containing two logic cells. Each CLB column has two IOBs above and below it that are part of the same configuration frames as the CLBs. On the left and right edges there are three IOBs per CLB row. Configuration data for the IOBs on the left and right sides of the chip are organized as separate columns of frames. See the Virtex Data Sheet for more information on these resources.

Warning! This document discusses mechanisms for manipulating the configuration bits for the Virtex devices. If portions of the bitstream other than those described here are altered, the device may be damaged. Xilinx is not liable for any consequences of misprogramming a device.

Frame Addressing

A review of configuration data organization is helpful before discussing configuration commands for reading/writing data to/from a Virtex device.

Configuration data are grouped into frames. Frames are organized vertically; the top bits in the frame are read or written first (see Figure 3).

A column of CLBs and the two IOBs above and below each column of CLBs consists of forty-eight frames. Each IOB column, one on the left and one on the right, has columns of 54 frames of configuration data.

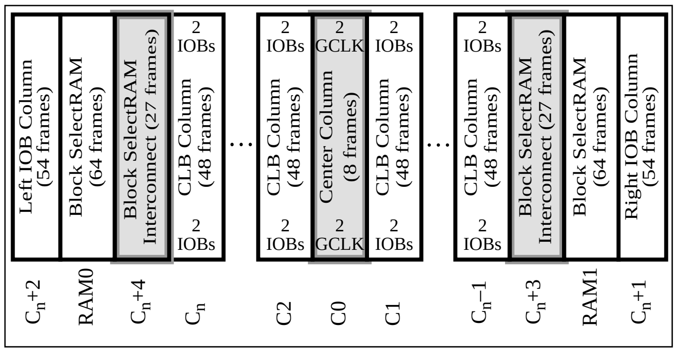

Figure 3: Allocation of Frames to Chip Resources

Each Block SelectRAM has 64 frames addressed by a Major Address and a Minor Address. The Major Address space is divided into CLB and RAM sub-spaces. CLB, IOB, Block SelectRAM interconnect, and center column circuitry belong to the CLB address space. Block SelectRAMs are addressed in the Block SelectRAM address space.

Major Addresses for the CLB columns in the XCV50 are shown in Figure 6. The first row of the figure shows the CLB column numbers. The second row shows the Major Address for each CLB column.

Note: The Major Addresses alternate between left and right sides of the chip, as shown in Figure 6. For example, C0 (representing Major Address 0 in the CLB address space), starts at address 0 in the center of the chip. The center column does not contain CLBs. Major Address 1 refers to the CLB column lying to the right of the center of the chip; Major Address 2 refers to the CLB column that lies to the left of center, and so on, alternating from right to left, up to CLB column Cn, where n is the maximum number of CLB columns. The next Major Addresses in the CLB address space are the Block SelectRAM Interconnect columns, followed by the IOB columns. Block SelectRAM is in a separate address space from the CLBs. The Major Addresses for each Block SelectRAM alternate from left to right, with odd-numbered addresses on the right side and even-numbered addresses on the left side of the chip.

Frames are read and written sequentially with ascending addresses for each operation. Multiple consecutive frames can be read or written with a single configuration command. The smallest amount of data that can be read or written with a single command is a single frame. The entire CLB array plus the IOBs and Block SelectRAM interconnect can be read or written with a single command. Each Block SelectRAM must be read or written separately.

Frame Sizes

Frame size depends on the number of rows in the chip. The number of configuration bits in a frame is 18 ×(# CLB_rows+2), and is padded with zeroes on the right (bottom) to fit in 32-bit words. See “Frame Organization” for more details. An additional padding word is needed at the end of each frame for pipelining. Table 1 shows the frame sizes for all Virtex devices. This table also shows the size, in words, of the bitstream for the CLB address space and the number of words in each RAM block.

Table 1: Frame Sizes

Device	Row × Col	Bits/ Frame	Words/ Frame	# of read-back 32-bit Words	
				CLB	RAM
XCV50	16 × 24	384	12	15,876	780
XCV100	20 × 30	448	14	22,554	910
XCV150	24 × 36	512	16	30,384	1,040
XCV200	28 × 42	576	18	39,366	1,170
XCV300	32 × 48	672	21	51,975	1,365
XCV400	40 × 60	800	25	76,275	1,625
XCV600	48 × 72	960	30	108,810	1,950
XCV800	56 × 84	1088	34	142,902	2,210
XCV1000	64 × 96	1248	39	186,381	2,535

Frame Organization

Each frame sits vertically in the chip, with the “front” of the frame at the top. As shown in Figure 4, it is convenient to consider the frame horizontally when it is viewed as part of a bitstream. The top IOBs are shown on the left followed by the CLBs in the column and the bottom IOBs on the right. Bits in frames are allocated as follows: for CLB columns, the first 18 bits control the two IOBs at the top of the column; then 18 bits are allocated for each CLB row; finally, the next 18 bits control the two IOBs at the bottom of the CLB column. The frame then contains enough “pad” bits to make it an integral multiple of 32 bits. For left and right IOB columns, the frame allocates 18 bits per three IOB rows, Figure 5. When reading and writing frames, bits are grouped into 32-bit words, starting on the left (corresponding to the top of the chip). If the last word does not completely fill a 32-bit word, it is padded on the right with zeroes.

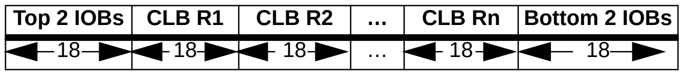

Figure 4: CLB Frame Organization

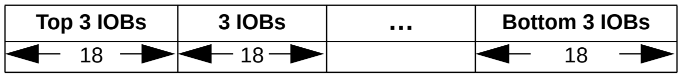

Figure 5: IOB Frame Organization

Location of LUT SelectRAM Bits

With respect to the beginning of a configuration frame, relative locations of LUT SelectRAM bits within the bitstream are the same for every CLB slice. LUT SelectRAM bits spread across 16 consecutive frame Minor Addresses. Each frame Minor Address contains all instances of a single bit. These 16 frames contain all 16 bits of the LUT SelectRAM for a column of CLB slices. You must read or write the 16 frames containing those bits to read or write the entire LUT SelectRAM.

Note: LUT SelectRAM bits are stored inverted. Flip-Flop data are stored in their true sense. When reading or writing LUT SelectRAM data from the bitstream, it is negated from the logic sense of the data. For example, a 4-input AND gate has the truth table `LUT[15:0] = 1000000000000000`.

This truth table is stored internally in the LUT SelectRAM as $\overline{\texttt{LUT[15:0]}}$ `= 0111111111111111`. Of course, user logic reading the data from the LUT SelectRAM would read the correct logic value, `LUT[15:0] = 1000000000000000`.

C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
MA	24	22	20	18	16	14	12	10	8	6	4	2	1	3	5	7	9	11	13	15	17	19	21	23

Figure 6: XCV50 Major Addresses for CLB Columns

Configuration Data Operations

Reading Configuration Data

Configuration information can be read from the Virtex devices if readback is enabled in the current configuration. (See the description of the Configuration Option Register field SBITS on page 9.) CLB and IOB configuration data can be read while the chip is operating.

When reading data from a Virtex device, a padding frame is read before any of the data frames. This is in addition to the pad word in each data frame. The pad frame must be included in the word count to be read from the device.

Each frame read from the device consists of the number of words shown in Table 1.

For example, the XCV150 has 16-word frames. When reading frames, the first word of a frame is a pad word. Including the pad frame, the XCV150 has 1,899 (30,384/16) frames. (See “XCV150 Frame Padding for Device Read” on page 3.)

Pad Frame (16 words)	
Pad Word	Data Frame 0 (15 words)
⋮	⋮
Pad Word	Data Frame n (15 words)

Figure 7: XCV150 Frame Padding for Device Read

Writing Configuration Data

Configuration information can be written to a Virtex device while the device is running if writing is enabled in the current configuration. (See the Control Register field SBITS on page 9.) Re-writing the same configuration data does not generate any transient signals.

Changing configuration data may generate transient signals, especially if LUT values or signal routing is changed. For this case, all the logic cells and routing can be placed in a non-contentious state by asserting the GHIGH_B signal. See the description for the “Command Register (CMD)” on page 7,

When writing configuration data to the Virtex device, whether from bitstream files (`.bit` or `.rbt` file extensions), the SelectMAP or JTAG interfaces, the data frames are written to the device with each frame followed by a pad word. After all the data frames are written, a pad frame must be written (to flush internal pipelines). The pad words and pad frame must be included in the number of words to be written to the device. (Table 1.)

For example, the XCV1000 has 39-word frames. When writing frames, the last word of the frame is a pad word. Including the pad frame, the XCV1000 has 4,779 (186,381/39) frames. (See Figure 8.)

Data Frame 0 (38 words)	Pad Word
⋮	
Data Frame n (38 words)	Pad Word
Pad Frame (39 words)	

Figure 8: XCV1000 Frame Padding for Device Write

Altering Configuration Data

Virtex devices support the Read-Modify-Write (RMW) method of changing LUT SelectRAM data. Therefore it is possible to read or alter the contents of one or more LUT SelectRAMs through the JTAG or SelectMAP interfaces. *When writing data to one or more LUT SelectRAMs, all the bits in the frame **must** have valid configuration information.* This can be assured by altering valid configurations from bitstream files or from frames read from a properly configured Virtex chip. When using the RMW method, it may be expeditious to read the data, ignoring the pad frame and the pad word of only the first frame. This aligns the remaining configuration data in the DeviceWrite format. After modifying the configuration, the altered data can be written to the Virtex device followed by a pad word and a pad frame.

It is not necessary to retain the contents of the pad frame or pad words. However, the pad words **are** included in the CRC calculation

Note: Frames span an entire column of CLB slices (or IOBs). Thus, when changing LUT SelectRAM bit 0 for a single CLB slice (*e.g.*, R3C4.S1), LUT SelectRAM bit 0 for *all* slices in that column (*i.e.*, R*C4.S1) is written with the same command. One must ensure that either all other data in the slice is constant or changed externally through partial configurations.

LUT SelectRAMs not configured externally should not lie in the same slice with LUTs or LUT SelectRAMs that are reconfigured externally. Such a mixture can cause the unrelated LUT SelectRAM data to be reconfigured when the frames are written to the device. Figure 9 shows what can happen if this restriction is not observed.

In this example, it is the objective to perform a Read-Modify-Write on the LUT SelectRAM in R2C3.S1 in column 3, which is shown in the first column in the figure. Row 2 contains a LUT containing the value AB. Row 3 contains a SelectRAM containing the value C3. Because the Read and Write operations operate on an entire frame, the RAM in R3C3.S1 is also read and written when R2C3.S1 is read and written. Performing the Readback operation reads all the LUT and SelectRAM values for the column. Before the new value for the LUT is written, it is possible for the on-chip circuitry to write a new value, such as 14, into the SelectRAM. When the new value, BD, is written via the configuration port into LUT R2C3.S1, the value C3 is also written into RAM R3C3.S1. This may not always be desirable (It is design-dependent).

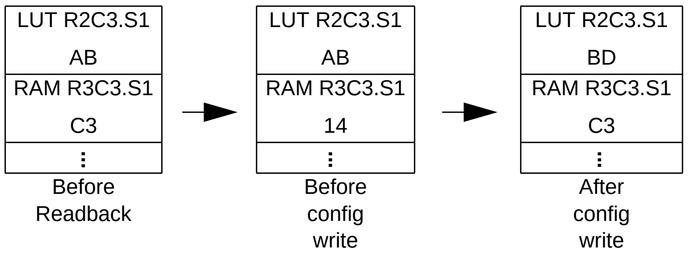

Figure 9: Potential Write Conflicts

Definitions

Two sets of variables are defined to determine where a desired bit is in the configuration data. The first is a set of “independent” variables, or design attributes, namely characteristics of the design that are known, *i.e.,* the chip size and which CLB and bit(s) within the CLB to locate, Table 2. The second set is a set of “dependent” variables or design variables, namely values that must be calculated to find the bit(s) of interest listed in Table 3.

Table 2: Design Attributes (Independent Variables)

Term	Definition
Chip_Cols	# of CLB columns on the Virtex chip
Chip_Rows	# of CLB rows on the Virtex chip
CLB_Col	Column number of the desired CLB.
Slice	0 or 1
FG	0 for the F-LUT, 1 for the G-LUT
lut_bit	The desired bit from the given LUT. Bits in the LUT are indexed from 0 to 15.
FL	# of 32-bit words in the frame.
XY	0 for the X Flip-Flop, 1 for the Y Flip-Flop
RW	1 for Read, 0 for Write

Table 3: Design Variables (Dependent Variables)

Term	Definition
MJA	Frame Major Address
MNA	Frame Minor Address
fm_st_wd	The index of the word in the full configuration of the starting word of the frame. Numbered starting at 1.
fm_wd	The number of the 32-bit word within a frame that contains the desired bit. Words in a frame are numbered starting at 0.
fm_wd_bit_idx	The bit index of the desired bit within frame word fm_wd. Words are indexed in the “big-endian” style, with bit 31 on the left and bit 0 on the right.
fm_bit_idx	Bit index within a frame of the given bit. Numbered starting with 0 as the left-most (first) bit. Bit numbering within a frame continues across all the words in the frame.

Equations

Table 4 shows equations for the LUT SelectRAM “Dependent Variables”, That were defined in Table 3.

Table 4: Equations for LUT SelectRAM Dependent Variables

Term	Definition

MJA =	if (CLB_Col $\le$ Chip_Cols/2)
	then Chip_Cols – CLB_Col $\times$ 2 + 2
	else 2 $\times$ CLB_Col – Chip_Cols – 1

MNA =	lut_bit + 32 – Slice $\times$ (2 $\times$ lut_bit + 17)
fm_bit_idx =	3 + 18 $\times$ CLB_Row – FG + RW $\times$ 32
fm_st_wd =	FL $\times$ (8 + (MJA –1) $\times$ 48 + MNA) + RW $\times$ FL
fm_wd =	floor$^{(\dagger)}$ (fm_bit_idx/32)
fm_wd_bit_idx =	31 + 32 $\times$ fm_wd – fm_bit_idx

† *floor(x)* is the largest integer not larger than *x. E.g.*, floor(3.1) = 3, because the next largest integer, 4, is larger than 3.1, floor(3)=3.

LUT SelectRAM Examples

See “LUT SelectRAM Examples” on page 13 for several examples of reading and evaluating configuration data. The examples illustrate how to make use of these equations to find the desired data in a bitstream.

CLB Flip-Flops

Equations for the CLB flip-flops can be found in Table 5. Their locations are calculated similarly to the LUT SelectRAM locations. Equations for the CLB FF “Dependent Variables” are defined in Table 4.

Table 5: Equations for CLB FF Dependent Variables

Term	Definition

MJA=	if (CLB_Col $\le$ Chip_Cols/2)
	then Chip_Cols – CLB_Col $\times$ 2 + 2
	else 2 $\times$ CLB_Col – Chip_Cols – 1

MNA=	Slice $\times$ (12 $\times$ XY – 43) – 6 $\times$ XY + 45
fm_bit_idx=	(18 $\times$ CLB_Row) + 1 + (32 $\times$ RW)
fm_st_wd=	FL $\times$ (8 + (MJA –1) $\times$ 48 +MNA) + RW $\times$ FL
fm_wd=	floor(fm_bit_idx/32)
fm_wd_bit_idx=	31 + 32 $\times$ fm_wd – fm_bit_idx

IOB Values

Each IOB contains four values that can be captured into special registers. These values are:

- I — the input flip-flop
- O — the output flip-flop
- T — the flip-flop for the tri-state control
- P — the value of the I/O pad.

These values are captured by utilizing the CAPTURE_VIRTEX symbol in your design. The Libraries Guide has more details on the use of this symbol. The following registers will be read as part of the readback data.

Access to the IOB flip-flops is different for the top and bottom IOBs and for the left and right IOBs.

The top and bottom IOBs are part of the CLB column frames. There are two IOBs above and below each CLB column.

The left and right IOBs are in columns by themselves. There are three IOBs per CLB row.

IOBs are numbered clockwise around the die. Pad 1 is located at the left side of the top edge, above CLB column 1. The equations for where to find IOB flip-flops in the bitstream are based on the ***pad number*** *which is the same for a given size chip**,** not the package pin name, which varies from package to package*. The mapping from package pin names to pad numbers can be found in *EPIC* or *fpga_editor.*

Table 6 contains the numeric pad indices for the pads on all four edges of the chip in terms of the number of CLB columns and rows on the chip.

Table 6: IOB Pad Indices

Pad Location	Pad Index i
Top	1 $\le$ i $\le$ Chip_Cols $\times$ 2
Right	Chip_Cols $\times$ 2 +1 $\le$ i $\le$ Chip_Cols $\times$ 2 + Chip_Rows $\times$ 3
Bottom	Chip_Cols $\times$ 2 + Chip_Rows $\times$ 3 + 1 $\le$ i $\le$ Chip_Cols $\times$ 4 + Chip_Rows $\times$ 3
Left	Chip_Cols $\times$ 4 + Chip_Rows $\times$ 3 + 1 $\le$ i $\le$ Chip_Cols $\times$ 4 + Chip_Rows $\times$ 6

Table 7 shows the equations for the dependent variables for the IOB flip-flops. The variable ***i*** in this table refers to the index of pad ***i***.

Table 7: Equations for IOB Flip-Flop Dependent Variables

Term		Definition	

MJA	Top		if $(i \le \text{Chip\_Cols})$
			then $\text{Chip\_Cols} - \text{ceiling}^{(\dagger)}(i/2) \times 2 + 2$
			else $2 \times \text{ceiling}(i/2) - \text{Chip\_Cols} - 1$

	Right		$\text{Chip\_Cols} + 1$

	Bottom		if $(i > 3 \times (\text{Chip\_Cols} + \text{Chip\_Rows}))$
			then $2 \times \text{ceiling}((i - 3 \times \text{Chip\_Cols} - 3 \times \text{Chip\_Rows})/2)$
			else $\text{Chip\_Cols} - 2 \times \text{floor}((i - 2 \times \text{Chip\_Cols} - 3 \times \text{Chip\_Rows} - 1)/2) - 1$

	Left		$\text{Chip\_Cols} + 2$
MNA	Top	**I**	$-25 \times (i\%^{(\ddagger)}2) + 45$
		O	$-13 \times (i\%2) + 39$
		T	$-5 \times (i\%2) + 35$
		P	$-4 \times (i\%2) + 25$
	Right	**I**	$t = (i - 2 \times \text{Chip\_Cols}) \% 3$; $\text{MNA} = 27.5 \times t^2 - 57.5 \times t + 32$
		O	$t = (i - 2 \times \text{Chip\_Cols}) \% 3$; $\text{MNA} = 21.5 \times t^2 - 51.5 \times t + 38$
		T	$t = (i - 2 \times \text{Chip\_Cols}) \% 3$; $\text{MNA} = 17.5 \times t^2 - 47.5 \times t + 42$
		P	$\text{MNA} = 50$
	Bottom	**I**	$25 \times ((i - \text{Chip\_Rows})\%2) + 20$
		O	$13 \times ((i - \text{Chip\_Rows})\%2) + 26$
		T	$5 \times ((i - \text{Chip\_Rows})\%2) + 30$
		P	$4 \times ((i - \text{Chip\_Rows})\%2) + 21$
	Left	**I**	$t = (i - \text{Chip\_Cols})\%3$; $\text{MNA} = 17.5 \times t^2 - 47.5 \times t + 45$
		O	$t = (i - \text{Chip\_Cols})\%3$; $\text{MNA} = 23.5 \times t^2 - 53.5 \times t + 39$
		T	$t = (i - \text{Chip\_Cols})\%3$; $\text{MNA} = 27.5 \times t^2 - 57.5 \times t + 35$
		P	50
fm_bit_idx	Top		$32 \times \text{RW}$
	Right	**I O & T**	$t = (i - 2 \times \text{Chip\_Cols}) \% 3$; $\text{fm\_bit\_idx} = 18 \times (1 + \text{floor}((i - 2 \times \text{Chip\_Cols} - 1)/3)) + 6 \times t^2 - 17 \times t + 15 + 32 \times \text{RW}$
		P	$\text{fm\_bit\_idx} = 18 \times (1 + \text{floor}((i - 2 \times \text{Chip\_Cols} - 1)/3)) + 32 \times \text{RW}$
	Bottom		$\text{fm\_bit\_idx} = 18 \times (\text{Chip\_Rows} + 1) + 32 \times \text{RW}$
	Left	**I O & T**	$\text{fm\_bit\_idx} = 18 \times (\text{Chip\_Rows} - \text{floor}((i - 4 \times \text{Chip\_Cols} - 3 \times \text{Chip\_Rows} - 1)/3)) + 32 \times \text{RW}$
		P	$t = (i - 4 \times \text{Chip\_Cols} - 3 \times \text{Chip\_Rows})\%3$; $\text{fm\_bit\_idx} = 18 \times (\text{Chip\_Rows} - \text{floor}((i - 4 \times \text{Chip\_Cols} - 3 \times \text{Chip\_Rows} - 1)/3)) - 10.5 \times t^2 + 21.5 \times t + 4 + 32 \times \text{RW}$

fm_st_wd			if $(\text{MJA} > \text{Chip\_Cols} + 1)$
			then
			$\text{FL} \times (54 \times \text{MJA} - 46 + \text{MNA} - 6 \times \text{Chip\_Cols}) + \text{RW} \times \text{FL}$
			else $\text{FL} \times (48 \times \text{MJA} - 40 + \text{MNA}) + \text{RW} \times \text{FL}$

fm_wd			$\text{floor}(\text{fm\_bit\_idx}/32)$
fm_wd_bit_idx			$31 + 32 \times \text{fm\_wd} - \text{fm\_bit\_idx}$

†: *Ceiling (x)* is the smallest integer greater than or equal to *x*. For example, ceiling(3.2) = 4, because 4 is greater than 3.2, and 3 is not, ceiling (4) = 4.

‡: % is the modulus operation. 5 % 2 = 1, 5 % 3 = 2, 5 % 5 = 0

Configuration Logic Basics

Configuration Data

Configuration data is organized as 32-bit words. There are two major commands that the configuration data can contain — Read and Write. A configuration command is executed when the configuration command is read or written to the appropriate command register.

The OP field contains `01` for a Read operation, and `10` for a Write operation. (See Figure 10).

OP	CODE
Read	01
Write	10

Figure 10: OP Field Code

A command is organized as a packet with a header word and optional data words. The header word is the first word written to the appropriate command register for a read or write operation. The header word contains a type field (001), an operand field, a register address field, and a word count field. The format for the command header is shown in Figure 11.

The Register Address field defines the target of this command, as defined in Table 8 on page 7.

The header word count field contains an integer between 0 and 2,047 and indicates the number of words that follow the header. Larger word counts (between 2,048 and 2,097,152 words) are achieved by setting the header word count to 0. The Extension header word has a type field = 010, an OP field that must match the OP field in the preceding Command Header word, and a 21-bit word count, this format is shown Figure 12.

Configuration Flow

Virtex devices are configured by presenting configuration data to the SelectMAP or JTAG ports in a specific sequence.

For Initial Configuration

1. Issue one or more pad words (SelectMAP only).
2. If initial configuration or configuration was aborted, issue Sync word (SelectMAP only).
3. Reset CRC
4. Set Frame Size if initial configuration.
5. Set COR.
6. Set MASK.
7. Set CTL. (Set PERSIST if you want to keep SelectMAP active after this configuration.)
8. Set CCLK.

Reading Configuration

These commands can be issued to read a full configuration or for a partial configuration after the chip has been completely configured.

1. Issue a Sync word (SelectMAP only) if the previous configuration command was aborted.
2. Set the FAR to the starting address.
3. Issue a RCFG (read configuration) command to the CMD register.
4. Write the number of words to be read to the FDRO register.
5. Flush the command pipeline with a pad word.
6. Read data frames.

Writing Configuration

These commands can be issued either as part of an initial configuration or for a partial configuration after the chip has been configured.

1. Issue a Sync word (SelectMAP only) if the previous configuration command was aborted.
2. Set the FAR to the starting address.
3. Issue a WCFG (write configuration) command to the CMD register.
4. If the frames being written can cause contention, then assert the GHIGH_B signal.
5. Write the number of words to be written to the FDRI register.
6. Write data frames.
7. If the GHIGH_B signal was asserted, de-assert it and write one pad frame.

Type			OP		Register Address																Word Count										
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
0	0	1	x	x	0	0	0	0	0	0	0	0	0	0	x	x	x	x	0	0	x	x	x	x	x	x	x	x	x	x	x

Notes: Locations within fields containing a zero or one must have these values An *X* in a bit field indicates that the value is variable and must be set.
Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 11: Command Header Format

Type			OP		Word Count																										
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
0	1	0	x	x	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Notes: Locations within fields containing a zero or one must have these values An X in a bit field indicates that the value is variable and must be set.
Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 12: Large Block Count Header Extension Format

Configuration Registers

Configuration logic is accessed and controlled via a collection of 32-bit registers called the Configuration Registers (see Table 8). Registers are described in the following sections.

Table 8: Configuration Register Addresses

Register Name	Mnemonic	R/W	Binary Address
CRC	CRC	R/W	0000
Frame Address	FAR	R/W	0001
Frame Data Input	FDRI	W	0010
Frame Data Output	FDRO	R	0011
Command	CMD	R/W	0100
Control	CTL	R/W	0101
Control Mask	MASK	R/W	0110
Status	STAT	R	0111
Legacy Output	LOUT	W	1000
Configuration Option	COR	R/W	1001
Reserved	—	—	1010
Frame Length	FLR	R/W	1011
Reserved	—	—	1100
Reserved	—	—	1101
Reserved	—	—	1110
Reserved	—	—	1111

Command Register (CMD)

The content of the Command Register (CMD) is interpreted by the configuration state machine. Configuration commands control the operation of the configuration state machine, the Frame Data Register (FDR), and some of the global signals. The command in the Command Register is executed each time the FAR is loaded with a new value. The effect of each command is defined in Table 9.

Table 9: Configuration Commands

Cmd	Code	Description
Rsvd	0000	—
WCFG	0001	**Write Configuration Data —** Used prior to writing configuration data to the FDRI. It takes the internal configuration state machine through a sequence of states that control the shifting of the FDR and the writing of the configuration memory. (See “Frame Data Input Register (FDRI)” on page 11).
Rsvd	0010	—
LFRM	0011	**Last Frame —** This command is loaded prior to writing the last (pad) data frame if the last GHIGH_B signal was asserted. This command is not necessary if the GHIGH_B signal was not asserted. This allows overlap of the last frame write with the release of the GHIGH_B signal.
RCFG	0100	**Read Configuration Data —** Used prior to reading frame data from the FDRO. Similar to the WCFG command in its effect on the FDR (see “Frame Data Output Register (FDRO)” on page 11).
START	0101	**Begin Startup Sequence —** Starts the startup sequence. This command is also used to start a shutdown sequence prior to partial reconfiguration. The Startup Sequence begins with the next successful CRC check (see “Cyclic Redundancy Check (CRC)” on page 10).
RCAP	0110	**Reset Capture —** Used when performing capture in single-shot mode. This command must be used to reset the capture signal if single-shot capture has been selected.
RCRC	0111	**Reset CRC —** Used to reset CRC register in the event of an error condition. This command is mainly for testing or troubleshooting (see “Cyclic Redundancy Check (CRC)” on page 10).
AGHIGH	1000	**Assert GHIGH_B Signal —** Used prior to reconfiguration to prevent contention while writing new configuration data. All CLB outputs and signals are forced to a one.
SWITCH	1001	**Switch CCLK Frequency —** Used to change (increase) the frequency of the Master CCLK. The new frequency is specified in the “The Configuration Option Register (COR) is used to select configuration options that are illustrated in Figure 13 and defined in Table 10. Entries in this table are further explained in the following tables.” on page 8.
Rsvd	1010	—
Rsvd	1011	—
Rsvd	1100	—
Rsvd	1101	—
Rsvd	1110	—
Rsvd	1111	—

Configuration Option Register (COR)

The Configuration Option Register (COR) is used to select configuration options that are illustrated in Figure 13 and defined in Table 10. Entries in this table are further explained in the following tables.

	DONE_PIPE	DRIVE_DONE	SINGLE	OSCFSEL						SSCLKSRC		LOCK_WAIT				SHUTDOWN	DONE_CYCLE			LCK_CYCLE			GTS_CYCLE			GWE_CYCLE			GSR_CYCLE		
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
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	x	x

Notes: Locations within fields containing a zero or one must have these values An *X* in a bit field indicates that the value is variable and must be set. Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 13: COR (Configuration of Option Register) Fields

Table 10: Configuration Option Register Fields

Field	Bit Indices	Description	Bitgen Default
DONE_PIPE	30	0: No pipeline stage for DONEIN. 1: Add pipeline stage to DONEIN. FPGA waits on DONE that is delayed by one (1) cycle of the StartupClk rather than the pin itself. Use this option when StartupClk is running at high speeds.	0
DRIVE_DONE	29	0: DONE pin is actively driven high (default). 1: DONE pin is open drain.	0
SINGLE	28	Readback capture is one-shot.	0
OSCFSEL	27:22	Select CCLK frequency in Master Serial configuration mode.	2
SSCLKSRC	21:20	Startup sequence clock source 00: Cclk 01: UserClk 1x: JTAGClk	0
LOCK_WAIT	19:16	4-bit mask indicating which DLL lock signals to wait for during LCK cycle. The 4 bits (from MSB to LSB) correspond to DLLs TL=3, TR=2, BL=1, BR=0. The default is not to wait for any DLL lock signals.	0
SHUTDOWN	15	Indicate whether doing a startup or shutdown sequence. 0: Startup (default) 1: Shutdown sequence	0
DONE_CYCLE	14:12	Startup phase in which DONE pin is released	3
LCK_CYCLE	11:9	Stall in this startup phase until DLL locks are asserted.	7
GTS_CYCLE	8:6	Startup phase in which I/Os switch from tri-state to user design	4
GWE_CYCLE	5:3	Startup phase in which the global write-enable is asserted	5
GSR_CYCLE	2:0	Startup phase in which the global set/reset is negated	5

Table 11 shows the allowed values for the OSCFSEL field of the COR. Setting OSCFSEL to one of these values will set the Master CCLK frequency to the specified value.

Table 11: OSCFSEL-Specified Master CCLK Frequencies

CCLK (MHz)	OSCFSEL	CCLK (MHz)	OSCFSEL	CCLK (MHz)	OSCFSEL
4.3	000010	13	001010	41	100111
5.4	010001	15	001101	45	110011
6.9	000100	20	010111	51	101010
8.1	000101	26	011010	55	110100
9.2	000110	30	011101	60	101101
10.0	000111	34	110010	—	—

Note: These values are accurate to +45%, – 30%.

Table 12 shows the values of the DONE_CYCLE, LCK_CYCLE, GTS_CYCLE, GWE_CYCLE, and GSR_CYCLE fields in COR. This table shows the step in the start-up sequence when each of these signals becomes active.

Table 12: COR Startup Cycle Fields

Field Value	DONE_CYCLE (DONE Active)	GTS_CYCLE (GTS_CFG Inactive)	GSR_CYCLE (GSR Inactive)	GWE_CYCLE (GWE Active)	LCK_CYCLE
000	1	1	1	1	0
001	2	2	2	2	1
010	3	3	3	3	2
011	4 (default)	4	4	4	3
100	5	5 (default)	5	5	4
101	6	6	6 (default)	6 (default)	5
110	—	DoneIn†	DoneIn†	DoneIn†	6
111	Keep State	Keep State	Keep State	Keep State	Don’t Wait (default)

† DONE if DonePipe = No, else the delayed version of DONE.

Control Register (CTL)

The Control Register (CTL) fields are illustrated in Figure 14 and defined in Table 13.

																								SBITS		PERSIST						GTS_USR_B
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**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		x	x	x	0	0	0	0	0	x

Notes: Locations within fields containing a zero or one must have these values An *X* in a bit field indicates that the value is variable and must be set. Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 14: Control Register Fields

Table 13: Control Register Bits

Name	Bit Indices	Description
SBITS	8:7	Security level: 0: Read/Write OK 1: Readback disabled. 2,3: Readback disabled, Writing disabled except CRC register.
PERSIST	6	Configuration port remains after configuration. 0: No (default) 1: Yes
GTS_USR_B	0	Active-low global tristate I/Os. Turn off pull-ups if GTS_CFG_B is also asserted.

Cyclic Redundancy Check (CRC)

A data input error checking mechanism is provided through the Cyclic Redundancy Check (CRC) register. When data is written to any configuration register (except LOUT) a 16-bit CRC value is calculated using both the register data and the address. This value is saved in the CRC register. At the end of any series of writes a pre-calculated CRC blockcheck value may be written to the CRC register. If the resulting value is non-zero, an error is indicated. The CRC_ERROR bit is accessible through the Status Register. If a CRC error is detected, configuration logic is put in the ERROR mode. An algorithm for computing CRC is as follows.

CRC Algorithm

```
/* Initialization */
crc ‹ 0;
skip_pad ‹ true;
more_words ‹ true
/* Skip pad */
while (more_words && skip_pad) {
  w ‹ next_word;
  skip_pad ‹ (w[31:27]  ’00101’
    & w[31:27]  ’00101’);
}
/* Check for write operation. */
do {
  if (w[31:27] = ’00101’) {
    /* A Read OP. Don’t use in CRC*/
    wc ‹ w[10:0];
    if (wc = 0) {
      w ‹ next_word;
      wc ‹ w[20:0];
    }
    while (wc – –  > 0) {
      w ‹ next_word;
    }
  }
  elsif (w[31:27] = ’00110’) {
    /* A Write OP. Use in CRC. */
    addr ‹ w[16:13];
    if (addr Œ {0,1,2,4,5,6,9,D,B}) {
      wc ‹ w[10:0];
      if (wc = 0) {
          /* wc is in next word. */
        w ‹ next_word;
        wc ‹ w[20:0];
      }
      while (wc––  > 0) {
        w ‹ next_word;
        sw ‹ addr, word;
        for (i‹ 0; i<36; i++) {
          x16 ‹ crc[15]  sw[i];
          x15 ‹ crc[14]  x16;
          x2 ‹ crc[1]  x16;
          crc‹ x15,crc[14:2],x2,crc[1:0];
        }
      }
    }
  }
} while (more_words)
```

Frame Address Register (FAR)

The Frame Address Register (FAR) holds the address of the current frame. The address is divided into three parts, the Block Type, the Major Address, and the Minor Address. The Block Type field indicates whether the CLB or Block RAM address space is used. The command in the Command Register is executed each time the FAR is loaded with a new value.

The Major Address selects the CLB or RAM column, the Minor Address selects the frame within the column. The Minor Address is incremented each time a full data frame is read from or written to the Frame Data Register. If the last frame within the CLB column is selected when the increment occurs, the Major Address is incremented and the Minor Address is reset to zero, otherwise the Minor Address is incremented. The Block RAM Major Address is not incremented automatically.

See Figure 15 for the definitions of valid values for the Block Type field. The FAR field definitions are shown in Figure 16.

Type	Codes
CLB	00
RAM	01

Figure 15: Block Type Codes

					Block Type		Major Address (CLB Column)								Minor Address (Frame Address)																
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
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

Notes: Locations within fields containing a zero or one must have these values An *X* in a bit field indicates that the value is variable and must be set.
Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 16: Frame Address Fields (FAR)

Frame Data Input Register (FDRI)

The Frame Data Input Register (FDRI) is used to load configuration frame data into a Virtex chip.

The Frame Data Register (FDR) is a shift register into which data is loaded prior to transfer to the configuration memory. Configuration data is written to the Virtex part by loading the Command Register with the WCFG command and then loading the FDR 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 frame (the pad frame) is always dummy data which is not actually written to the configuration memory. Each frame write must include enough 32 bit data words to load the frame fully. There is one pad word at the end of each frame which is required for the pipelining hardware.

Frame Data Output Register (FDRO)

The Frame Data Output Register (FDRO) is for reading configuration data or captured data from the Virtex chip, a process which is called Readback. Readback is performed by loading the command register with the RCFG command and then addressing the FDRO with a Read Command.

Frame Length Register (FLR)

Near the beginning of the configuration bitstream the Frame Length Register (FLR) is written with the length of a frame, as measured in 32-bit words. This length count is used to provide sequencing information for the configuration read and write operations. Note that the FLR must be written before any FDR operation works. It is not necessary to set the FLR more than once. If the number of bits in a frame is not divisible by 32, the length count of the frame must be rounded up to the next highest integer. The values for the FLR for all the current Virtex devices are given in Table 14.

Note: The FLR contains a value that is one less than the number of words that are read from or written to the device. This is because of the extra word needed for pipelining.

Table 14: Frame Length Register Value

Device	Row × Col	Frame Length	# Frame Words	FLR Value
XCV50	16 × 24	384	12	11
XCV100	20 × 30	448	14	13
XCV150	24 × 36	512	16	15
XCV200	28 × 42	576	18	17
XCV300	32 × 48	672	21	20
XCV400	40 × 60	800	25	24
XCV600	48 × 72	960	30	29
XCV800	56 × 84	1088	34	33
XCV1000	64 × 96	1248	39	38

Legacy Output Register (LOUT)

The Legacy Output Register (LOUT) is used for daisy chaining the configuration bitstream to other Xilinx devices. Data written to the LOUT is serialized and appears on the DOUT pin.

Mask Register (MASK)

The Mask Register (MASK) is a mask register for writes to the CTL register. A one in bit N of the mask allows that bit position to be written in the CTL register. The default value of the mask is all zeros.

Status Register (STAT)

The Status Register (STAT) is loaded with current values of several control or status signals. The register can be read via the reconfiguration block or via JTAG. The fields in the Status register are illustrated in Figure 17. The values of the signals given in Table 15 can be read from the status register.

																	DONE	INIT	MODE			GHIGH_B	GSR_B	GWE_B	GTS_CFG	IN_ERROR	LOCK				CRC_ERROR
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
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	x

Notes: Locations within fields containing a zero or one must have these values An *X* in a bit field indicates that the value is variable and must be set. Heavy vertical lines are used to separate fields. Light vertical lines separate nibbles in the word.

Figure 17: Status Register Fields

Configuration Interface

Table 15: Status Register Bits

Name	Bit Indices	Description
DONE	14	Input from DONE pin
INIT	13	Value of $\overline{\text{INIT}}$
MODE	12:10	Value of M2, M1, M0 mode pins
GHIGH_B	9	0 = Force all interconnect high
GSR_B	8	0 = Reset/Set all flip-flops
GWE_B	7	1 = Prevent writing to flip-flops and Block SelectRAM
GTS_CFG	6	0 = I/Os are tri-stated
IN_ERROR	5	Legacy input error
LOCK	4:1	Output from DLL lock signals. 1 = DLL is locked.
CRC_ERROR	0	Indicates that a CRC error has occurred.

There are two configuration interfaces to the Virtex chips — the bit-serial Boundary Scan interface and the 8-bit byte-serial SelectMAP interface. Conceptually, XCV50 configuration data appears as in Figure 18.

Data Frame 0 (11 words)	Pad Word
⋮	
Data Frame *n* (11 words)	Pad Word
Pad Frame (12 words)	

Figure 18: XCV50 Frame Padding for Reads

Frames and words within frames are written in the same order in both configuration interfaces, starting with Frame 0, word 0 (the left-most in the picture), followed by word 1, etc. Bits within each word are written from left to right (*MSB first*) in the bit-serial configuration interfaces.

Within the SelectMAP interface, data is written a byte at a time. A sample word is shown in Figure 19. The top row indicates the chip pin names. The bottom row indicates the bit indices within a configuration word. Byte 0 loads first, followed by byte 1, *et cetera*. The MSB of each byte (*i.e.*, bits 31, 23, 15, and 7) is loaded on pin D0. The LSB of each byte (*i.e.*, bits 24, 16, 8, and 0) is loaded on pinD7.

Byte 0								Byte 1								Byte 2								Byte 3							
D0	D1	D2	D3	D4	D5	D6	D7	D0	D1	D2	D3	D4	D5	D6	D7	D0	D1	D2	D3	D4	D5	D6	D7	D0	D1	D2	D3	D4	D5	D6	D7

Figure 19: SelectMAP Byte and Bit Ordering

LUT SelectRAM Examples

Several examples of reading and evaluating configuration data are provided to illustrate the following.

Example 1: Read and Write Semaphores in an XCV100 at CLB R1 C1, Slice 0.

Semaphores are a useful communication mechanism documented in XAPP 153, “Status and Control Semaphore Registers using Partial Reconfiguration”. Figure 20 shows an abstraction of a microprocessor writing control information to an FPGA and reading status information. One convention for implementing semaphores in Virtex is to use two bits of a 16-bit, dual-port RAM, as illustrated in Figure 21. This occupies one CLB slice with the F-LUT implementing the Control Semaphore and the Status Semaphore implemented in the G-LUT. Address 15 of the G-LUT is used for on-chip writes to the semaphore and address 14 of the F-LUT is used for on-chip reads. Thus, from an off-chip point-of-view, this is reading the G-LUT[15], and writing F-LUT[14].

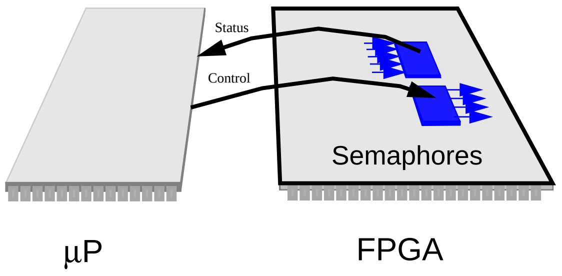

Figure 20: Semaphore Abstraction

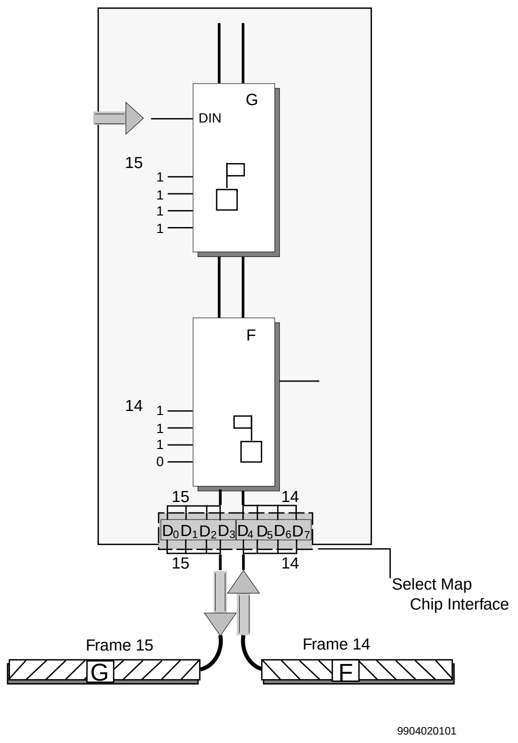

Figure 21: Semaphore Read/Write Implementation

Using a Semaphore Abstraction dual-port RAM ensures that the LUT SelectRAMs are placed in the CLB in a predictable manner. *When writing data to one or more LUT SelectRAMs or flip-flops on the device, all bits in the frame **must** have valid configuration information.* This is assured by altering valid configurations from bitstream files or from frames read from a properly configured Virtex chip. The latter approach is used in this example.

Attributes for this design are summarized in Table 16.

Table 16: Design Attributes for Example 1

	G-LUT [15]	F-LUT [14]	
Attribute	**Read**	**Read**	**Write**
Chip_Rows	20		
Chip_Cols	30		
FL	14		
CLB_Row	1		
CLB_Col	1		
Slice	0		
FG	1	0	
lut_bit	15	14	
RW	1	1	0

From the equations in Table 4 on page 4, the values shown in Table 17 can be calculated.

Table 17: Semaphore Example Variables, Equations, and Values

Variable	Equation	Value(s)		
		G-LUT [14]	F-LUT [15]	
		Read	**Read**	**Write**
MJA	$1 \le 30/2 \Rightarrow 30 - 1 \times 2 + 2$	30		
MNA	lut_bit + 32– 0 × (...)	47	46	
fm_bit_idx	3 + 18 × 1 – FG + RW × 32	52	53	21
fm_st_wd	$14 \times (8 + (30 - 1) \times 48 + \{46,47\}) + RW \times (14 + 1)$ $= 14 \times (1{,}400 + \{46,47\}) + 14 \times RW$ $= 19{,}600 + 14 \times \{46,47\} + 14 \times RW$	20,272	20,258	20,244
fm_wd	floor (20/32)	1	1	0
fm_wd_bit_idx	$31 + 32 \times \{1,1,0\} - \{52,53,21\}$	11	10	10

From off-chip, for reading the G-LUT[15] bit, read one frame (MNA=47). For writing the F-LUT[14] bit, we show how to read then write one frame (MNA=46), as opposed to modifying data from a bitstream file (both are valid methods). The frames on the XCV100 contain 13 32-bit words and one pad word. Remember that fm_st_wd is calculated assuming the entire configuration has been read. However, only the pad frame and then frames 46 and 47 from Major Address 30 are being read. The desired bit is in Frame 1, word #1, which is word 15.

The commands for reading both frames (and the pad frame) are given in Figure 22.

Instruction	Data																																
	Hex	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
Sync Word	AA99 5566	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
Write next (1) word to FAR	3000 2001	0	0	1	1	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	0	1
CLB MJA=30, MNA=46	003C 5C00	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Write next word to CMD	3000 8001	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
Register value for RCFG	0000 0004	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	0
Read from FDRO	2800 602A	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	1	0	1	0
Flush pipe	0000 0000	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	0	0	0
(read 42 words)																																	

Binary data is grouped in two ways for ease of interpretation. Thin vertical lines separate nibble boundaries. Heavy vertical lines separate field boundaries.

Figure 22: Commands to Read Two Data Frames

The 42 words read are shown in Figure 23. F-LUT[14] is in the second frame (frame=1, word=1) at bit 10. It is a physical one (1), but, because the LUT bits are inverted, the logic value is zero (0). G-LUT[15] is in word one (1) of the third data frame (frame=2, word=1) at bit 11. It also is a physical one (1), which is a logic zero (0).

Frame	Word	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	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	G[15]
	2	1	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	
	3	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	
	4	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	
	5	1	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	
	6	1	1	1	0	1	1	0	0	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	
	7	0	0	1	1	1	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	
	8	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	
	9	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
	10	1	0	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	
	11	1	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	
	12	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	
	13	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	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	0	0	0	0	
	1	1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	F[14]
	2	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	
	3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	
	4	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
	5	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	1	1	
	6	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
	7	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	
	8	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
	9	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
	10	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
	11	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	
	12	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	
	13	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 23: Three Frames Containing the Semaphores

To write to the F-LUT semaphore, start with the second data frame (frame=2) that was just read. Words 1–13 of that frame will become words 0 through 12 of the frame to be written. Word 0 of this new frame is a pad word and can have any value. In this new frame, set bit 11 of word 0 (zero) to the desired value of the semaphore. A pad frame must follow the data frame. The commands from Figure 24 write these two frames into the device at the proper location.

Instruction	Data																																
	Hex	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
Sync Word	AA99 5566	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
Write next (1) word to FAR	3000 2001	0	0	1	1	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	0	1
MJA=30, MNA=47	003C 5D00	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0
Write next word to CMD	3000 8001	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
Register value for WCFG	0000 0001	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	0	0	1
Write 28 words to FDRI	2800 401C	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0
Data Word 0	A01A EC00	1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
Data Word 1	FF0F 3FC0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Data Word 2	0FF0 02FC	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0
⋮	⋮																⋮																
Data Word 27	0000 0000	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	0	0	0

Binary data is grouped in two ways for ease of interpretation. Thin vertical lines separate nibble boundaries. Heavy vertical lines separate field boundaries.

Figure 24: Commands to Write a Semaphore Value

Example 2: Reading the Complete Configuration From an XCV50.

Steps:

1. If flip-flop values are needed, strobe the on-chip signal, CAPTURE, to capture flip-flop values. See the Xilinx Libraries Guide for use of the CAPTURE_VIRTEX cell.
2. Write the starting frame address (CLB column 0, row 0) into the FAR.
3. Write the RCFG command to the CMD register.
4. Address the FDRO register with a READ operation and word count equal to the number of 32-bit words in the CLB frames.
5. Read the data following the timing diagrams in the SelectMAP interface section.
6. Write the address for RAM block 0 to the FAR.
7. Address the FDRO register with a read operation and word count equal to the number of 32-bit words in the RAM block.
8. Read the data.
9. Write the address for RAM block 1 to the FAR.
10. Address the FDRO register with a read operation and word count equal to the number of 32-bit words in the RAM block.
11. Read the data

When using SelectMAP mode to read data words from the Virtex chip, de-assert $\overline{\text{CS}}$, de-assert $\overline{\text{WRITE}}$, assert $\overline{\text{CS}}$, then clock the data out. When using JTAG, load the JTAG IR (instruction register) with the CFG_OUT instruction. Then go to the SDR (Shift-DR) state and shift the data out. (See Figure 25)

Instruction	Data																																
	Hex	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
Sync Word	AA99 5566	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1
Write next (1) word to FAR.	3000 2001	0	0	1	1	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	0	1
CLB MJA=0, MNA=0	0000 0000	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	0	0	0
Write next word to CMD register.	3000 8001	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
Register value for RCFG	0000 0004	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	0
Read from FDRO register.	2800 6000	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
15876 words	4800 6E04	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0
Flush pipe	0000 0000	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	0	0	0
(Read 15876 words.)																																	
Write to FAR register.	3000 2001	0	0	1	1	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	0	1
RAM MJA=0, MNA=0	0200 0000	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read 780 words from FDRO reg.	2800 630C	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1	0	0
Flush pipe	0000 0000	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	0	0	0
(Read 780 words.)																																	
Write to FAR.	3000 2001	0	0	1	1	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	0	1
RAM MJA=1, MNA=0	0202 0000	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read 780 words from FDRO reg.	2800 630C	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1	0	0
Flush pipe.	0000 0000	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	0	0	0
(Read 780 words.)																																	

Figure 25: Example 2 - Read Complete Configuration for XCV50

Example 3: Read the Slice 0 G-LUT From CLB R1 C1 From the Complete Configuration of an XCV50.

The commands for reading the bitstream from the Virtex chip are given in "Example 1: Read and Write Semaphores in an XCV100 at CLB R1 C1, Slice 0." on page 14. Using an XCV50 device, the independent attributes are show in Table 18:

Table 18: XCV50 Independent Attributes

Independent Attributes	Values
Chip_Rows	16
Chip_Cols	24
FL	12
CLB_Row	1
CLB_Col	1
FG	1
Slice	0
RW	1

From the equations given earlier in Table 4, calculating the range of values for fm_st_wd indicates that the 13,740th word of the configuration is the starting word of the 12-word (FL=12) frame containing bit 0 of the G-LUT in Slice 0 of CLB R1C1 (Table 19)

Table 19: Variables, Equations and Values for Slice 0 G-LUT

Variables	Equations	Values
MJA	$1 \leq 24/2 \Rightarrow 24 - 1 \times 2 + 2$	24
MNA	$0 \times (\ldots) + \text{lut\_bit} + 32$	0: 32, 1: 33, 2: 34, 3: 35 4: 36, 5: 37, 6: 38, 7: 39 8: 40, 9: 41, 10: 42, 11: 43 12: 44, 13: 45, 14: 46, 15: 47
fm_bit_idx	$3 + 18 \times 1 - 1 + \text{RW} \times 32$	52
fm_st_wd	$12 \times (8 + (24 - 1) \times 48$ $+ [32{:}47]) + 1 \times 12$ $= 12 \times (1112 + [32{:}47])$ $+12$ $= 13{,}356 + 12 \times 32{:}47]$	13,740:13,920
fm_wd	floor(52/32)	1
fm_wd_bit_idx	$31 + 32 - 52$	11

The configuration bits for the given frame are as follows. Bit 0 is in fm_wd=0, at bit #11.

Bits Word	Frame 32-bit Word																																Word #
	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	
13740	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	0	0	0	0
13741	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1
13742	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	2
13743	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	3
13744	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	4
13745	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	5
13746	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	6
13747	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	7
13748	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	8
13749	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	9
13750	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	10
13751	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11

Figure 26: Configuration Bits for Slice 0, CLB R1C1 G-LUT [0]

All 16 LUT SelectRAM bits are in the following words at the same bit index, 11.

Bit Words	Frame 32-bit Word																																LUT Bit
	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	
13741	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
13753	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1
13765	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	2
13777	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	3
13789	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	4
13801	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	5
13813	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	6
13825	0	1	1	0	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	7
13837	1	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	8
13849	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	9
13861	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	10
13873	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0	11
13885	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	12
13897	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	13
13909	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	14
13921	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	15

Figure 27: Location of all 16 LUT SelectRAM Bits

The 16 bits are LUT[0:15]=1111111111111110. The LUT SelectRAM bits are inverted from their logic values. The “logical” contents are LUT[0:15]=0000000000000001. Thus, this G-LUT implements a 4-input AND function.

Example 4: Read the Slice 1 F-LUT from CLB R19 C16 from an XCV100.

Commands for reading the bitstream from the Virtex chip are given in Figure 28. Use the following independent attributes to find the given F-LUT.

Table 20: Virtex Bitstream Command Attributes

Independent Attributes	Values
Chip_Rows	20
Chip_Cols	30
FL	14
CLB_Row	19
CLB_Col	16
FG	0
Slice	1
RW	1

From the equations in Table 4, calculating fm_st_wd indicates the starting word with respect to a configuration that starts at MJA=0, MNA=0. Because the frames we are interested in start at MJA=1, MNA=0, which is fm_st_wd = 126, so the first 125 words are not needed (0 to 124). Therefore, to find the given Slice 1 F-LUT, see Table 21.

Table 21: Variables, Equations, and Values for Slice 1 F-LUT

Variables	Equations	Values			
MJA	16 > 30/2 ⇒ 2×16 – 30 –1	1			
MNA	lut_bit + 32 – Slice × (2 × lut_bit +17) = [0:15] + 32 – (2 × [0:15] + 17) = 15 – [0:15]	0: 15 1: 14 2: 13 3: 12	4: 11 5: 10 6: 9 7: 8	8: 7 9: 6 10: 5 11: 4	12: 3 13: 2 14: 1 15: 0
fm_bit_idx	3 + 18 × 19 – 0+ 32	377			
fm_st_wd	14 × (8 + (1– 1) × 48 + [15:0]) +1 × 14 = 14 × (8 + [15:0]) +14 = 126 + 14 × [15:0]	= 336:126 (w.r.t. (0,0)) = 210:0 (w.r.t. (1,0))			
fm_wd	floor(377/32)	11			
fm_wd_bit_idx	31 + 32 × 11 – 377	6			

Instruction	Data																																
	Hex	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
Sync Word	AA99 5566	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1
Write next (1) word to FAR.	3000 2001	0	0	1	1	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	0	1
CLB MJA = 1, MNA = 0	0002 0000	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	0	0	0	0	0	0
Write next word to CMD.	3000 8001	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
Register value for RCFG	0000 0004	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	0
Read from FDRO.	2800 60E0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0
Flush pipe.	0000 0000	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	0	0	0
(Read 224 words.)																																	

Figure 28: Commands to read Slice 1 F-LUT

Sixteen frames need to be read, one for each bit in the LUT SelectRAM. The bits in LUT SelectRAMs in Slice 1 occur in the opposite order that they do for Slice 0 LUT SelectRAMs.

Frames are read sequentially with ascending addresses. If read in LUT bit order, LUT[0], LUT[1], …, LUT[15]. These are stored in descending addresses which require sixteen separate read operations each reading one data frame and one pad frame. However, if read in ascending address order, LUT[15], LUT[14], …, LUT[0], all sixteen data frames are read with a single read operation. This requires only one pad frame for all sixteen frames. Thus, it takes less time to read ascending frames starting at MJA=1, MNA=0 and finishing with frame MJA=1, MNA=15. The frames in the XCV100 contain 14 32-bit words and a single pad word. Commands for reading only the F-LUT data are given in Figure 28.

The 377th bit of each frame is the bit in the F-LUT. This LUT SelectRAM bit is in the frame’s 11th word, bit 6.

The configuration bits for the given frame are as follows: LUT Bit 15 is in MJA=1, MNA=0, fm_wd=11, at bit #6.

Word	Frame 32-bit Word																															
	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
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	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
5	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
6	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
7	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
8	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1
10	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0
12	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0

Figure 29: Configuration Bits

LUT bit 0 is in MJA=1, MNA=15, fm_wd=11, at bit #7.

Bit stream	Frame 32-bit Word																																FmWd
	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	
210	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	0	0	0	0
211	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
212	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	2
213	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	3
214	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	4
215	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	5
216	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	6
217	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	7
218	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	8
219	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	9
220	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	10
221	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	1	0	0	0	0	11
222	0	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	12
223	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123

Figure 30: Frame Containing F-LUT [0]

For the sake of brevity, here are the sixteen 11th words in the order they appear in the bitstream.

Bitstream	Frame 32-bit Word																																LUT Bit
	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	
011	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	15
025	1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0	14
039	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0	13
053	1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0	12
067	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	11
081	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	0	0	0	10
095	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	9
109	1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0	8
123	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	7
137	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0	0	0	0	6
151	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	0	5
165	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	4
179	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	3
193	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0	0	0	0	2
207	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	1
221	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Figure 31: Sixteen Words Containing the F-LUT bit

The bits are $\overline{\text{LUT[15:0]}}$=`0111111111111111`.The LUT SelectRAM bits are inverted from the logic sense. The logical contents are LUT[15:0]=`0000000000000001`. Thus, this F-LUT implements a 4-input AND gate.

Example 5: Read All Bits in Slice 0 G-LUTs From CLB C2 and XCV50.

Given the following attributes, the necessary values to find the G-LUT data can be computed.

Table 22: All Bits: Slice 0 G–LUTs From CLB C2 and XCV50

Independent Attributes	Values
Chip_Rows	16
Chip_Cols	24
FL	12
CLB_Row	1:16
CLB_Col	2
FG	1
Slice	0
lut_bit	0:15
RW	1

So, from equations in Table 4, the dependent variables can be calculated.

Table 23: Dependent Variables, Equations, and Values

Variables	Equations	Values			
MJA	$2 \leq 24/2 \Rightarrow 24 - 2\times 2 + 2$	22			
MNA	lut_bit + 32 – Slice $\times$ (2 $\times$ lut_bit + 17) = lut_bit + 32 – 0 $\times$ (2 $\times$ lut_bit + 17) = [0:15] + 32	0: 32 1: 33 2: 34 3: 35	4: 36 5: 37 6: 38 7: 39	8: 40 9: 41 10: 42 11: 43	12: 44 13: 45 14: 56 15: 47
fm_bit_idx	3 + 18 $\times$ CLB_Row – 1+ 32 = 34 + 18 $\times$ [1:16]	1: 52 2: 70 3: 88 4: 106	5:124 6:142 7:160 8:178	v9: 196 10: 214 11: 232 12: 250	13: 268 14: 286 15: 304 16: 322
fm_st_wd	12 $\times$ (8 + (22 –1) $\times$ 48 + MNA) + 1 $\times$ 12 = 12 $\times$ (1,016 + MNA) + 12 = 12,204 + 12 $\times$ MNA	0: 12,588 1: 12,600 2: 12,612 3: 12,624	4: 12,636 5: 12,648 6: 12,660 7: 12,672	8: 12,684 9: 12,696 10: 12,708 11: 12,720	12: 12,732 13: 12,744 14: 12,756 15: 12,768
fm_wd	floor (fm_bit_idx/32)	1:1 2: 2 3: 2 4: 3	5: 3 6: 4 7: 5 8: 5	9: 6 10: 6 11: 7 12: 7	13: 8 14: 8 15: 9 16: 10
fm_wd_bit_idx	31 + 32 $\times$ fm_wd – fm_bit_idx	1: 11 2: 25 3: 7 4: 21	5: 3 6: 17 7: 31 8: 13	9: 27 10: 9 11: 23 12: 5	13: 19 14: 1 15: 15 16: 29

Note that fm_bit_idx, fm_wd, and fm_wd_bit_idx have 16 values, one for each row on the XCV50. Figure 33 shows where the data lies in the first frame, which contains G[0] for the entire column. The process is the same for the other 15 frames. The commands for reading the LUT SelectRAM data are given in Figure 32.

From the calculation for fm_st_wd, Frame 0 would start at word 12,588 reading the whole configuration. The instructions in Figure 32 start at that word, so the 0th word (ignoring the 12 words in the pad frame) is the same as word 12,588 of the entire CLB configuration. The 12 words in the frame are shown in Figure 33

The LUT SelectRAM bits have been shaded for ease of identification. It can be seen from the calculation of fm_bit_idx that the LUT SelectRAM bits are in the order 0:15. For example, from the above calculations for fm_wd and fm_wd_bit_idx, G-LUT[0] in R1C2 is in fm_wd 1, fm_wd_bit_idx 11. This bit is the shaded bit in word 1 at bit index 11. Similarly, the G-LUT[0] for the other fifteen CLB rows are also shaded in this table.

Instruction	Data																																
	Hex	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
Sync Word	AA99 5566	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1
Write next (1) word to FAR.	3000 2001	0	0	1	1	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	0	1
CLB MJA=22, MNA=32	0002 C800	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write next word to CMD.	3000 8001	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
Register value for RCFG	0000 0004	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	0
Read from FDRO.	2800 60D0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0
Flush pipe.	0000 0000	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	0	0	0
(Read 224 words.)																																	

Figure 32: Commands to read R*C2.S0 G-LUT

Word	Frame 32-bit Word																																CLB Row
	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	
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	0	0	0	0	
1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	1, 2
3	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	3, 4
4	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	5
5	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	6, 7
6	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	8, 9
7	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	10, 11
8	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	12, 13
9	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	14
10	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	15
11	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 33: Frame for R*C2.S0 G-LUTs, Bit G[0]

Glossary

Block SelectRAM	One of several large, fully-synchronous, dual-ported memories in the Virtex FPGAs. Each of these memories contain 4096 bits. The organization of each memory is configurable. The Block SelectRAM complements the smaller, distributed, LUT SelectRAMs.
Boundary Scan Port	One of the configuration ports on the Virtex chip. This is a bit-serial port. The Boundary Scan Port is also known as the JTAG port. Also see the SelectMAP port.
Captured Data	The Flip-Flop data saved from the logic cells into the bitstream. Use the CAPTURE_VIRTEX primitive in your HDL code to specify the trigger and clock for the capture operation.
CLB	See Configurable Logic Block.
Configurable Logic Block	The functional elements for constructing logic circuits. The Virtex CLB is made up of Slices, which contain Logic Cells.
Configuration Bitstream	Configuration commands, optionally with configuration data.
Configuration Commands	Instructions for the Virtex chip. There are two classes of Configuration Command — Major and Minor. The Major Commands read and write data to configuration registers in the Virtex Chip. The Minor commands instruct the Virtex configuration logic to perform specific functions. (See the Command Register description on page 11.)
Configuration Data	Bits that directly define the state of programmable logic. These are written to a Virtex chip in a configuration bitstream, and read as Readback Data from a Virtex chip.
Configuration Frame	The configuration bits in a Virtex chip are organized in columns. A column of CLBs with the I/O blocks above and below the CLBs contain 48 frames of configuration bits. The smallest number of bits that can be read or written through the configuration interfaces is one frame.
Configuration Port	A logical port on the Virtex chip through which configuration commands and data can be read and written. A port consists of one or more physical Device Pins.
Configuration Readback	The operation of reading Configuration Data (also known as Readback Data) from a Virtex chip.
Device Pin	One of the electrical connections on the package containing the Virtex chip.
Frame	See Configuration Frame.
Logic Cell (LC)	The basic building block of the Virtex CLB. An LC includes a 4-input function generator, carry logic, and a storage element.
LUT SelectRAMs	Shallow RAM structures implemented in CLB Lookup Tables (LUTs). See also Block SelectRAM.
Pad	Pad bits are extra bits used to make the total number of bits in a frame an integral multiple of 32, the number of bits in a configuration word. A Pad Word is an extra word used at the end of a Configuration Frame for pipelining. A Pad Frame is an extra Configuration Frame used at the beginning of a Configuration Readback and at the end of a Configuration Write for pipelining.
Readback Data	Configuration data read from a Virtex chip. The data is organized as Configuration Frames.
SelectMAP Port	One of the configuration ports on the Virtex chip. This is a byte-serial port. The pins in the SelectMAP port may be used as User I/O after configuration has been completed or remain configured as a configuration port.
Slice	A subdivision of the Virtex CLB. There are two, vertical, slices in a Virtex CLB. Each slice contains two Logic Cells.
Sync Word	A 32-bit word with a value that is used to synchronize the configuration logic.

Revision History

Date	Revision #	Nature of Change
6/17/99	Version 1.0	Initial Release
7/27/99	Version 1.1	Updated the following tables: Table 5: equation for MNA; Table 7: equations for MJA & fm_bit_idx.
9/30/99	Version 1.2	Changes in page 5. Most formulae in Table 7 have been revised. Documented pad capture value P. Corrected bit locations in Examples.