XAPP660 (v1.0) January 13, 2003

Partial Reconfiguration of RocketIO Pre-emphasis and Differential Swing Control Attributes

Author: Derek R. Curd

Summary

This application note describes a pre-engineered solution for Virtex-II Pro devices using the IBM PowerPC™ 405 core to perform a partial reconfiguration of the RocketIO™ Multi-gigabit Transceivers (MGTs) pre-emphasis and differential swing control attributes. This solution is ideal for applications where these attributes must be modified to optimize the MGT signal transmission for various system environments while leaving the rest of the FPGA design unchanged. The hardware and software elements of this solution can be easily integrated into any Virtex-II Pro design. The associated reference design supports the following devices: XC2VP4, XC2VP7, XC2VP20, and XC2VP50.

The design discussed in this document uses the PPC405 core device control register (DCR) bus interface to implement a simple solution with a minimum of FPGA resources. Application note **XAPP662**, documents the partial reconfiguation of MGT attributes using an IPIF interface to a PLB or OPB bus. This can be a preferred solution for designs already incorporating one of these bus structures. It also provides for a terminal interface using a serial port connection, allowing MGT attribute settings to be changed through command line entries. Design modules are also included to facilitate bit-error-rate tests (BERT) and PRBS diagnostics.

Introduction

Virtex-II Pro devices contain between four and 20 RocketIO MGTs for the creation of high-speed serial links between devices (up to 3.125 Gb/s per channel). The MGTs have four levels of pre-emphasis and five levels of differential swing control selectable by RocketIO primitive attributes. These attributes can be modified to optimize the transmission of the high-speed serial signals. For more details on these attributes or the MGTs in general, please refer to the "RocketIO User Guide."

Variations in the system environment can impact the transmission characteristics of the high-speed serial signals. In these situations it can be beneficial to modify the pre-emphasis or swing control levels for one or all of the MGTs on a Virtex-II Pro device while leaving the rest of the FPGA design unchanged. For example, when using the MGTs to create high-speed serial links across a backplane, the distance the signals must travel can change significantly depending on the slot position of the board. Adjusting the pre-emphasis level to compensate for the change in distance allows for the highest quality signal transmission at the intended baud rate.

This application note demonstrates a partial reconfiguration controller using the PPC405 core to modify the MGT pre-emphasis (TX_PREEMPHASIS) and differential swing control (TX_DIFF_CTRL) attributes based on a port (or slot) identification code provided to the FPGA. After power-up and initial configuration with a base FPGA design, the PPC reads the 4-bit Port ID and modifies the MGT pre-emphasis and swing control attributes as necessary through the internal configuration access port (ICAP). The predefined MGT attribute settings for each Port ID (up to 16 ports in the current implementation) are set with simple modifications to a "C" file array structure. No partial bitstreams or external devices are required to implement this solution.

The body of this document, **Using the MGT Partial Reconfiguration Controller**, is a guide for incorporating the partial reconfiguration controller into a Virtex-II Pro design without making significant modifications to the hardware or software modules provided in the associated reference design. It provides the information necessary to understand the basic operation of the partial reconfiguration controller and to customize the MGT attribute settings for each MGT/Port ID combination. No detailed knowledge of the PPC405 core or C programming is required to use the pre-engineered solution in this manner.

Using the MGT Partial Reconfiguration Controller

Reference Design Environment

Verilog and VHDL versions for Revision 1.0 of the MGT partial reconfiguration controller reference design can be downloaded from the Xilinx FTP site:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp660.zip.

The design was constructed in the Virtex-II Pro Platform FPGA Developers Kit (V2PDK, March 2002 release) environment. Install the V2PDK prior to working with the reference design. VHDL users will also need to have Service Pack 1 (SP1) of the V2PDK installed to ensure full VHDL support.

To use the reference design, extract the **xapp660.zip** file into the V2PDK installation directory (normally $XILINX/V2PDK/). Both Verilog and VHDL versions of the design will be installed and a new design sub-directory created using the following path:

⇒ $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0

All the required files for working with the hardware portion of the design are contained in this directory and its sub-directories. All necessary software design files, including the files to define the MGT attribute settings for each Port ID combination, will be installed under the following path:

⇒ $XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0

Please refer to the documentation provided with the V2PDK for additional information regarding the directory structure, design flow, and other details of the V2PDK environment.

Hardware Module Integration

The primary hardware modules of the MGT partial reconfiguration controller design are shown in Figure 1. The following subsections describe each of these modules in detail, including performance specifications and recommended methods for integrating the modules into a new design.

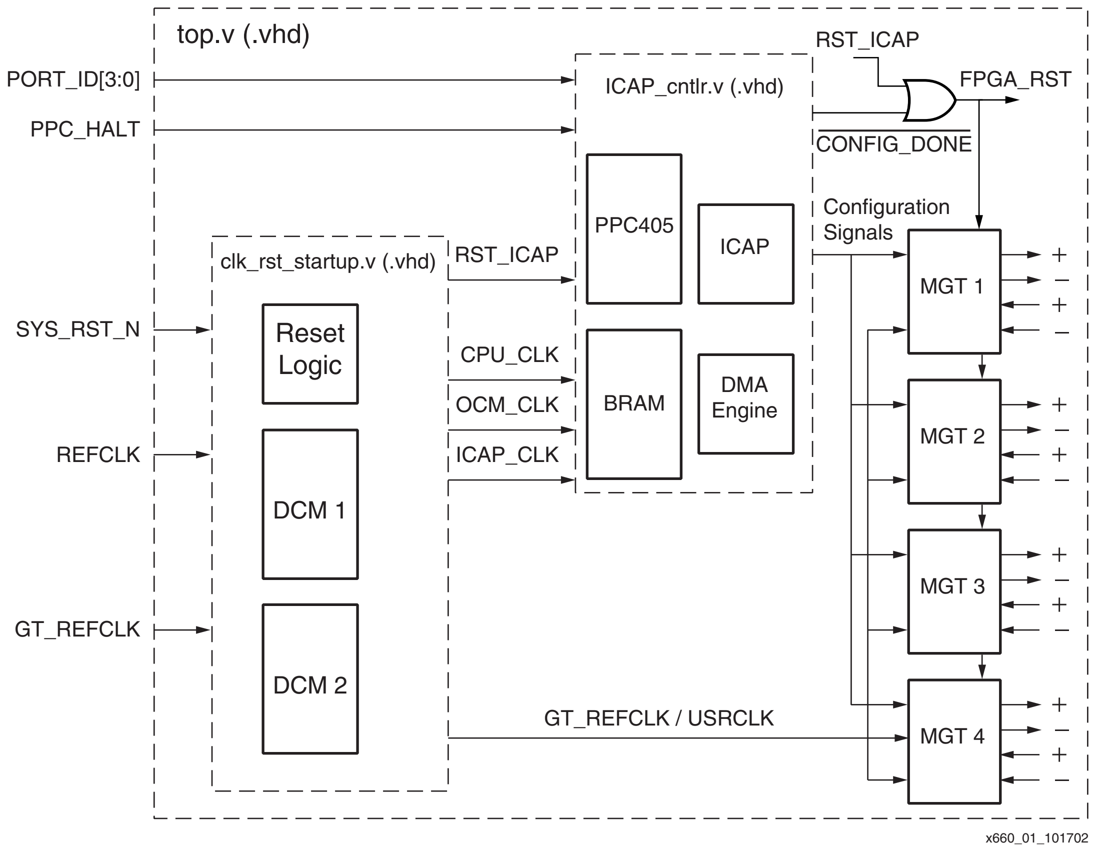

Figure 1: **Primary Hardware Modules**

ICAP Controller Module

The ICAP controller module (Verilog: `ICAP_cntlr.v`, VHDL: `ICAP_cntlr.vhd`) is the main module responsible for controlling the partial reconfiguration of the MGT pre-emphasis and swing control attributes. Though it contains many sub-modules with various functions, it is structured as a single HDL module for easy integration in a new design.

The ICAP controller module contains the PPC405 processor core. This core is responsible for managing all of the partial reconfiguration processes necessary to modify the MGT attributes. The PPC405 polls the PORT_ID[3:0] bus after the power-up and standard configuration operations are complete. Based on the PORT_ID value, the PPC405 manages a READ-MODIFY-WRITE operation on the MGT configuration data to update the user specified pre-emphasis and swing control attribute values for each MGT.

Three 18-Kb block RAMs (BRAMs) are also contained in the ICAP controller module. Two of the BRAMs are used to store the instruction code for the PPC405 core. The third is used to store the PPC405 program data and configuration frame data used to modify the MGT attributes. The data BRAM is used in dual-port mode. One port interfaces to the PPC405 core, while the other port interfaces to the DMA engine. The DMA engine controls the flow of configuration data between the BRAM and the ICAP.

All Virtex-II Pro devices contain an ICAP module. It is a hard-wired module interfacing the user logic with the dedicated configuration logic. The ICAP module allows read and write access to the configuration data frames within the device. The ICAP interface is essentially identical to the SelectMAP interface found on the pins of the FPGA, except it is accessible through the general interconnect inside the device. The opportunity for very powerful self-modifying design techniques exists without the need for external devices or signals.

Table 1 defines the port connections for the ICAP controller module.

Table 1: **ICAP Controller Module Port Connections**

Port	I/O	Description
RST_ICAP	I	ICAP controller module reset (active High) – Resets all logic within *ICAP_cntlr* module, including PPC405 and DMA engine. Does NOT reset any of the device configuration memory or logic.
ICAP_CLK	I	ICAP clock – Clock signal for all logic within *ICAP_cntlr* module with the exception of the PPC405 and BRAMs.
OCM_CLK	I	OCM clock – Clock signal for the instruction and data BRAMs within the *ICAP_cntlr* module.
CPU_CLK	I	CPU clock – Clock signal for the PPC405 core. Note: Must be clocked at a 3:1 ratio with respect to the OCM_CLK.
HALT	I	HALT signal for PPC405 (active Low) – Can be used to temporarily stop the PPC405 from executing instructions. Leave inactive (High) in most applications.
PORT_ID[3:0]	I	Port identification code (4 bits) – Assumed to be a static 4-bit code provided to the FPGA to define the specific backplane slot for the card. Can also be changed dynamically to continuously update the MGT attributes (requires RST_ICAP to be toggled).
CONFIG_DONE	O	Configuration done flag – This output will be asserted to an active High when the *ICAP_cntlr* module finishes a partial reconfiguration operation on the MGTs. It is de-asserted Low when RST_ICAP is active or each time a new partial reconfiguration operation is initiated.

All three clocks (ICAP_CLK, OCM_CLK, and CPU_CLK) need to be in phase and generated to meet both the performance requirements and clock ratio requirements of the ICAP controller module. These requirements are shown in Table 2.

Table 2: **ICAP Controller Clock Requirements**

Clock	Clock Ratio (with respect to OCM_CLK)	Max Frequency (MHz)
ICAP_CLK	Not Critical	35.0
OCM_CLK	1X	Speed Grade Dependent
CPU_CLK	3X	Speed Grade Dependent

In the reference design, the following frequencies are assumed for the clock generation module and timing constraints:

ICAP_CLK	=	33.3 MHz
OCM_CLK	=	100 MHz
CPU_CLK	=	300 MHz.

Other frequencies can be used as long as the ICAP_CLK does not exceed 35.0 MHz, the CPU_CLK:OCM_CLK ratio is exactly 3:1, and both the OCM_CLK and CPU_CLK frequencies meet the timing specifications of the FPGA fabric and PPC405 core for the selected speed grade.

In the reference design, the RST_ICAP input is held active High until eight OCM_CLK cycles after the DCM generating the three specified clocks is locked. This provides stable clock signals to the PPC405 core and other logic in the ICAP controller when the reset is released. Use a similar scheme when integrating the ICAP controller module into a new design.

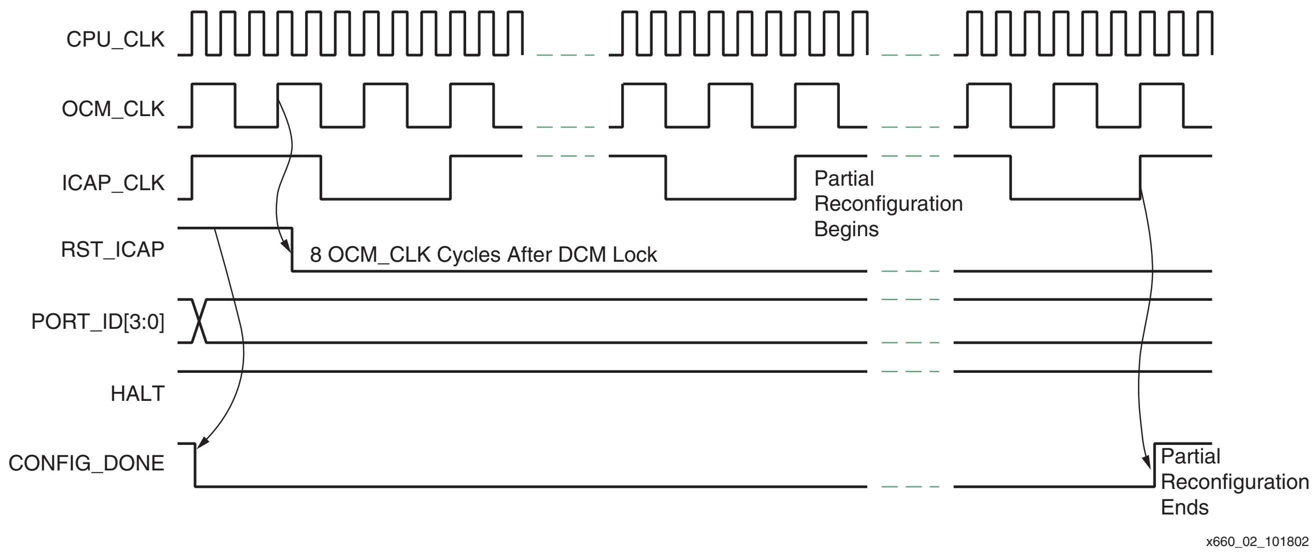
CPU_CLK
OCM_CLK
ICAP_CLK
RST_ICAP
PORT_ID[3:0]
HALT
CONFIG_DONE
8 OCM_CLK Cycles After DCM Lock
Partial Reconfiguration Begins
Partial Reconfiguration Ends
x660_02_101802

Figure 2: **ICAP Controller Module Waveforms**

A typical relationship between the various clocks and other signals of the ICAP controller module as defined in the reference design is shown in Figure 2. The CONFIG_DONE signal is the only output of the module. It indicates when a partial reconfiguration of the MGTs starts and ends. As shown in Figure 1, the complement of the CONFIG_DONE signal can be OR'd together with the RST_ICAP signal to create a reset (e.g. FPGA_RST) for the rest of the FPGA design. This allows normal FPGA operation to be delayed until the standard configuration and partial reconfiguration are completed. For example, in the reference design the FPGA_RST signal is used to hold off operation of the MGTs until the ICAP controller module is finished updating the pre-emphasis and swing control attributes.

The ICAP controller module can be integrated into any Virtex-II Pro design by simply instantiating it into the HDL source code of the design and following the previously mentioned guidelines. Instantiating it in the Top design module with the instance name *ICAP_cntlr0* ensures compatibility with the PPC405 software integration discussed in the **Software Module Integration** section.

Clock/Reset Module

The Clock/Reset module (Verilog: `clk_rst_startup.v`, VHDL: `clk_rst_startup.vhd`) of the reference design is provided to show one method of creating the necessary clock and reset signals for the ICAP controller module. The ICAP_CLK, OCM_CLK, and CPU_CLK signals are generated from a single DCM using the CLKDV, CLK0, and CLKFB outputs, respectively. An input clock source (CLKIN) of 100 MHz is assumed. The LOCKED signal coming from this DCM can also be used to control the logic generating the RST_ICAP signal. After the DCM locks, eight OCM_CLK cycles are counted before RST_ICAP is released (de-asserted Low), ensuring stable clock signals are provided to the ICAP controller module before it starts functioning.

The Clock/Reset module includes a second DCM used to generate the various USRCLK signals for the four MGTs included in the reference design's top level. The GT_REFCLK signal (125 MHz is assumed in the reference design) serves as the primary reference clock for the MGTs, and also acts as the input reference clock for this DCM.

If desired, the Clock/Reset module can be included in a design by simple instantiation into HDL source code. However, the clock and reset signals required for the ICAP controller module and MGT blocks can be derived or borrowed from other portions of the existing design. These signals are to be generated in compliance with the guidelines mentioned in the **ICAP Controller Module** section.

Top Module

The Top module (Verilog: `top.v`, VHDL: `top.vhd`) of the reference design is provided simply to create a complete FPGA design. It includes all the necessary components to demonstrate the capabilities of the ICAP controller module. In addition to the ICAP controller module, the Clock/Reset module, and four MGT blocks are instantiated in the Top module to create a simple system for evaluation of the complete MGT partial reconfiguration controller design. The Top module also includes all the necessary input and output buffer instantiations to interface to the I/O pins of a Virtex-II Pro device.

In the Top module, the *txdata* MGT inputs are hard-wired to transmit two K28.5 control characters repeatedly for demonstration purposes only. As mentioned in the **ICAP Controller Module** section, the reset signal for the MGTs (FPGA_RST) is an OR function of the RST_ICAP and the $\overline{\text{CONFIG\_DONE}}$ signals. This allows both the standard and partial reconfiguration operations to complete before releasing the device for general operation.

Verilog and VHDL test benches are provided with the reference design to simulate the complete system. It can also be targeted at one of the various Virtex-II Pro development boards for hardware evaluation.

Resource Utilization

The MGT partial reconfiguration controller design provides a very powerful mechanism for modifying MGT attributes in response to changing system demands. However, it uses relatively few FPGA fabric (logic and memory) resources because the PowerPC 405 core controls the decision making and configuration data manipulations. This allows the main ICAP controller module to be integrated into nearly any Virtex-II Pro design.

Table 3 shows the FPGA fabric resource utilization for the Virtex-II Pro devices supported by the current release of the reference design. Only the resources required to implement the ICAP controller module are represented. The resources for generating the necessary clock and reset signals are assumed to be shared with other portions of the design.

Table 3: **FPGA Resource Utilization**

ICAP Controller Module Resources		Utilization by Device			
		XC2VP4	XC2VP7	XC2VP20	XC2VP50
Slices Used	67	2.2%	1.4%	0.7%	0.3%
BRAMs Used	3	10.7%	6.8%	3.4%	1.4%

The XC2VP4 and XC2VP7 devices each contain a single PowerPC 405 core. If the PPC405 core is not used already in a design, it can be dedicated solely to manage the MGT attribute updates. Alternatively, the code for performing the partial reconfiguration of the MGT attributes can be added to the primary design code as an initialization routine.

The XC2VP20 and XC2VP50 devices each contain two PowerPC 405 cores. For these devices, one processor core can be dedicated to managing the MGT attribute updates, while the other core runs the primary design code. Although the MGT partial reconfiguration controller design only harnesses a small portion of the power of the PPC405 core, this application is simpler and more cost effective to implement using the processor core rather than standard logic gates.

Software Module Integration

The software modules of the MGT partial reconfiguration controller design are structured to allow easy customizing of the MGT attribute settings for a new design without requiring any significant software programming. Table 4 describes the primary software files specific to this design. All of these files can be found in one of the sub-directories in the following path:

⇒ $XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0

Table 4: **Software Design Files**

File Name	Sub Directory	Description
`MGT_reconfig.c`	`./`	Contains function main ( ) and various sub-functions – Primary body of code used to manage the partial reconfiguration flow for MGT blocks.
`$device_MGT_data.c`	`./data/$device`	Contains the arrays used to define the location of the MGT attributes within the configuration data frames.
`$device_MGT_attributes.c`	*`./attributes/$device`*	Contains the arrays used to allow the user to select the MGT attribute settings for each Port_ID combination.
`includes.h`	`./includes/$device`	Included in all *.c* files. Contains *macros* and *defines* for all modules, as well as *global variable* declarations.
`$device_MGT_data.h`	`./includes/$device`	Contains device specific *defines* and *array* declarations for configuration data frame mapping.
`$device_MGT_attributes.h`	`./includes/$device`	Contains device specific *array* declarations for defining MGT attribute settings.

Notes:
1. `$device` = 2VP4, 2VP7, 2VP20, or 2VP50

Note that the *attributes*, *data*, and *includes* sub-directories all contain additional sub-directories for each of the supported devices (XC2VP4, XC2VP7, XC2VP20, and XC2VP50). During the software build process, the appropriate files for the device specified in the `flow.cfg` file will be automatically incorporated (`flow.cfg` is found in $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0). For example, if *DEVICE = xc2vp4* is specified in the `flow.cfg` file, then the following software design files will be incorporated specifically for the XC2VP4 device:

$XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0/includes/2vp4/includes.h

$XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0/includes/2vp4/2vp4_MGT_data.h

$XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0/includes/2vp4/2vp4_MGT_attributes.h

$XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0/data/2vp4/2vp4_MGT_data.c

$XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0/attributes/2vp4/2vp4_MGT_attributes.c

Several firmware files must also be included with these design files to complete the software module: `boot.S`, `eabi.S`, `crt0.S`, and `ppc-asm.h`. The details of these files are beyond the scope of this document and need not be understood to successfully integrate the software module into a new design. These files are automatically included by the *Makefile* found in the main software directory for the design:

⇒ $XILINX/V2PDK/source/sw/apps/MGT_reconfig_rev1.0

When integrating the software module into a new design, the majority of the design files listed in Table 4 do not require any modification. Only the `$device_MGT_attributes.c` (e.g., 2vp4_MGT_attributes.c) file needs to be updated to define MGTs to be partially reconfigured and to assign the pre-emphasis and swing control settings for each PORT_ID[3:0] value. The details of the necessary modifications are discussed in the **Modifying the MGT Attribute Values** section.

In some situations, it can also be necessary to make minor modifications to the `$device_MGT_data.h` (e.g., `2vp4_MGT_data.h`) file. This file contains a *#define* declaration for the identifier CONFIG_OPS. During initial configuration of the FPGA, the configuration option register (COR) is loaded with various user selectable options from bitstream generation (i.e., BitGen options). Prior to beginning the partial reconfiguration of MGT attributes, the MGT partial reconfiguration controller design overwrites the COR with the value of the CONFIG_OPS identifier to ensure that CRC checking is disabled. This is necessary because the reference design does not calculate a CRC value in association with the partial reconfiguration operations. When the CONFIG_OPS value is written to the COR, default values are assigned to various other user options. The option settings defined by the current CONFIG_OPS value are:

CRC:	Disable
DCMShutdown:	Disable
DonePipe:	No
DriveDone:	No
ConfigRate:	4
StartUpClk:	CClk
DONE_cycle:	4
Match_cycle:	NoWait
LCK_cycle:	NoWait
GTS_cycle:	5
GWE_cycle:	6

If any of these settings will prevent correct operation of the FPGA or system following completion of the partial reconfiguration operations, the value of the CONFIG_OPS identifier should be changed accordingly. Otherwise, there is no need to modify the `$device_MGT_data.h` file. Refer to Chapter 3 of the Virtex-II Pro Platform FPGA handbook for more information on setting the configuration option register values. The CRC setting must be left as *Disable*, even if other options are modified.

Modifying the MGT Attribute Values

The `$device_MGT_attributes.c` file is the only file requiring modification to customize the MGT pre-emphasis and swing control attributes for a new design. This file defines a number of arrays for the user to select the MGTs undergoing partial reconfiguration as well as the attribute settings to apply to those MGTs. The user-defined section of this file is shown below, using the XC2VP4 device as an example. The XC2VP4 device has four MGT blocks.

Excerpt from file: *2vp4_MGT_attributes.c*

```
/*************************************************************************/
/*****************  USER DEFINED SECTION BEGINS HERE*********************/
/*************************************************************************/
/* Set MGTs to be reconfigured to "YES" - otherwise "NO" */
INT8U   MGT_Used[NUM_MGTS] =         { YES,    /* MGT_X0Y0 */
                                       YES,    /* MGT_X0Y1 */
                                       YES,    /* MGT_X1Y0 */
                                       YES };  /* MGT_X1Y1 */
/* Set Swing Control and Pre-emphasis values for each Port/MGT combination */
/* Swing Control Values */
/* For 400mV setting - Enter "4" */
/* For 500mV setting - Enter "5" */
/* For 600mV setting - Enter "6" */
/* For 700mV setting - Enter "7" */
/* For 800mV setting - Enter "8" */
/*                                            M  M  M  M   */
/*                                            G  G  G  G   */
/*                                            T  T  T  T   */
/*                                            X  X  X  X   */
/*                                            0  0  1  1   */
/*                                            Y  Y  Y  Y   */
/*                                            0  1  0  1   */
INT8U MGT_S_Values[NUM_PORTS][NUM_MGTS]  =  { {4, 8, 7, 6},   /* Port 0 */
                                              {5, 4, 8, 7},   /* Port 1 */
                                              {6, 5, 4, 8},   /* Port 2 */
                                              {7, 6, 5, 4},   /* Port 3 */
                                              {4, 8, 7, 6},   /* Port 4 */
                                              {5, 4, 8, 7},   /* Port 5 */
                                              {6, 5, 4, 8},   /* Port 6 */
                                              {7, 6, 5, 4},   /* Port 7 */
                                              {4, 8, 7, 6},   /* Port 8 */
                                              {5, 4, 8, 7},   /* Port 9 */
                                              {6, 5, 4, 8},   /* Port 10 */
                                              {7, 6, 5, 4},   /* Port 11 */
                                              {4, 8, 7, 6},   /* Port 12 */
                                              {5, 4, 8, 7},   /* Port 13 */
                                              {6, 5, 4, 8},   /* Port 14 */
                                              {7, 6, 5, 4} }; /* Port 15 */
/* Pre-emphasis Values */
/*                                            M  M  M  M   */
/*                                            G  G  G  G   */
/*                                            T  T  T  T   */
/*                                            X  X  X  X   */
/*                                            0  0  1  1   */
/*                                            Y  Y  Y  Y   */
/*                                            0  1  0  1   */
```

```
INT8U MGT_P_Values[NUM_PORTS][NUM_MGTS]   =  { {0, 0, 0, 0},    /* Port 0 */
                                               {0, 0, 0, 0},    /* Port 1 */
                                               {0, 0, 0, 0},    /* Port 2 */
                                               {0, 0, 0, 0},    /* Port 3 */
                                               {1, 1, 1, 1},    /* Port 4 */
                                               {1, 1, 1, 1},    /* Port 5 */
                                               {1, 1, 1, 1},    /* Port 6 */
                                               {1, 1, 1, 1},    /* Port 7 */
                                               {2, 2, 2, 2},    /* Port 8 */
                                               {2, 2, 2, 2},    /* Port 9 */
                                               {2, 2, 2, 2},    /* Port 10 */
                                               {2, 2, 2, 2},    /* Port 11 */
                                               {3, 3, 3, 3},    /* Port 12 */
                                               {3, 3, 3, 3},    /* Port 13 */
                                               {3, 3, 3, 3},    /* Port 14 */
                                               {3, 3, 3, 3} };  /* Port 15 */
```

The user defines the MGTs selected for partial reconfiguration with the *MGT_Used* array. The array is edited with a YES for MGTs slated to be reconfigured with new attribute settings. The array is edited with a NO for MGTs either not used or not intended to be modified using partial reconfiguration.

The differential swing control settings are defined using the *MGT_S_Values* array for each MGT at each PORT_ID[3:0] value. When the ICAP controller module performs the partial reconfiguration operation following standard configuration, these new attribute settings will overwrite the previously loaded base FPGA design settings. As described in the attribute file, the array is edited with the first digit of the desired swing control value (e.g., enter a "4" for a 400 mV swing control setting, a "5" for a 500 mV swing control setting, etc.).

The *MGT_P_Values* array also allows the pre-emphasis settings to be user defined for each MGT at each PORT_ID[3:0] value. This array is edited with the desired pre-emphasis setting for each case (enter a "0" for a 10% pre-emphasis, a "1" for a 20% pre-emphasis, etc.).

Table 5 and Table 6 summarize the settings allowed for the swing control and pre-emphasis attribute arrays, respectively.

Table 5: **Differential Swing Control Attribute Settings**

***MGT_S_Values* Array Setting**	**TX_DIFF_CTRL Setting**
4	400 mV
5	500 mV
6	600 mV
7	700 mV
8	800 mV

Table 6: **Pre-emphasis Attribute Settings**

***MGT_P_Values* Array Setting**	**TX_PREEMPHASIS Setting**
0	10%
1	20%
2	25%
3	33%

The MGTs are identified by their physical location on the device (e.g. *MGT_X0Y0*). Care should be taken to lock the position of each MGT instance to a specific physical MGT block using the User Constraints File (e.g. `top.ucf`). This allows a consistent association between the settings defined in the attribute file and a specific MGT. If the MGT instances are not locked to specific physical MGT blocks, the FPGA implementation tools can choose to assign an MGT instance to a different MGT block each time a new revision of the design is implemented. This would result in a loss of the association between the MGT and its specific attribute settings.

Once all the edits are made to the `$device_MGT_attributes.c` file, the software executable can be generated and incorporated into a BRAM initialization file for either HDL simulation or device configuration. Details of the design flow steps required to implement these operations are covered in the **Design Flow and Implementation Details** section.

Design Flow and Implementation Details

The design is ready for simulation and implementation once the ICAP controller module (and any other required hardware modules) is integrated into the HDL source code of a new design and the attribute settings are edited in the `$device_MGT_attributes.c` file. The V2PDK design flow is used for the MGT partial reconfiguration controller reference design.

Edits to *flow.cfg*

The `flow.cfg` file is the master file for controlling the design flow (`flow.cfg` is found in *$XILINX/V2PDK/platforms/MGT_reconfig_rev1.0*). It allows the user to specify information regarding tools, tool options, design software files, simulation, and implementation. For further information on the `flow.cfg` file, see Volume 4: Design Flow of the V2PDK Documentation.

The `flow.cfg` file in the reference design is setup to use Verilog as the HDL language. VHDL users will need to edit the SIM_TOP variable to *cfg_testbench* and edit the HDL variable to *vhdl.*

The `flow.cfg` file can also be edited to change HDL simulator tools and synthesis tools as necessary. In addition, the DEVICE, PACKAGE, and SPEEDGRADE variables should be modified as appropriate for new designs. Finally, the SW_MAKE and SW_ELF variables should be updated to reflect any changes to the software directory structure.

Functional Simulation

Verilog and VHDL test benches are provided for functional simulation of the MGT partial reconfiguration controller reference design. The test benches are available at:

$XILINX/V2PDK/platforms/MGT_reconfig_rev1.0/sim/testbench/verilog

or

$XILINX/V2PDK/platforms/MGT_reconfig_rev1.0/sim/testbench/vhdl

The design can be easily simulated within the V2PDK environment. Open a V2PDK shell to the $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0 directory and then type:

prompt> *make func_sim*

This will start a functional simulation using the PPC405 and MGT SWIFT models. The *data2bram* utility converts the software executable file (`MGT_reconfig.elf`) into a BRAM initialization file. This file is used to initialize the PPC405 instruction and data BRAM contents for simulation.

A functional simulation using the bus functional models can also be created by editing the PPC_MODEL variable in the `flow.cfg` file to "bfm" before running the make *func_sim* command. However, due to the relatively small amount of code being executed by the PPC405 and the small size of the design overall, the performance of the SWIFT model based simulations is generally sufficient. For more details on running a bus functional model simulation, please refer to the V2PDK documentation.

No simulation model is available for the ICAP module hard-wired into all Virtex-II Pro devices. A dummy module of the ICAP_VIRTEX2 block is included in both the Verilog and VHDL versions of the reference design for simulation purposes only. In the Verilog version of the design, an *ICAP_VIRTEX2_SIM* module is instantiated into the ICAP controller module for simulation only. In the VHDL version, a separate `ICAP_VIRTEX2.vhd` module file is compiled only for simulation (see the src.lst file in the $XILINX/V2PDK/MGT_reconfig_rev1.0/sys/vhdl directory). The synthesis tools will ignore the dummy modules and treat the *ICAP_VIRTEX2* module as a black box. It is then replaced by the FPGA implementation tools with the hard-wired ICAP block.

The dummy ICAP module logic emulates the behavior of the real ICAP block by capturing bytes of data from the input port (I[7:0]) during a configuration WRITE command, and by driving the output port (O[7:0]) with an 8-bit binary counter output during a configuration READBACK command. It does not model the actual configuration logic of a Virtex-II Pro device in any way and will not produce valid configuration data frames during READBACK.

Synthesis

Synthesizing the MGT partial reconfiguration controller design, or a new design with the partial reconfiguration hardware and software modules integrated into it, should be straightforward within the V2PDK environment. Simply open a V2PDK shell to the $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0 directory and then type:

prompt> *make synth*

The ICAP module will be passed as a black box through synthesis. If any portion of the ICAP dummy module logic is synthesized, incorrect behavior will result when the design is implemented.

FPGA Implementation

The FPGA implementation flow integrates the hardware and software portions of the reference design to target an actual Virtex-II Pro device. Implementation is initiated by opening a V2PDK shell to the $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0 directory and then typing:

prompt> *make fpga*

This command will first create the software executable file (`MGT_reconfig.elf`) and then use the *data2bram* utility to create a BRAM initialization file (Verilog: `bram_init.ucf`, VHDL: `bram_init_vhdl.ucf`). This file will be automatically appended to the `top.ucf` file found in the sub-directory: $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0/par/v2p. During place and route, the BRAM contents in the `top.ucf` (or `top_vhdl.ucf`) file are used to initialize the PPC405 instruction and data BRAMs, effectively integrating the software code into the hardware design.

The `top.ucf` file contains simple timing constraints for the reference design as well as the necessary statements to *lock* the MGT instances to specific physical MGT blocks. As mentioned earlier, this is necessary to ensure correct association between the MGT attribute settings in software and the physical MGT blocks. In addition, pin constraints contained in this file can be un-commented to target the XC2VP4 Development Board available from Insight (**www.insight-electronics.com**).

The `bram_init.bmm` or `bram_init_vhdl.bmm` file found in the $XILINX/V2PDK/platforms/MGT_reconfig_rev1.0/sys sub-directory is used by the *data2bram* utility to determine the address map for the software design and the hierarchical path to the instruction and data BRAMs. This file assumes that the ICAP controller module is instantiated at the *top* level with an instance name *ICAP_cntlr0*. If this is not the case, the `bram_init.bmm` file needs to be updated to reflect the correct hierarchical path.

The *make fpga* command will result in fully routed design that can then be turned into a bitstream by running:

prompt> *make bit*

The resulting bitstream file (e.g., `top.bit`) is ready for downloading to a Virtex-II Pro device.

Design Operation

After the standard configuration and startup routines have completed, the ICAP controller module will partially reconfigure the MGT pre-emphasis and swing control attributes based on the PORT_ID[3:0] value and the pre-defined attribute settings in the `$device_MGT_attributes.c` file.

In the typical application, it is assumed that the PORT_ID value is hard-wired or static. However, if the PORT_ID value will be changing during operation, the RST_ICAP input to the ICAP controller module can be toggled to generate another partial reconfiguration operation. The PORT_ID value should be changed while the RST_ICAP input is active (High) to ensure that the PPC405 is not trying to read the PORT_ID value while it is changing. Releasing RST_ICAP Low will then initiate a new partial reconfiguation of the MGT attributes.

The CONFIG_DONE flag while go active (High) when the partial reconfiguration of the MGTs is complete. The time it takes to complete the partial reconfiguration operations is dependent on how many MGTs are being reconfigured and the frequency of the three ICAP clocks. In general, however, this time is relatively small compared to the standard configuration routine.

Conclusion

This application note has shown how to incorporate a pre-engineered solution for Virtex-II Pro devices to partially reconfigure the MGT pre-emphasis (TX_PREEMPHASIS) and differential swing control (TX_DIFF_CTRL) attributes using a PPC405 based controller module. The design uses a minimum of FPGA fabric resources and can, therefore, be easily integrated into nearly any Virtex-II Pro based application. Though the reference design was developed within the V2PDK environment, the hardware and software modules required to implement the partial reconfiguration functionality can be integrated into other design flows, as long as the design guidelines discussed in this document are followed. Figure 3 shows the general design flow for hardware and software module integration.

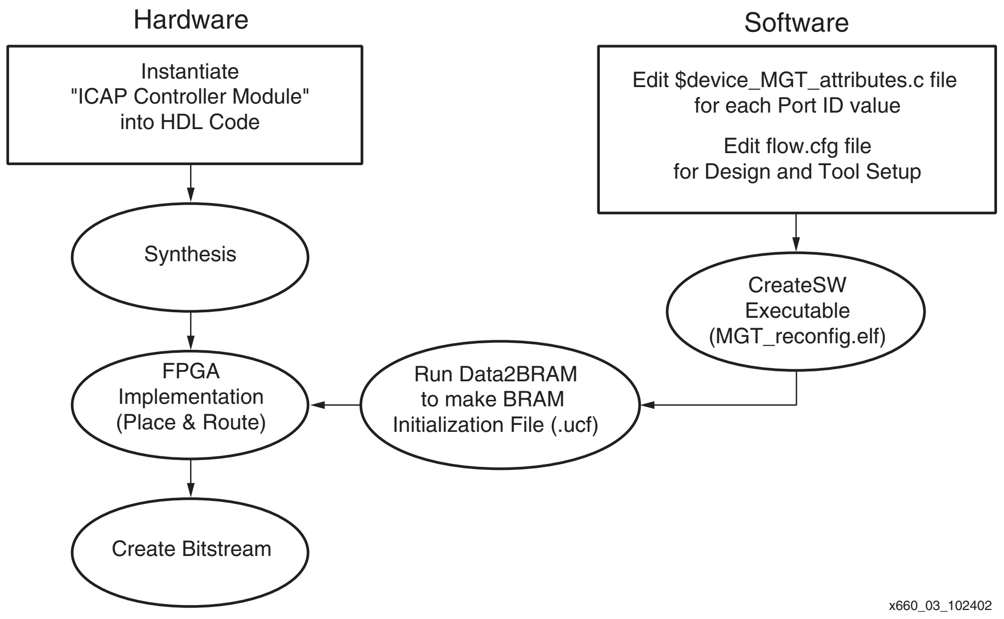
Hardware
Software
Instantiate "ICAP Controller Module" into HDL Code
Synthesis
FPGA Implementation (Place & Route)
Create Bitstream
Edit $device_MGT_attributes.c file for each Port ID value
Edit flow.cfg file for Design and Tool Setup
CreateSW Executable (MGT_reconfig.elf)
Run Data2BRAM to make BRAM Initialization File (.ucf)
x660_03_102402

Figure 3: **Hardware / Software Integration Flow**

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/13/03	1.0	Initial Xilinx release.