XAPP517 (v1.0) December 2, 2011

Dual Use of ICAP with SEM Controller

Author: John Ayer Jr.

Summary

This application note includes a method for sharing an internal configuration access port (ICAP) between the user design and the soft error mitigation (SEM) controller in the Spartan®-6 and Virtex®-6 devices. There are various reasons why the user might want to share the ICAP between the SEM controller and other functions in the design. The reference design provided with this document uses an example of reading back the device IDCODE. This method also enables customers to create multi-boot designs that require access to the ICAP.

Introduction

When instantiated in a user design, the SEM controller interfaces with the ICAP to detect, correct, and classify soft errors that occur in the FPGA. The user application might also need to access the ICAP for other reasons. The reference design shows how a user design can share the ICAP with the SEM controller. This design is a proof of concept, showing that the user can successfully interface with the ICAP and return control to the SEM controller without causing adverse effects.

Designers can use these techniques in their larger designs if this type of functionality is needed. The user must ensure that the SEM controller is in the OBSERVATION state, and then move the SEM controller to the IDLE state. After the SEM controller is in IDLE, the user can take control of the ICAP for other uses and return the ICAP to the SEM controller when finished. After this is done, the SEM controller is moved back to the OBSERVATION state. The reference design steps the user through this process on the ML605 and SP605 boards.

Design Overview

The design has one purpose—to illustrate that it is possible to share the ICAP with the SEM controller and other user-determined functions. There are limitations as to what can be implemented with the ICAP. See Limitations and Considerations, page 13, for more information.

The two main components in this example design are:

- The SEM controller
- Some simple logic to read the IDCODE from the device.

The design modifies the provided SEM controller system-level design example to add new ports and instantiate the IDCODE module. Table 1 describes the ports used to implement the design. Most of the outputs are used to monitor the status of the design using the onboard ML605 or SP605 LEDs. These outputs originate from the controller and change at rates not visible to the eye. When the outputs are asserted, the values are latched and driven constantly until cleared using the LED clear pushbutton pb_led_clear. This means that at any time, multiple LEDs might be on, and this illustrates that the controller has passed through or is in one of the indicated states. When targeting the reference design to the SP605 board for the Spartan-6 FPGA, only four LEDs are available to the user. LEDs not available in the SP605 reference design are noted as such in Table 1.

Table 1: **Design Ports**

Name	Direction	Sense	Description
clk	In	High	This is the master clock for the design. It is used for all synchronous logic elements and to clock the ICAP. It has these frequencies: • Virtex-6 FPGA: 66 MHz • Spartan-6 FPGA: 20 MHz
status_heartbeat_led	Out	High	These pulses indicate that the controller heartbeat is active. The controller status_heartbeat output is slowed down in the design so that LED blinking is perceivable. The rate at which the LED blinks depends on the clock frequency.
status_initialization_led	Out	High	The initialization LED signal is active and remains active after the SEM controller goes through initialization. The signal remains High after the controller moves out of the initialization state until cleared. This signal is not available on the SP605 board.
status_observation_led	Out	High	This signal is active while the controller is in the OBSERVATION state. It goes Low after the controller leaves OBSERVATION.
status_correction_led	Out	High	The correction LED signal is active during controller correction of an error or during transition through this controller state if correction is disabled. This signal remains active after correction is completed until cleared. The correction LED signal is not available on the SP605 board.
status_classification_led	Out	High	The classification LED signal is active during controller classification of an error or during transition through this controller state if classification is disabled. This signal remains active after classification is completed until cleared. The classification LED signal is not available on the SP605 board.
status_idle_led	Out	High	This signal is High when the controller is in the IDLE state. It goes Low after the controller leaves IDLE.
icap_sel_led	Out	High	This signal is active while the ICAP is selected for user control. The SEM controller cannot access the ICAP while this signal is High.
idcode_ready_led	Out	High	This signal is active after the `IDCODE` has been successfully read from the ICAP. The signal is not available on the SP605 board.
monitor_tx	Out	Low	This is the serial 8-N-1 transmit data from the example design UART.
monitor_rx	In	Low	This is the serial 8-N-1 receive data to the example design UART.
pb_led_clear	In	High	This signal clears the LED status signals and clears the captured IDCODE value when active.

Table 1: **Design Ports *(Cont'd)***

Name	Direction	Sense	Description
pb_read_idcode	In	High	This signal instructs the IDCODE read submodule to access the ICAP and get the device IDCODE.
pb_icap_sel	In	High	This signal multiplexes control of the ICAP between the SEM controller and the IDCODE read submodule. Each button push toggles this signal. The value out of reset is 0. • 0: ICAP controlled by SEM controller • 1: ICAP controlled by IDCODE submodule

The design works by allowing the user to take control of the ICAP to perform other functions when necessary. During this time, the SEM controller must be in the IDLE state. While in the IDLE state, single event upset (SEU) error detection and correction is suspended. Figure 1 shows the structure of the multiplexing logic and how the ICAP is controlled. The multiplexing logic must be purely combinational.

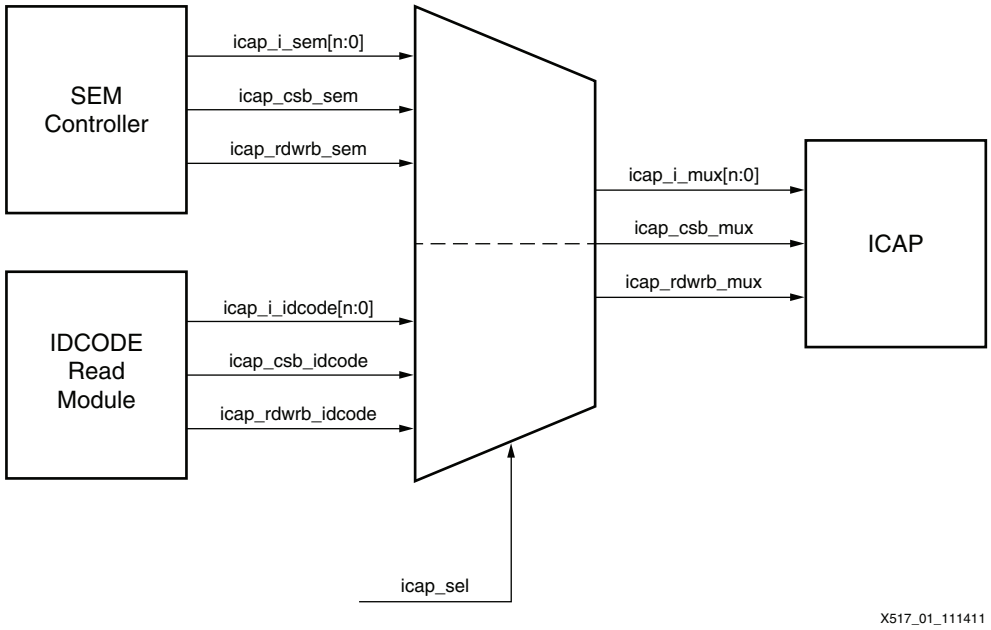


Figure 1: **ICAP Multiplexing Between the SEM Controller and IDCODE Read Submodule**

Note: In Figure 1, n equals 31 for Virtex-6 devices and 15 for Spartan-6 devices.

The multiplexing is done in the sem_example.{v|vhd} file as follows:

```
assign icap_i_mux = icap_sel ? icap_i_idcode : icap_i_sem;
assign icap_csb_mux = icap_sel ? icap_csb_idcode : icap_csb_sem;
assign icap_rdwrb_mux = icap_sel ? icap_rdwrb_idcode : icap_rdwrb_sem;
```

The VHDL equivalent of the above is:

```
icap_i_mux <= icap_i_idcode when icap_sel = '1' else icap_i_sem;
icap_csb_mux <= icap_csb_idcode when icap_sel = '1' else icap_csb_sem;
icap_rdwrb_mux <= icap_rdwrb_idcode when icap_sel = '1' else icap_rdwrb_sem;
```

These steps summarize the actions necessary to transfer control of the ICAP from the SEM controller to a user function and back:

- The SEM controller is in the OBSERVATION state.
- The SEM controller is put in the IDLE state.
- Control of the ICAP is moved to the user function by changing the MUX select.
- The user function issues a SYNC command to the ICAP.
- The user controls the ICAP and performs function-specific requirements (see Limitations and Considerations, page 13, for restrictions).
- The user transfers control back to the SEM controller by changing the MUX select. The user does not issue a DESYNC command to the ICAP.
- The SEM controller is put back into the OBSERVATION state.

IDCODE Read Submodule

This module is used to illustrate that the ICAP can be used for other purposes when control is taken away from the SEM controller. This is not the only function that can be performed, but this is a simple example for this design. Table 2 defines the ports of the `IDCODE` read submodule.

Table 2: **IDCODE Read Submodule Ports**

Name	Direction	Sense	Description
clk	In	High	This signal clocks the main state machine and other synchronous elements.
reset	In	High	When active, this signal resets the module to its initial state.
start	In	High	When this signal goes active, it starts the process of accessing the ICAP to obtain the device `IDCODE`.
icap_output[n:0]	In	N/A	This is the data returned from the ICAP: • Virtex-6 FPGA: n = 31 • Spartan-6 FPGA: n = 15
ready	Out	High	When active, this signal indicates that the `IDCODE` is available on idcode_result[31:0]. This signal remains active until cleared by pulsing reset.
idcode_result[31:0]	Out	N/A	This bus contains the device `IDCODE` value when ready is active.
input_to_icap[n:0]	Out	N/A	This bus transfers data to the ICAP: • Virtex-6 FPGA: n = 31 • Spartan-6 FPGA: n = 15
csb_to_icap	Out	Low	This is the ICAP enable signal.
rdwrb_to_icap	Out	N/A	This signal selects between reading and writing to the ICAP: • 1: ICAP in read mode • 0: ICAP in write mode

Upon receiving the start signal, the module interacts with the ICAP to obtain the device `IDCODE`. After the `IDCODE` is obtained, it is made available on the idcode_result bus. The module continues to output the `IDCODE` value until reset.

Design Overview

The reference design is implemented using SEM Controller v2.1 and the Xilinx® ISE® software v13.2. An implemented bitstream for either the ML605 board (`ml605.bit`) or SP605 board (`sp605.bit`) is in the implement directory.

Using the Reference Design

The reference design structure is as follows:

`XAPP517/<Virtex-6` or `Spartan-6>/<Verilog` or `VHDL>/`

- `implement`: Implementation scripts
- `reference_design_source`: Source code for reference design
- `sem_core`: SEM controller IP
- `coregen.cgp`: CORE Generator project file
- `chipscope_icon.xco`: ChipScope Pro tool integrated controller (ICON) core `.xco` file
- `chipscope_ila.xco`: ChipScope Pro tool integrated logic analyzer (ILA) core `.xco` file
- `chipscope_vio.xco`: ChipScope Pro tool virtual input/output (VIO) core `.xco` file
- `sem_core.xco`: SEM core `.xco` file

Any generated SEM controller IP configuration works with this design. The provided reference design has these settings:

- Component name: seu_ip
- Error injection: Enabled
- Error injection shim: ChipScope
- Error correction: Enabled
- Error correction method: Repair
- Error classification: Disable
- Controller clock frequency: 66 MHz (Virtex-6 device); 20 MHz (Spartan-6 device)

To test this design with other core settings, for example, using error correction or replacement, the user can generate a new core and then modify the `sem_example.{v|vhd}` file to include the `IDCODE` read submodule, ICAP multiplexing, and other associated logic. The provided `sem_example.{v|vhd}` file can be used as a guide.

UCF File Changes

The user constraint file (UCF) must be modified to add new ports and expand the HID shim area group due to the addition of the integrated logic analyzer (ILA) core. In the provided reference design UCF, the comments `##Reference Design Edits` show the modifications.

ML605 Board UCF Ports

```
NET "clk"                          LOC = "U23" ; ##USER_CLOCK SMA Pin X5.5
NET "status_initialization_led"      LOC = "AC22" ; ##GPIO_LED_0 J62 Pin 1
NET "status_observation_led"         LOC = "AC24" ; ##GPIO_LED_1 J62 Pin 2
NET "status_correction_led"          LOC = "AE22" ; ##GPIO_LED_2 J62 Pin 3
NET "status_classification_led"      LOC = "AE23" ; ##GPIO_LED_3 J62 Pin 4
NET "status_injection_led"           LOC = "AB23" ; ##GPIO_LED_4 J62 Pin 5
NET "status_idle_led"                LOC = "AG23" ; ##GPIO_LED_5 J62 Pin 6
NET "icap_sel_led"                   LOC = "AE24" ; ##GPIO_LED_6 J62 Pin 7
NET "status_heartbeat"               LOC = "AD24" ; ##GPIO_LED_7 J62 Pin 8
NET "status_uncorrectable_led"       LOC = "AD21" ; ##GPIO_LED_W
NET "monitor_tx"                     LOC = "J25"  ; ##USB_1_RX
NET "monitor_rx"                     LOC = "J24"  ; ##USB_1_TX
NET "pb_led_clear"                   LOC = "A18"  ; ##GPIO_SW_S SW6.2
```

```
NET "pb_read_idcode"                LOC = "G26"  ; ##GPIO_SW_C SW9.2
NET "pb_icap_sel"                   LOC = "A19"  ; ##GPIO_SW_N SW5.2
```

SP605 Board UCF Ports

```
NET "clk"                           LOC = "AB13";
NET "status_observation_led"        LOC = "D17" ; ##GPIO_LED_0 DS3
NET "status_idle_led"               LOC = "AB4" ; ##GPIO_LED_1 DS4
NET "icap_sel_led"                  LOC = "D21" ; ##GPIO_LED_2 DS5
NET "status_heartbeat"              LOC = "W15" ; ##GPIO_LED_3 DS6
NET "monitor_tx"                    LOC = "B21" ; ##USB_1_RX
NET "monitor_rx"                    LOC = "H17" ; ##USB_1_TX
NET "pb_led_clear"                  LOC = "F3"  ; ##GPIO_BUTTON_0 SW4.2
NET "pb_read_idcode"                LOC = "G6"  ; ##GPIO_BUTTON_1 SW7.2
NET "pb_icap_sel"                   LOC = "F5"  ; ##GPIO_BUTTON_2 SW5.2
```

ChipScope Pro Analyzer

The reference design uses ChipScope Pro analyzer software to exercise and analyze the results of the design. This is in addition to the board-level pushbuttons and LEDs that can also be used to exercise the design. When generating the SEM controller, the option to use the ChipScope Pro analyzer for injecting errors should be selected. This creates an example design file called `sem_hid.{v|vhd}` containing the basic ChipScope Pro analyzer instantiations for the ICON and VIO cores to inject errors and analyze the results. The reference design modifies this file to add an ILA core and new connections to the ILA and VIO cores. The ChipScope Pro analyzer files are already included in the reference design ChipScope Pro analyzer files directory. However, the files can be generated again if needed. *LogiCORE™ IP Soft Error Mitigation Controller v2.1 User Guide* [Ref 1] has information about generating the ChipScope Pro analyzer cores. However, due to modifications in the reference design, the instructions described in this section should be followed instead.

The ChipScope Pro analyzer ICON core must be generated for the target device using the default core name `chipscope_icon` with these parameters:

```
ENABLE_JTAG_BUFG = TRUE
NUMBER_CONTROL_PORTS = 2
USE_EXT_BSCAN = FALSE
USE_SOFTBSCAN= FALSE
USE_UNUSED_BSCAN = FALSE
```

The USER_SCAN_CHAIN can be set as desired. The VIO core must be generated for the target device using the default core name `chipscope_vio` with these parameters:

```
ENABLE_ASYNCHRONOUS_INPUT_PORT = FALSE
ENABLE_ASYNCHRONOUS_OUTPUT_PORT = FALSE
ENABLE_SYNCHRONOUS_INPUT_PORT = TRUE
ENABLE_SYNCHRONOUS_OUTPUT_PORT = TRUE
INVERT_CLOCK_INPUT = FALSE
SYNCHRONOUS_INPUT_PORT_WIDTH = 9
SYNCHRONOUS_OUTPUT_PORT_WIDTH = 38
```

The ILA core must be generated for the target device using the default core name `chipscope_ila` with these parameters:

```
COUNTER_WIDTH_1 through 16 = DISABLED
DATA_PORT_WIDTH = 0
DISABLE_SAVE_KEEP = FALSE
ENABLE_STORAGE_QUALIFICATION = TRUE
ENABLE_TRIGGER_OUTPUT_PORT = FALSE
EXAMPLE_DESIGN = FALSE
EXCLUDE_FROM_DATA_STORAGE_1 through 16 = FALSE
MATCH_TYPE_1 through 16 = BASIC_WITH_EDGES
MAX_SEQUENCE_LEVELS = 1
```

```
NUMBER_OF_TRIGGER_PORTS = 1
SAMPLE_DATA_DEPTH = 4096
SAMPLE_ON = RISING
TRIGGER_PORT_WIDTH_1 = 45
```

The simplest way to generate these cores is to open the provided CORE Generator™ tool project file (`coregen.cgp`) in the reference design, and regenerate the cores as needed using the GUI. For the implementation scripts of the provided design to work without modification, these files need to be placed in the ChipScope Pro tool files directory of the reference design. Alternatively, this can be done at the command line using the CORE Generator tool file and core `.xco` files as follows:

```
coregen -p coregen.cgp -b chipscope_icon.xco
coregen -p coregen.cgp -b chipscope_ila.xco
coregen -p coregen.cgp -b chipscope_vio.xco
```

The user can see the HID shim, `sem_hid.{v|vhd}`, to inspect the ChipScope Pro tool instantiation and interconnection of the ICON, VIO, and ILA cores.

Implementation Details

The reference design can be implemented using the `implement.bat` or `implement.sh` script provided in the `implement` directory of the reference design. A `results` directory is created with a bitstream titled `routed.bit`. The `routed.bit` file should be used to program the FPGA.

Using the Design on Xilinx Reference Boards

This section describes how to exercise the reference design on Xilinx reference boards. The ChipScope Pro analyzer is used to download the bitstream through the JTAG cable and verify the functionality of the design. Users should ensure that the ML605 or SP605 board is properly set up using the Getting Started Guide [Ref 2] [Ref 3] and User Guide [Ref 4] [Ref 5] for each board as a reference. The USB-to-UART bridge is used for the monitor interface connection. The Silicon Labs Virtual COM Port (VCP) driver must be installed to communicate with the SEM controller monitor interface via TeraTerm or HyperTerm. The drivers are available for free on the Silicon Labs driver download site (http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx). Refer to the Getting Started Guide and User Guide for each board for more information on installing the driver and using the USB-to-UART bridge.

Terminal Setup

The SEM controller monitor interface is used to control the state of the SEM controller and inject errors. This task is done using a communications terminal such as HyperTerm or TeraTerm. The terminal is set up as follows:

- Baud: 9600
- Settings: 8-N-1
- Flow control: None
- Terminal setup: VT100
- TX newline: CR
- RX newline: CR+LF
- Local echo: No

Demonstrating the Design

The steps and screen captures in this section guide the user through using the design on the ML605 board or SP605 board:

1. Power up the board, then open and set up the terminal.
2. Using the ChipScope Pro analyzer (or iMPACT software), download the bitstream. The expected terminal output looks like this:

```
V6_SEM_V2_1
SC 01
FS 02
ICAP OK
RDBK OK
INIT OK
SC 02
O>
```

The first line indicates the device and core version, so the line might vary depending on what is being implemented. For example, the Spartan-6 FPGA solution would read `S6_SEM_V2_1`.

If the ICAP is not controlled by the SEM controller at this point, the output does not get past the **ICAP** line, and **OK** is not present because the controller is trying to exchange data with the ICAP without success. The output then looks like this:

```
V6_SEM_V2_1
SC 01
FS 02
ICAP
```

Figure 2 shows the initial LED status indicating that the controller has moved through the initialization state and is in the OBSERVATION state. The heartbeat indicator should be blinking at this point.

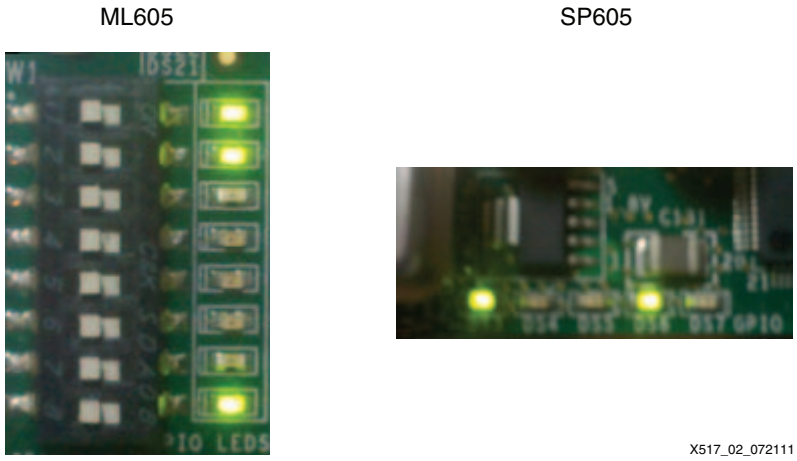
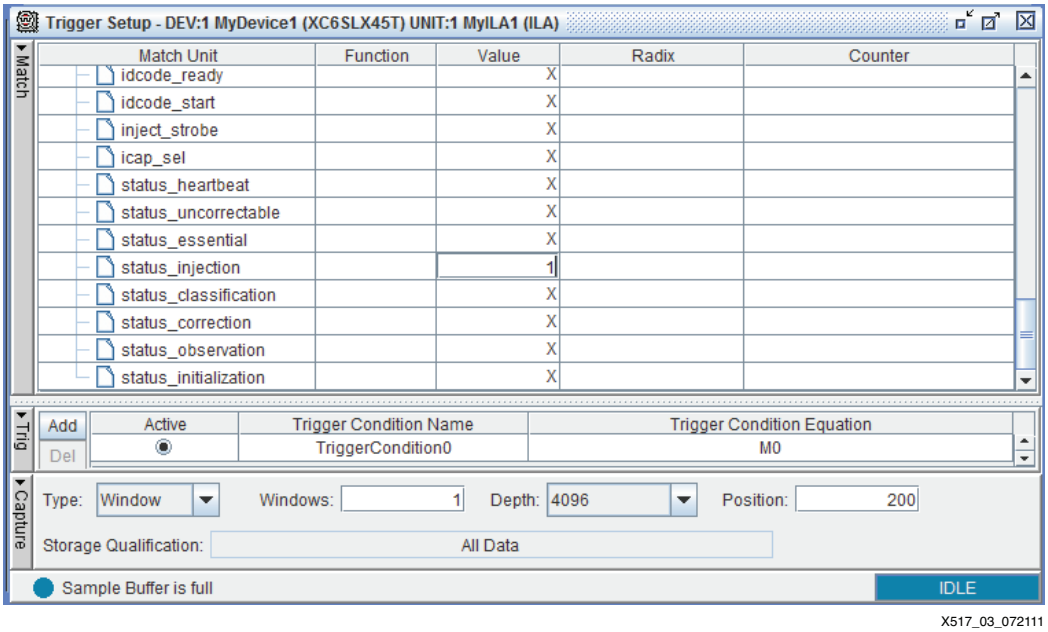


Figure 2: **Initial LED State for ML605 and SP605 Boards**

3. Set up the ChipScope Pro analyzer to trigger on any of the status signals to observe changes in the SEM controller state. Figure 3 shows triggering when the SEM controller indicates that an error is being injected.



X517_03_072111

Figure 3: **Setting Up Trigger to Capture Error Injection**

4. Use the terminal to inject an error using the monitor interface and observe the behavior:
   a. Idle the controller by typing **I** at the prompt. This results in the output `SC 00` and a new prompt indicating that the controller is in idle.
   b. Inject an error at linear frame address 7, word address 2, and bit address 1 on the ML605 board by typing **N C00007041**.
   c. Move the core back to OBSERVATION by typing **O** at the prompt. The resulting terminal information is shown here for the ML605 board:

```
O> I                    :Type I to idle controller
SC 00                   :State Change to IDLE
I> N C00007041          :Type to inject error
SC 10                   :State Change to INJECTION
SC 00                   :State Change to IDLE
I> O                    :Type O to move back to observation
SC 02                   :State Change to OBSERVATION
O>
SC 04                   :State Change to CORRECTION
SED OK                  :Single bit ECC error; SYNDROME valid
PA 000007               :Physical Address
LA 000007               :Linear Address
WD 02 BT 01             :Word and Bit location
COR                     :Start of Error Correction Report
WD 02 BT 01             :Word and Bit Location of correction
END                     :End of report
FC 00                   :Flag Change Correctable
SC 08                   :State Change CLASSIFICATION
FC 40                   :Flag Change Essential
```

```
SC 02                    :State Change OBSERVATION
O>
```

The output reports that the injected error was a 1-bit error, SYNDROME is valid, and the error was corrected. The log indicates that the bit is classified as essential, but this might not always be the case with other implementations. Different implementations can place and route the design differently causing this address to not always be essential. These logs are produced using the bitstreams provided with the reference design. *LogiCORE IP Soft Error Mitigation Controller v2.1 User Guide* [Ref 1] has more details on how to interpret the error detection and correction report output.

Figure 4 shows the LED status at this point. All of these status LEDs, except for the one indicating IDLE, are on at this point because the controller has moved through each of the states (INJECTION, CORRECTION, and CLASSIFICATION) during the error injection sequence.

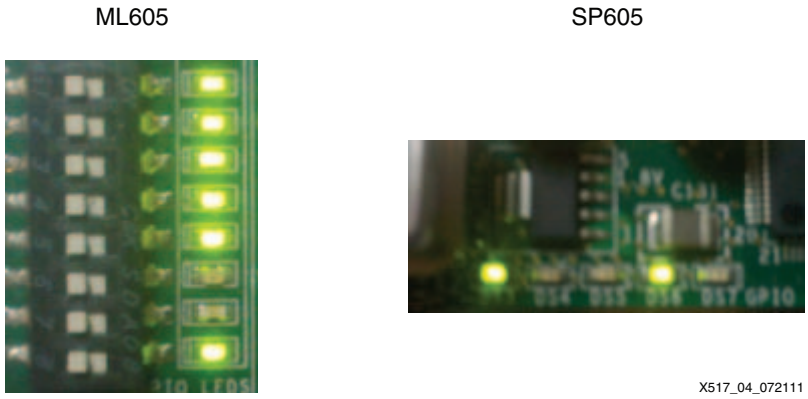


Figure 4: **LED Configuration after the Error for ML605 and SP605 Boards**

Due to the limited number of LEDs on the SP605 board, the LEDs representing injection, correction, and classification are not available as they are on the ML605 board. See Table 1 for more information. The SP605 board LED status only shows that the controller is in the OBSERVATION state and the heartbeat is toggling. However, the terminal information after injecting an error at linear frame address 7, word address 2, and bit address 1 (N C00007041) for the SP605 board is shown here:

```
O> I                     :Type I to idle controller
SC 00                    :State Change to IDLE
I> N C00007041           :Type to inject error
SC 10                    :State Change to INJECTION
SC 00                    :State Change to IDLE
I> O                     :Type O to move back to observation
SC 02                    :State Change to OBSERVATION
O>
SC 04                    :State Change to CORRECTION
SED OK                   :Single bit ECC error; SYNDROME valid
PA 000403                :Physical Address
LA 0007                  :Linear Address
WD 02 BT 01              :Word and Bit location
COR                      :Start of Error Correction Report
WD 02 BT 01              :Word and Bit Location of correction
END                      :End of report
FC 00                    :Flag Change Correctable
```

```
SC 08                    :State Change CLASSIFICATION
FC 40                    :Flag Change, Essential
SC 02                    :State Change OBSERVATION
O>
```

5. On the ML605 board, clear the LEDs using the pb_led_clear pushbutton. This step is not needed on the SP605 board. Due to the limited number of LEDs available, none of them retain their last state as they do on the ML605 board (see Table 1, page 2).
6. Now that a baseline of behavior has been established, idle the SEM controller by typing **I** on the terminal. Give control of the ICAP to the `IDCODE` read submodule, either by using pb_icap_sel or the VIO core. When targeting a Virtex-6 FPGA, the heartbeat still toggles in IDLE until the ICAP is accessed for other reasons. In the Spartan-6 FPGA, the heartbeat stops when the controller is placed into IDLE. See the Limitations and Considerations, page 13, for more information on the heartbeat signal behavior.
7. Set the ChipScope Pro tool trigger to detect the idcode_start signal and arm the trigger.
8. Push the pb_read_idcode pushbutton.
9. Figure 5 shows the results, including the `IDCODE` value (`0x24250093`) for the ML605 board. The expected `IDCODE` value using the SP605 board is `0x34028093`. These values might differ, depending on the board revision in use. For other valid device IDCODEs, refer to *Virtex-6 FPGA Configuration User Guide* [Ref 6] and *Spartan-6 FPGA Configuration User Guide* [Ref 7].

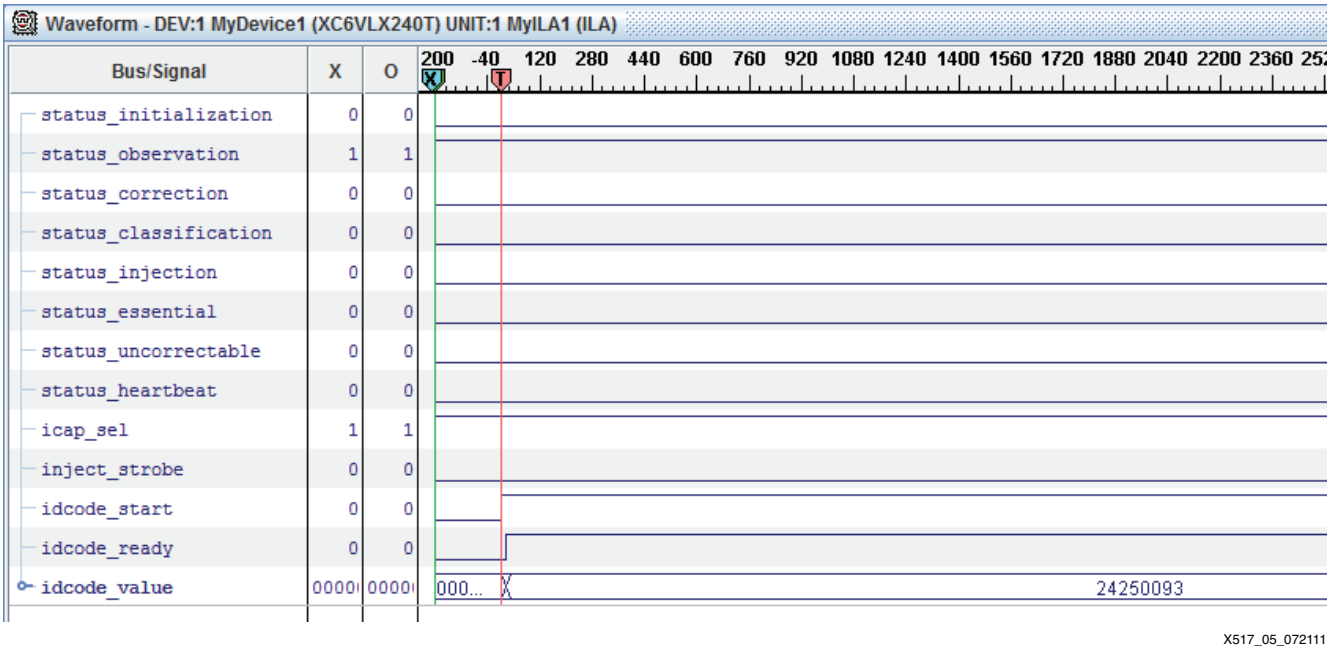


Figure 5: **ML605 Board IDCODE Result**

10. Return control of the ICAP back to the SEM controller by pushing pb_icap_sel or changing the icap_sel signal in the VIO core, depending on the method previously used.
11. Repeat the sub-steps in step 4, page 9 to use the terminal to inject an error using the monitor interface and observe the behavior. The results should again be the same, showing that a correctable error was injected and corrected by the SEM controller. This proves that the controller again has control of the ICAP, and is able to successfully detect and correct errors.

Further Experimentation

This section describes other ways to inject errors when using the SEM controller. To experiment further with error injection, these methods should be applied to the reference design included with this application note.

Error Injection Addresses

Errors can be injected into the configuration frames. The configuration frame size of the Virtex-6 FGPA is 2,592 bits wide, and the Spartan-6 FPGA configuration frame size is 1,040 bits wide. The frame size is always the same, but the number of frames depends on the device density. The SEM controller indicates the maximum linear frame address for a given device. This can be found using the status command at the terminal prompt. For example, on the ML605 board, the status command results in this output:

```
MF 0056F2 (Linear Count)
```

In this example, an error can be injected at any 17-bit long linear frame address from `0x00000` to `0x056F2`. However, not all errors injected in this range of addresses result in a corrected error by the SEM controller. The configuration memory is represented by different types of frames in the FPGA. During start-up, the SEM controller goes through a process of reading back the frames using the ICAP. The SEM controller then converts the physical address of the frame to the linear address and provides the upper boundary to the user through the status report. This value can be used to bound random address creation when performing extensive error injection testing. Depending on the device, certain frame types are discarded from the readback, so the linear address range only includes those frames that are scanned during readback. During error injection using linear frame addresses, the SEM controller converts the provided linear address to a physical address and injects the error at the appropriate place through the ICAP. The SEM controller also provides a way to inject errors using the physical frame address. However, this is more difficult and would mean that the injection logic would be customized to a particular device density. By using the linear count provided by the SEM controller, the logic can bound the upper address with the provided value and use the same method regardless of device density.

Errors can only be injected in type 0 frames and also when targeting the Virtex-6 FPGA, the center word of type 2 frames. Errors cannot be injected in other frame types. There are also whole frames that are unimplemented but are still addressable. Errors injected in these so-called “dummy” frames do not result in a correction by the SEM controller. This means that if random addresses are being used, it is likely that some injected errors are not corrected because they do not hit a valid address. There are many more type 0 frames than other types of frames where errors cannot be injected, and because of this, errors are corrected more often than not when performing error injection at random addresses.

The SEM controller checks that the provided address is within the valid address range. However, there are no negative side effects—other than no error is injected—if the address is above the maximum frame count provided by the controller. If the injection command is correctly formatted, the SEM IP enters the injection state. In the injection state, the address is checked against the upper limit previously stored by the controller. If the address is in bounds, the error is injected and the SEM IP returns to the IDLE state. If the address is out of bounds, the SEM IP returns to the IDLE state without doing anything.

Injecting an Error Using the Error Injection Interface

Along with the monitor interface, errors can also be injected using the error injection shim. Using the VIO core, the steps described in this section can be used to inject an error. Injecting an error on the error injection interface follows the same process as on the monitor Interface. The controller must be put into IDLE, the error is injected, and the controller is put back into OBSERVATION. The commands are presented by applying a 36-bit value on the inject_address bus and then asserting the inject_strobe signal. Apply these commands on the inject_address bus to change the controller state:

- Enter IDLE: `0xEXXXXXXXX`
- Enter OBSERVATION: `0xAXXXXXXXX`

Using the VIO core, follow these steps to inject an error:

1. Idle the controller by applying `0xE00000000` and then toggling the inject_strobe signal. Using the VIO console's pulse function, pulse the inject_strobe for the command to take effect. None of the status signals are asserted after toggling inject_strobe.
2. Apply the address to inject the error and toggle inject_strobe e.g., C_0000_7041. **C** is the command to inject the error, and the other bits indicate the linear frame address. Watch closely to observe a momentary change on status_injection.
3. Put the controller back in OBSERVATION. At this point, if the address targets a valid type 0 frame, changes occur on the status signals as the SEM controller moves through the states to correct the error.

For detailed information on using the error injection interface, see UG764, *LogiCORE IP Soft Error Mitigation Controller v2.1 User Guide.*

Limitations and Considerations

This section includes limitations and other information that users should be aware of when sharing control of the ICAP with the SEM controller. It is possible to corrupt the operation of the SEM controller IP when accessing the ICAP. For example, if the user overwrites configuration memory cells associated with the SEM controller via the ICAP, the overwrite could break the controller. Users must therefore exercise extreme caution when accessing the ICAP. Some other considerations are:

- Do not add any pipelining before or after the multiplexer selecting between the SEM controller and the user logic. Doing so prevents the SEM controller IP from being able to communicate with the ICAP.
- When in the IDLE state, the ability of the controller to detect, correct, and classify errors is suspended.
- When an error is injected, the INIT pin might go Low and a red light might be illuminated on the board. This indicates that the injected error has caused a cyclic redundancy check (CRC) failure as caught by the internal scanning logic. The injected error causes an error correction control (ECC) failure at the address, and after the scan finished, a CRC failure is found. After the SEM controller is moved back to the observation state, the red light goes off. This happens because, after the controller moves back to observation, it clears the CRC status output and the core also corrects the error. This means that when the next internal FPGA scan is finished, the ECC error no longer exists and no CRC failure is computed.
- Heartbeat:
  - In the Virtex-6 device, the heartbeat keeps operating even though the controller is placed in the IDLE state. This is because the heartbeat is an indication that the internal readback is functioning, which is done by the built-in silicon features on the Virtex-6 FPGA. The heartbeat does not stop until the ICAP is accessed for other reasons.
  - In the Spartan-6 device, the readback feature is performed through the ICAP interface instead of using the built-in silicon features. This means that when the ICAP is idled on the Spartan-6 device, the heartbeat stops.
- Issue a SYNC command to the ICAP when taking control over from the SEM core. This is shown in the `idcode_read.v[hd]` submodule available with the reference design. When giving control back to the SEM controller, do not issue a DESYNC command.

Command and Register Access Restrictions

The *Virtex-6 FPGA Configuration User Guide* [Ref 6] and *Spartan-6 FPGA Configuration User Guide* [Ref 7] list the various commands and register accesses available through the ICAP.

> ***Caution!*** Taking control of the ICAP as described in the application note has the potential to cause design failure. Any design that accesses the ICAP must be carefully tested to ensure it has no negative impact on the operation of the SEM controller and the rest of the design. Table 3 attempts to give some guidance on which commands or registers can be accessed and which should not.

In some cases, it would not be reasonable to use a given command anyway, and that is also pointed out as appropriate. Overall, the user should not do anything to write to any configuration frame memory location because doing so causes the CRC to fail on the next sweep made by the SEM controller, resulting in errors. It is generally permissible to read information from the device, but writes or changes of device content should be avoided as a general rule. Only access to the `IDCODE` register was tested as part of the reference design accompanying this application note. The rest of the conclusions in Table 3 through Table 8 are made based on information in the device configuration user guides and expectations of the SEM controller operation.

Table 3: **Packet Register Access Limitations**

Register	Device	Notes
CRC	Virtex-6 Spartan-6	Available to access. Users normally have no need to access this register because it is used for the configuration process.
FAR	Virtex-6	Available for access. This register should only be used for read operations.
FAR_MAJ	Spartan-6	Available for access. This register should only be used for read operations.
FAR_MIN	Spartan-6	Available for access. This register should only be used for read operations.
FDRI	Virtex-6 Spartan-6	Do not write to this register. Doing so introduces CRC failures that cannot be corrected by SEM controller.
FDRO	Virtex-6 Spartan-6	Do not access.
MASK	Virtex-6 Spartan-6	Available to access.
EYE_MASK	Spartan-6	Do not access.
LOUT	Virtex-6 Spartan-6	Available to access. However, it is unusual for a user to need to access this register.
MFWR	Virtex-6	Do not access this register. Doing so introduces CRC failures that cannot be corrected by the SEM controller.
CBC[_REG]	Virtex-6 Spartan-6	Available to access. However, it does not seem likely for the user to need to access this register.
IDCODE	Virtex-6 Spartan-6	Available to access.
CSBO	Virtex-6 Spartan-6	Available to access. However, it is unusual for a user to need to access this register.
AXSS	Virtex-6	Available to access.
DWC	Virtex-6	Available to access.

Table 4: **CMD Register Access Limitations**

Command	Device	Notes
Null	Virtex-6 Spartan-6	Available for use.
WCFG	Virtex-6 Spartan-6	Do not use.
MFW	Virtex-6 Spartan-6	Do not use.
LFRM	Virtex-6 Spartan-6	Available for use. However, it is unusual for a user to need to issue this command.
RCFG	Virtex-6 Spartan-6	Available for use.
START	Virtex-6 Spartan-6	Available for use. However, it is unusual for a user to need to issue this command.
RCRC	Virtex-6 Spartan-6	Do not use.
AGHIGH	Virtex-6 Spartan-6	Available for use. However, this stops the entire design, so operation of the SEM controller would be irrelevant after this command is issued. In most cases, the user would not want to do this.
SWITCH	Virtex-6	Available for use.
GRESTORE	Virtex-6 Spartan-6	Do not use.
SHUTDOWN	Virtex-6 Spartan-6	Available for use. However, this stops the entire design, so operation of the SEM controller would be irrelevant after this command is issued. In most cases, the user would not want to do this.
GCAPTURE	Virtex-6	Do not use.
DESYNC	Virtex-6 Spartan-6	Recommendation is not to issue DESYNC before handing control back to the SEM controller.
IPROG	Virtex-6 Spartan-6	Available for use. However, this command would initiate reconfiguration of the device. Thus, operation of the SEM controller would be irrelevant after this command is issued.
CRCC	Virtex-6	Do not use.
LTIMER	Virtex-6	Available for use.

Table 5: **Control Register 0 (CTL0)**

Register	Device	Notes
EFUSE_KEY USE_EFUSE_KEY	Virtex-6 Spartan-6	Available for access.
ICAP_SELECT	Virtex-6	Do not access.
OVERTEMPPOWERDOWN	Virtex-6	Available for access.
CONFIGFALLBACK	Virtex-6	Do not access.
GLUTMASK_B	Virtex-6	Available for access. However, it is important that if this register is changed, it MUST be changed back before returning the SEM controller to the OBSERVATION state.

Table 5: **Control Register 0 (CTL0) *(Cont'd)***

Register	Device	Notes
FARSRC	Virtex-6	Available for access. However, it is important that if this register is changed, it MUST be changed back before returning the SEM controller to the OBSERVATION state.
DEC	Virtex-6 Spartan-6	Available for access. However, because the SEM controller is not compatible with design security, the user should not be doing anything to enable design security anyway.
SBITS	Virtex-6 Spartan-6	Available for access. However, because the SEM controller is not compatible with design security, the user should not be doing anything to enable design security anyway.
PERSIST	Virtex-6 Spartan-6	Do not access.
GTS_USR_B	Virtex-6	Available for use. However, this stops the entire design, so operation of the SEM controller would be irrelevant after this command is issued. In most cases, the user would not want to do this.
CRC_EXTSTAT_DISABLE	Spartan-6	Do not access.

Table 6: **Other Registers**

Register	Device	Notes
FAR (Frame Address Register)	Virtex-6	Available for access.
STAT (Status Register)	Virtex-6 Spartan-6	Available for access.
Configuration Options Register 0 (COR0)	Virtex-6	Available for access. However, it would be unusual for the user to want to access this register through the ICAP.
Warm Boot Start Address (WBSTAR)	Virtex-6	Available for access.
Watchdog Timer (TIMER)	Virtex-6 Spartan-6	Available for access.
Boot History Status (BOOTSTS)	Virtex-6 Spartan-6	Available for access.

Table 7: **Virtex-6 FPGA Configuration Options Register 1**

Register	Notes
PERSIST_DEASSERT_AT_DESYNC	Do not access.
RBCRC_ACTION	Do not access.
RBCRC_NO_PIN	Do not access.
RBCRC_EN	Do not access.
BPI_1ST_READ_CYCLES	Available for access.
BPI_PAGE_SIZE	Available for access.

Table 8: **Other Spartan-6 FPGA Registers**

Register	Notes
COR1	Available for access. However, it would be unusual for the user to want to access this register through the ICAP.
COR2	Available for access. However, it would be unusual for the user to want to access this register through the ICAP.
PWRDN_REG	Do not access.
Frame Length	Do not access.
Multi-Frame Write	Do not access.
HC_OPT_REG	Do not access.
General Registers	Available for access.
MODE	Available for access.
CCLK_FREQ	Available for access, but it would be unusual to access this register through the ICAP.
PU_GWE	Do not access.
PU_GTS	Do not access.
SEU_OPT	Do not access.

Reference Design

The reference design can be downloaded from:

https://secure.xilinx.com/webreg/clickthrough.do?cid=177905.

The device utilization table is shown in Table 9.

Table 9: **Device Utilization Table**

Device	Device Utilization of ICAP Multiplexing Logic Only				Device Utilization of Entire Reference Design			
	LUTs	Flip-Flops	Block RAMs	BUFGs	LUTs	Flip-Flops	Block RAMs	BUFGs
Virtex-6 FPGA	34	0	0	0	951	1,118	9	2
Spartan-6 FPGA	18	0	0	0	1,286	1,174	22	2

The reference design checklist is shown in Table 10.

Table 10: **Reference Design Matrix**

Parameter	Description
General	
Developer Name	John Ayer
Target Devices (Stepping Level, ES, Production, Speed Grades)	Virtex-6 FPGA Spartan-6 FPGA
Source Code Provided?	Yes
Source Code Format	Verilog, VHDL
Design Uses Code or IP from Existing Reference Design, Application Note, 3rd party, or CORE Generator™ Software?	Yes, SEM IP

Table 10: **Reference Design Matrix *(Cont'd)***

Parameter	Description
Simulation	
Functional Simulation Performed?	N/A
Timing Simulation Performed?	N/A
Testbench Provided for Functional and Timing Simulations?	N/A
Testbench Format	N/A
Simulator Software and Version	N/A
SPICE/IBIS Simulations?	N/A
Implementation	
Synthesis Software Tools and Version	XST, ISE Design Suite 13.2
Implementation Software Tools and Version	ISE Design Suite 13.2
Static Timing Analysis Performed?	Yes
Hardware Verification	
Hardware Verified?	Yes
Hardware Platform Used for Verification	ML605 and SP605 boards

Conclusion

This design demonstrates that it is possible to share the ICAP used by the SEM controller with other designated user functions. This sharing is performed by implementing multiplexers on the ICAP control signals and switching control away from the SEM IP only when the controller is in the IDLE state. Using the Xilinx ML605 and SP605 reference boards and the ChipScope Pro analyzer, the SEM controller can relinquish the ICAP to allow the device `IDCODE` to be read back, take control again, and successfully correct injected errors. The user must be aware that by accessing the ICAP, it is possible to break the SEM controller's functionality, and that doing so is at the risk of the designer. Limitations and Considerations, page 13, lists some of the problems that might be caused when sharing the ICAP with other functionality.

References

This document uses the following references:

1. UG764, *LogiCORE IP Soft Error Mitigation Controller v2.1 User Guide*
2. UG533, *Getting Started with the Xilinx Virtex-6 FPGA ML605 Evaluation Kit*
3. UG525, *Getting Started with the Xilinx Spartan-6 FPGA SP605 Evaluation Kit*
4. UG534, *ML605 Hardware User Guide*
5. UG526, *SP605 Hardware User Guide*
6. UG360, *Virtex-6 FPGA Configuration User Guide*
7. UG380, *Spartan-6 FPGA Configuration User Guide*

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
12/02/11	1.0	Initial Xilinx release.

Notice of Disclaimer