Fault Injection Emulation Process on a FIR Filter

O. Ruano [1], F. García-Herrero [1], L.A. Aranda [1], A. Sánchez-Macián [1], L. Rodriguez [2] and J.A. Maestro [1], Senior Member, IEEE

[1] Universidad Antonio Nebrija; oruano@nebrija.es
[1] Universidad Antonio Nebrija; fgarciahe@nebrija.es
[1] Universidad Antonio Nebrija; laranda@nebrija.es
[1] Universidad Antonio Nebrija; asanche@nebrija.es
[2] Universidad Antonio Nebrija; lrodriguezs5@alumnos.nebrija.es
[1] Universidad Antonio Nebrija; jmaestro@nebrija.es

Step by step guide

A FIR filter is a common module which is often used in communication systems. It implements the following equation 1:

$$y[n] = \sum_{i=0}^{N-1} x[n-i] \cdot h[i] \tag{1}$$

where x[n] is the input signal, y[n] is the output, and h[i] are the filter coefficients [1] [2].

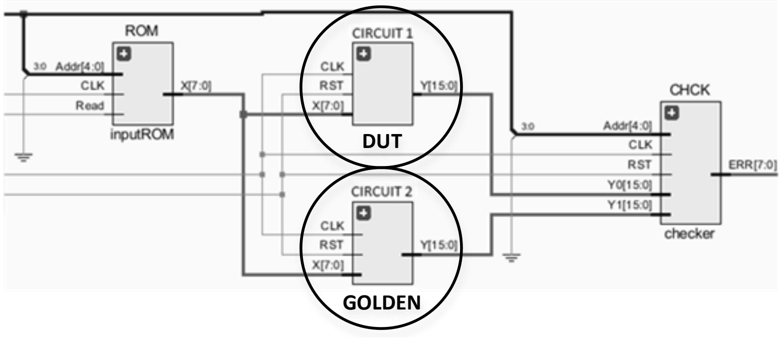


Figure 1 Design for the experimental setup.

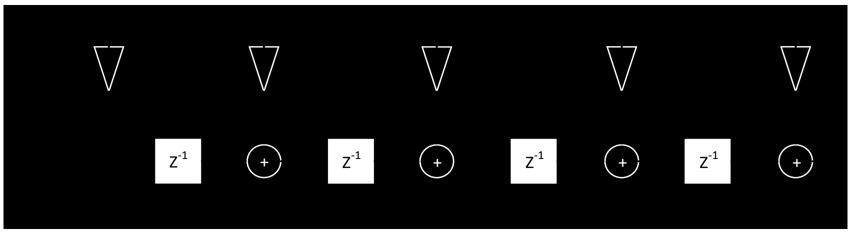
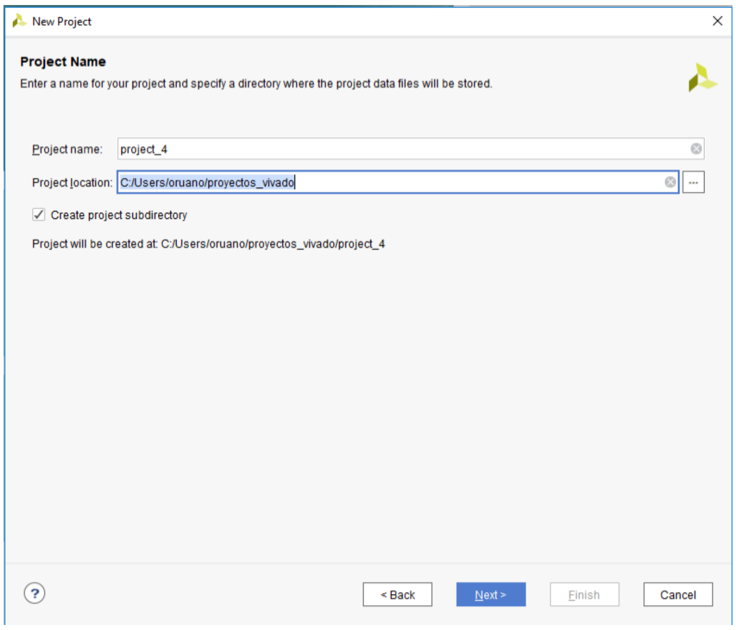


Figure 2 FIR filter implementation.

In the article "*Fault Injection Emulation for Systems in FPGAs: Tools, Techniques and Methodology*" has been explained that the set-up structure for the fault injection process consists of a ROM, two identical circuits (CIRCUIT 1 and CIRCUIT 2) and a checker module (CKCK). Figure 1 shows a diagram of this set-up with the different components to carry out the fault injection process on a FIR filter. The ROM module to server as the input signal x[n], the twin circuits 1 and 2 with theirs output signals y1[n] and y2[n] and the checker module to look for differences. From here on, the different stages are described step by step assuming that the circuits have been concreted in two FIR filter structures like the above in Figure 2.

1) *Create a new Vivado project:*



New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: project_4

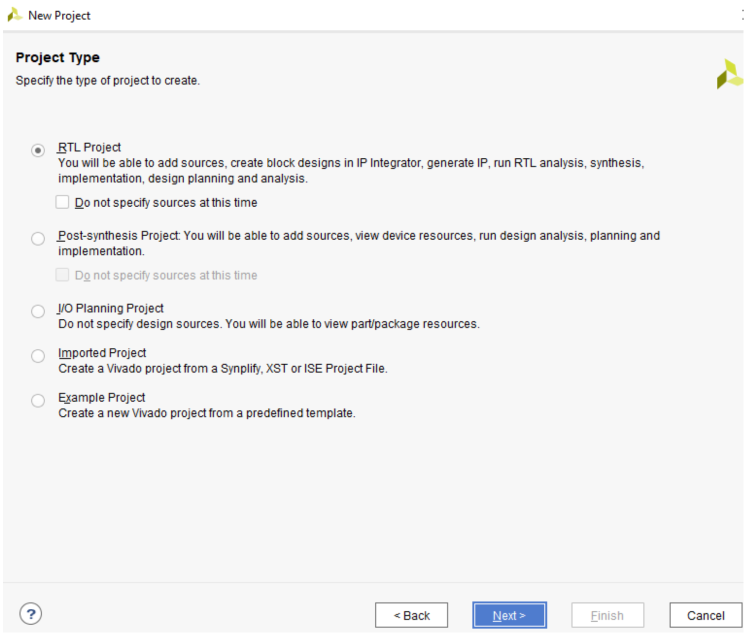
Project location: C:/Users/oruano/proyectos_vivado

☑ Create project subdirectory

Project will be created at: C:/Users/oruano/proyectos_vivado/project_4

Figure 3 Creating main project.

2) *Define an RTL project:*



New Project

Project Type

Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

☐ Do not specify sources at this time

○ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.

☐ Do not specify sources at this time

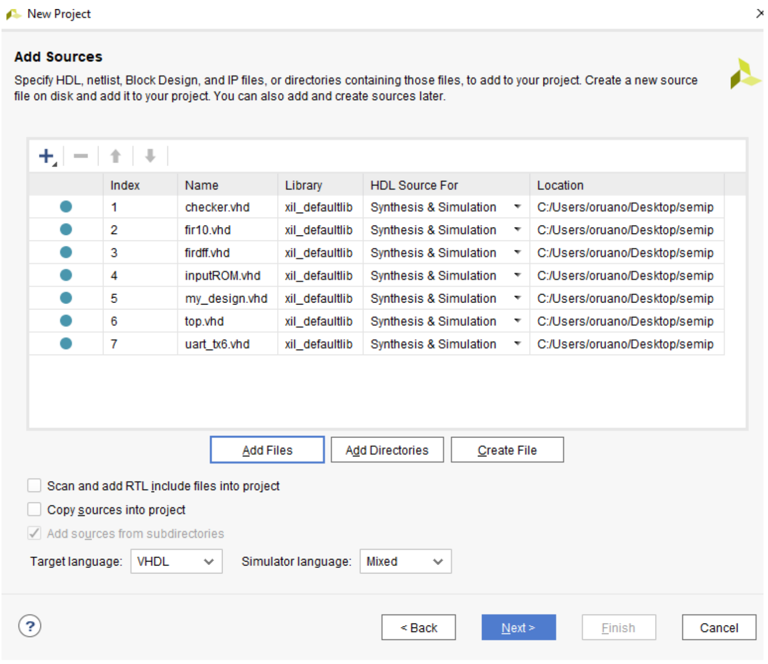
○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

○ Example Project
Create a new Vivado project from a predefined template.

Figure 4 RTL type project.

3) *Add the DUT HDL sources:*



New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

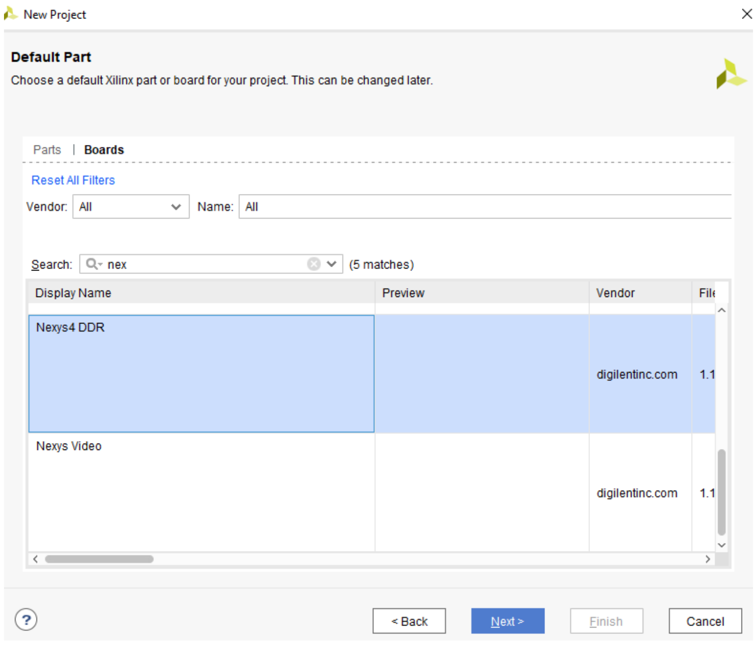
	Index	Name	Library	HDL Source For	Location
●	1	checker.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip
●	2	fir10.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip
●	3	firdff.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip
●	4	inputROM.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip
●	5	my_design.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip
●	6	top.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip
●	7	uart_tx6.vhd	xil_defaultlib	Synthesis & Simulation	C:/Users/oruano/Desktop/semip

☐ Scan and add RTL include files into project
☐ Copy sources into project
☑ Add sources from subdirectories

Target language: VHDL
Simulator language: Mixed

Figure 5 DUT sources files.

4) *Select a board model:*



New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Parts | **Boards**

Reset All Filters

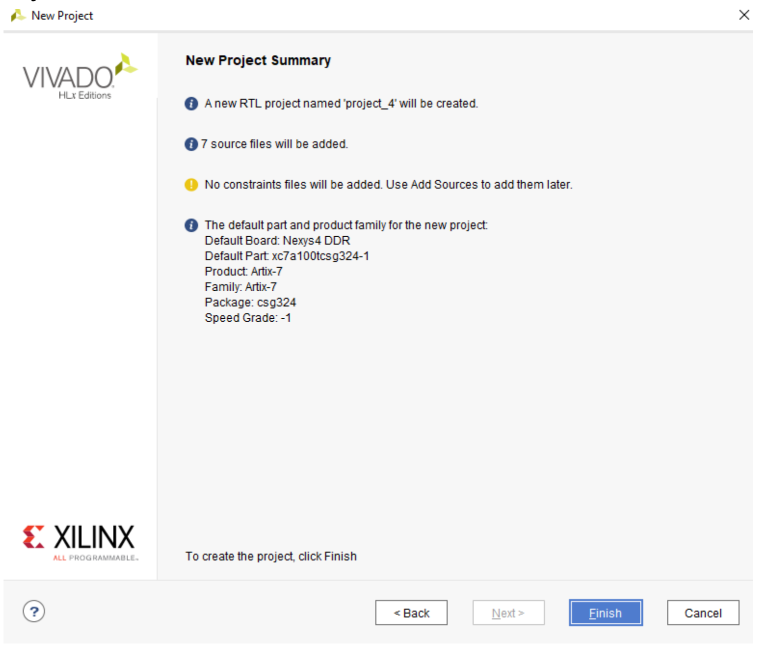
Vendor: All
Name: All

Search: nex (5 matches)

Display Name	Preview	Vendor	File
Nexys4 DDR		digilentinc.com	1.1
Nexys Video		digilentinc.com	1.1

Figure 6 Board selection.

5) *Project Summary:*


New Project

New Project Summary

A new RTL project named 'project_4' will be created.

7 source files will be added.

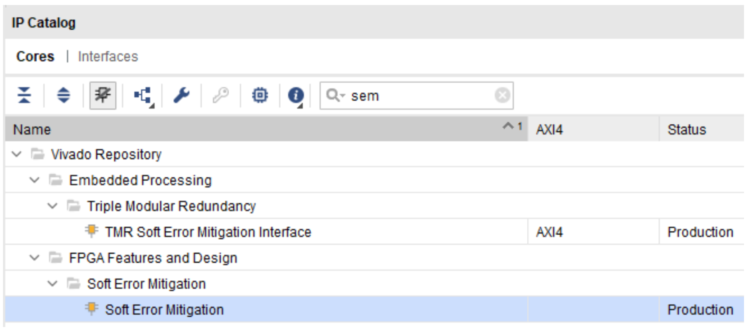
No constraints files will be added. Use Add Sources to add them later.

The default part and product family for the new project:
Default Board: Nexys4 DDR
Default Part: xc7a100tcsg324-1
Product: Artix-7
Family: Artix-7
Package: csg324
Speed Grade: -1

To create the project, click Finish

Figure 7 Project summary.

6) *Search from the IP Catalog the SEM IP core*


IP Catalog

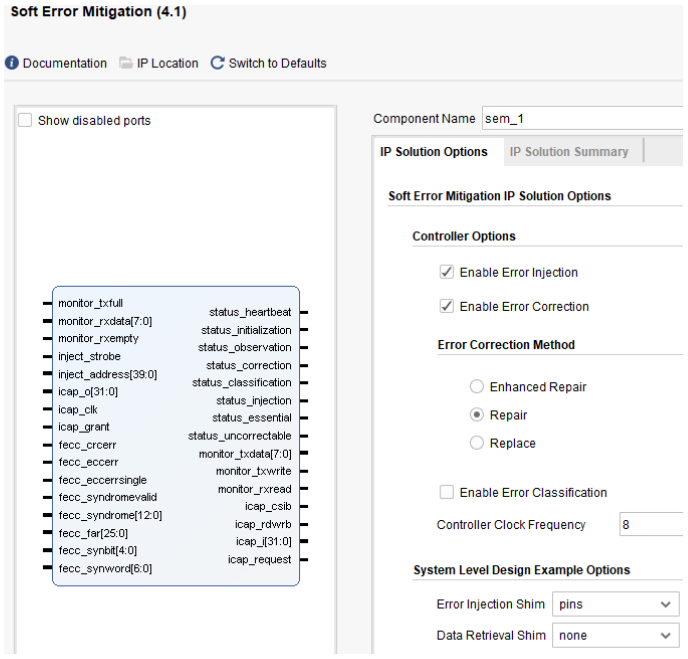
Cores | Interfaces

sem

Name	AXI4	Status
Vivado Repository		
Embedded Processing		
Triple Modular Redundancy		
TMR Soft Error Mitigation Interface	AXI4	Production
FPGA Features and Design		
Soft Error Mitigation		
Soft Error Mitigation		Production

Figure 8 SEM IP controller search.

7) *Configure the SEM IP*


Soft Error Mitigation (4.1)

Documentation   IP Location   Switch to Defaults

Show disabled ports

Component Name sem_1

IP Solution Options | IP Solution Summary

Soft Error Mitigation IP Solution Options

Controller Options

[✓] Enable Error Injection

[✓] Enable Error Correction

Error Correction Method

Enhanced Repair [✓ Repair] Replace

[ ] Enable Error Classification

Controller Clock Frequency 8

System Level Design Example Options

Error Injection Shim: pins

Data Retrieval Shim: none

Figure 9 SEM IP set-up.

8) *Open the SEM IP example to generate the source files. It creates a second project with the intention to generate the source files:*

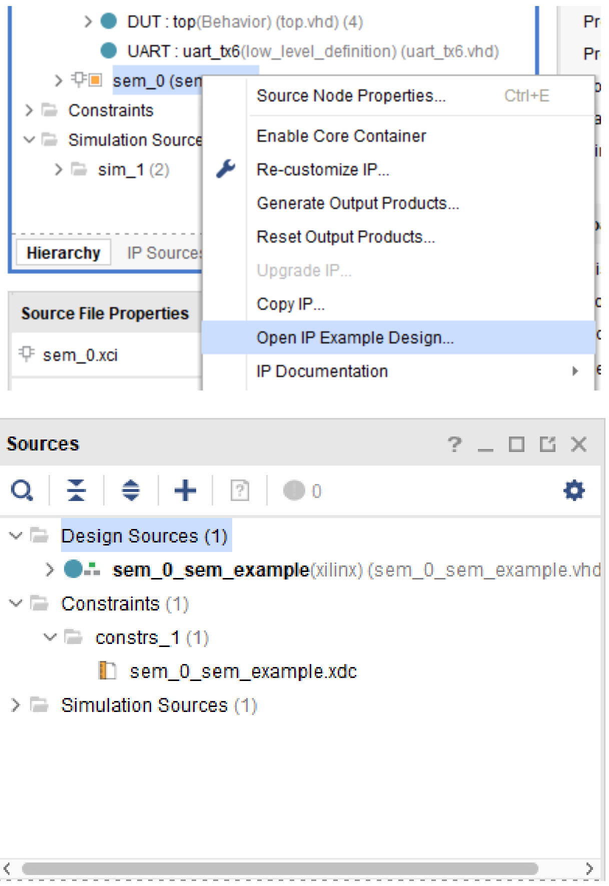
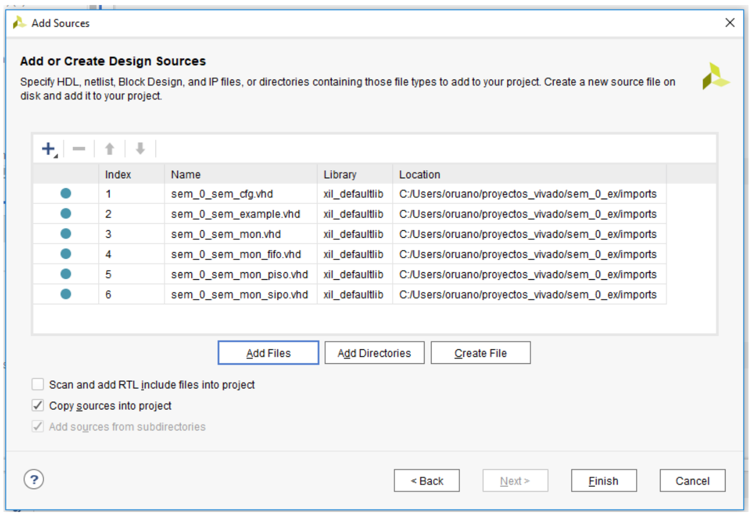


Figure 10 SEM IP controller example.

9) *Recycle SEM IP source files to be used in the main project:*


Add Sources

Add or Create Design Sources

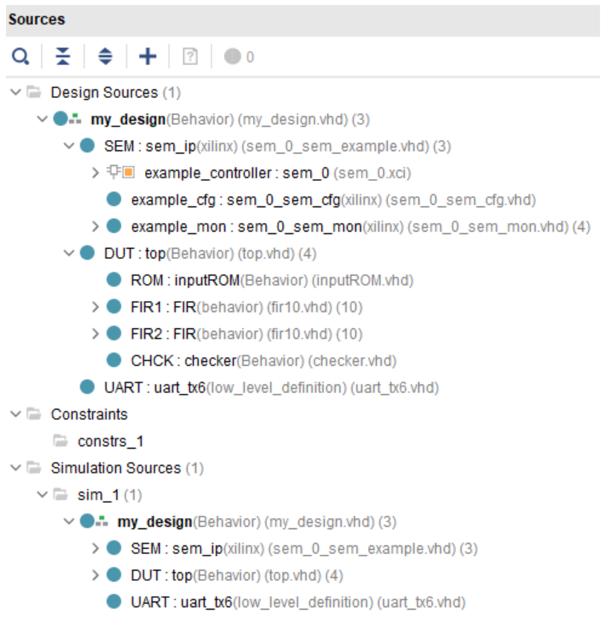
Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
●	1	sem_0_sem_cfg.vhd	xil_defaultlib	C:/Users/oruano/proyectos_vivado/sem_0_ex/imports
●	2	sem_0_sem_example.vhd	xil_defaultlib	C:/Users/oruano/proyectos_vivado/sem_0_ex/imports
●	3	sem_0_sem_mon.vhd	xil_defaultlib	C:/Users/oruano/proyectos_vivado/sem_0_ex/imports
●	4	sem_0_sem_mon_fifo.vhd	xil_defaultlib	C:/Users/oruano/proyectos_vivado/sem_0_ex/imports
●	5	sem_0_sem_mon_piso.vhd	xil_defaultlib	C:/Users/oruano/proyectos_vivado/sem_0_ex/imports
●	6	sem_0_sem_mon_sipo.vhd	xil_defaultlib	C:/Users/oruano/proyectos_vivado/sem_0_ex/imports

☐ Scan and add RTL include files into project
☑ Copy sources into project
☑ Add sources from subdirectories

Figure 11 *Adding SEM IP source files to the project.*

10) Add the SEM IP source files to complete the project final architecture:



```
Sources
Design Sources (1)
  my_design(Behavior) (my_design.vhd) (3)
    SEM : sem_ip(xilinx) (sem_0_sem_example.vhd) (3)
      example_controller : sem_0 (sem_0.xci)
      example_cfg : sem_0_sem_cfg(xilinx) (sem_0_sem_cfg.vhd)
      example_mon : sem_0_sem_mon(xilinx) (sem_0_sem_mon.vhd) (4)
    DUT : top(Behavior) (top.vhd) (4)
      ROM : inputROM(Behavior) (inputROM.vhd)
      FIR1 : FIR(behavior) (fir10.vhd) (10)
      FIR2 : FIR(behavior) (fir10.vhd) (10)
      CHCK : checker(Behavior) (checker.vhd)
    UART : uart_tx6(low_level_definition) (uart_tx6.vhd)
Constraints
  constrs_1
Simulation Sources (1)
  sim_1 (1)
    my_design(Behavior) (my_design.vhd) (3)
      SEM : sem_ip(xilinx) (sem_0_sem_example.vhd) (3)
      DUT : top(Behavior) (top.vhd) (4)
      UART : uart_tx6(low_level_definition) (uart_tx6.vhd)
```

Figure 12 Final architecture for the main project.

11) Adjust the SEM IP parameters with the design, in the original sem_0_sem_example file, such as:

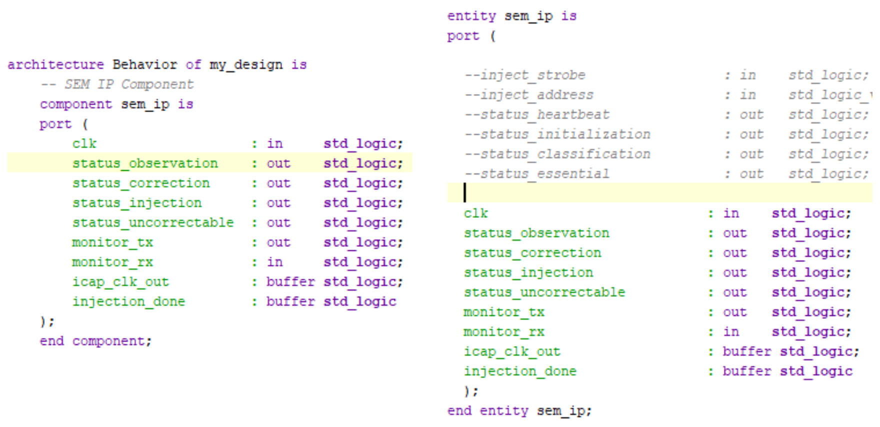
a. The SEM IP entity name (sem_ip).

b. Some entity ports are commented, because they remain unused:

inject_strobe : in std_logic;
inject_address : in std_logic_vector(39 downto 0);
status_heartbeat : out std_logic;
status_initialization : out std_logic;
status_classification : out std_logic;
status_essential : out std_logic;

c. Two buffer std_logic ports are added to distribute both clock and reset signals from the SEM IP controller to the rest of the design:

icap_clk_out : buffer std_logic;
injection_done : buffer std_logic



```
architecture Behavior of my_design is
    -- SEM IP Component
    component sem_ip is
    port (
        clk                   : in     std_logic;
        status_observation    : out    std_logic;
        status_correction     : out    std_logic;
        status_injection      : out    std_logic;
        status_uncorrectable  : out    std_logic;
        monitor_tx            : out    std_logic;
        monitor_rx            : in     std_logic;
        icap_clk_out          : buffer std_logic;
        injection_done        : buffer std_logic
    );
    end component;
```

```
entity sem_ip is
port (

  --inject_strobe                : in    std_logic;
  --inject_address               : in    std_logic_
  --status_heartbeat             : out   std_logic;
  --status_initialization        : out   std_logic;
  --status_classification        : out   std_logic;
  --status_essential             : out   std_logic;

  clk                          : in    std_logic;
  status_observation           : out   std_logic;
  status_correction            : out   std_logic;
  status_injection             : out   std_logic;
  status_uncorrectable         : out   std_logic;
  monitor_tx                   : out   std_logic;
  monitor_rx                   : in    std_logic;
  icap_clk_out                 : buffer std_logic;
  injection_done               : buffer std_logic
  );
end entity sem_ip;
```

Figure 13 SEM IP instanced component defined in my_design file – SEM IP entity from sem_0_sem_example file with the modifications.

These changes entail the following modifications for the example controller: sem_0 initialization, belonging to the sem_0_sem_example file (Table 1 in bold):

***sem_0_sem_example* file modified file**
example_controller : sem_0
port map (
status_heartbeat => open,
status_initialization => open,
status_observation => status_observation_internal,
status_correction => status_correction_internal,
status_classification => open,
status_injection => status_injection_internal,
status_essential => open,
status_uncorrectable=>status_uncorrectable_internal,
monitor_txdata => monitor_txdata,
monitor_txwrite => monitor_txwrite,
monitor_txfull => monitor_txfull,
monitor_rxdata => monitor_rxdata,
monitor_rxread => monitor_rxread,
monitor_rxempty => monitor_rxempty,
inject_strobe => '0',
inject_address => (others => '0'),
fecc_crcerr => fecc_crcerr,
fecc_eccerr => fecc_eccerr,
fecc_eccerrsingle => fecc_eccerrsingle,
fecc_syndromevalid => fecc_syndromevalid,
fecc_syndrome => fecc_syndrome,
fecc_far => fecc_far,
fecc_synbit => fecc_synbit,
fecc_synword => fecc_synword,
icap_o => icap_o,
icap_i => icap_i,
icap_csib => icap_csib,
icap_rdwrb => icap_rdwrb,
icap_clk => icap_clk,
icap_request => icap_unused,
icap_grant => icap_grant
);
icap_grant <= '1';
--status_heartbeat <= status_heartbeat_internal;
--status_initialization<=status_initialization_internal;
status_observation <= status_observation_internal;
status_correction <= status_correction_internal;
--status_classification<=status_classification_internal;
status_injection <= status_injection_internal;
--status_essential <= status_essential_internal;
status_uncorrectable<=not(status_uncorrectable_internal);
injection_done <= status_injection_internal;

Table 1 Modified sem_0_sem_example file.

Also, the status_uncorrectable signal, mapped on the reset button of the board has modified to work with negative logic:

status_uncorrectable <= not (status_uncorrectable_internal);

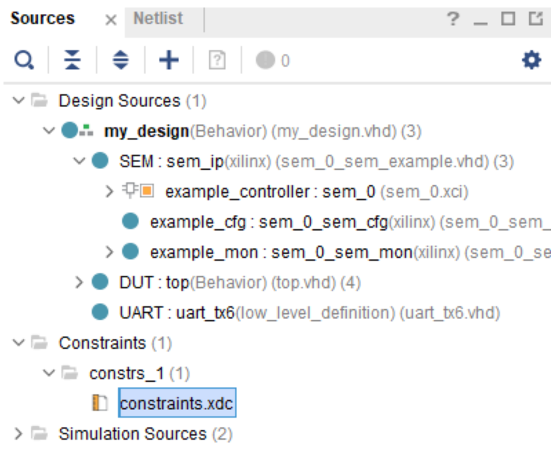
12) *Add the SEM IP constraint file to the project, from the sem_0_sem_example (Table 2 and Figure 14). As can be observed in Table 2, some add-on features related to the DUT and UART placement are*

added. Also, three additional commands are included regarding not to use DPS blocks, generating essential bit file (EBD), and enabling flash memory programming:

```
########################################
## Controller: Internal Timing constraints
########################################
set_max_delay 9.0 -from [get_pins
SEM/example_cfg/example_frame_ecc/*] -quiet -datapath_only
set_max_delay 20.0 -from [get_pins
SEM/example_cfg/example_frame_ecc/*] -to [all_outputs] -quiet -
datapath_only

########################################
## Master Clock
########################################
create_clock -name clk -period 10.0 [get_ports clk]
set_property IOSTANDARD LVCMOS25 [get_ports clk]

########################################
## Status Pins
########################################
set_property DRIVE 8            [get_ports status_observation]
set_property SLEW FAST            [get_ports status_observation]
set_property IOSTANDARD LVCMOS25    [get_ports
status_observation]

set_property DRIVE 8            [get_ports status_correction]
set_property SLEW FAST            [get_ports status_correction]
set_property IOSTANDARD LVCMOS25    [get_ports
status_correction]

set_property DRIVE 8            [get_ports status_injection]
set_property SLEW FAST            [get_ports status_injection]
set_property IOSTANDARD LVCMOS25    [get_ports status_injection]

set_property DRIVE 8            [get_ports status_uncorrectable]
set_property SLEW FAST            [get_ports status_uncorrectable]
set_property IOSTANDARD LVCMOS25    [get_ports
status_uncorrectable]

set_output_delay -clock clk -10.0 [get_ports [list status_observation
status_correction status_injection status_uncorrectable]] -max
set_output_delay -clock clk 0 [get_ports [list status_observation
status_correction status_injection status_uncorrectable]] -min

########################################
## MON Shim and Pins
########################################
set_property DRIVE 8               [get_ports monitor_tx]
set_property SLEW FAST               [get_ports monitor_tx]
set_property IOSTANDARD LVCMOS25       [get_ports monitor_tx]
set_property IOSTANDARD LVCMOS25       [get_ports monitor_rx]

set_input_delay -clock clk -max -10.0   [get_ports monitor_rx]
set_input_delay -clock clk -min 20.0    [get_ports monitor_rx]
set_output_delay -clock clk -10.0       [get_ports monitor_tx] -max
set_output_delay -clock clk 0           [get_ports monitor_tx] -min
```

```
########################################
## Logic Placement
########################################
create_pblock SEM_CONTROLLER
resize_pblock -pblock SEM_CONTROLLER -add
RAMB18_X1Y30:RAMB18_X1Y39
resize_pblock -pblock SEM_CONTROLLER -add
RAMB36_X1Y15:RAMB36_X1Y19
resize_pblock -pblock SEM_CONTROLLER -add
SLICE_X28Y75:SLICE_X35Y99
add_cells_to_pblock -pblock SEM_CONTROLLER -cells [get_cells
SEM/example_controller]
add_cells_to_pblock -pblock SEM_CONTROLLER -cells [get_cells
SEM/example_mon/*]

## Force ICAP to the required (top) site in the device.
## Force FRAME_ECC to the required (only) site in the device.
set_property LOC FRAME_ECC_X0Y0 [get_cells
SEM/example_cfg/example_frame_ecc]
set_property LOC ICAP_X0Y1      [get_cells
SEM/example_cfg/example_icap]

########################################
## I/O Placement
########################################
# SEM IP Ports
set_property LOC E3     [get_ports clk]
set_property LOC H17    [get_ports status_observation]
set_property LOC K15    [get_ports status_correction]
set_property LOC J13    [get_ports status_injection]
set_property LOC C17    [get_ports status_uncorrectable]
set_property LOC F16    [get_ports monitor_tx]
set_property LOC G16    [get_ports monitor_rx]

# Place My DUT
create_pblock MY_DUT
resize_pblock -pblock MY_DUT -add
SLICE_X0Y100:SLICE_X5Y149
add_cells_to_pblock -pblock MY_DUT -cells [get_cells DUT/FIR1/*]
set_property EXCLUDE_PLACEMENT true       [get_pblocks
MY_DUT]
set_property DONT_TOUCH true [get_cells DUT/FIR1/*]
set_property DONT_TOUCH true [get_cells DUT/ROM/*]
set_property DONT_TOUCH true [get_cells DUT/FIR2/*]
set_property DONT_TOUCH true [get_cells DUT/CHCK/*]

# Place My UART
create_pblock MY_UART
resize_pblock -pblock MY_UART -add SLICE_X0Y96:SLICE_X7Y99
add_cells_to_pblock -pblock MY_UART -cells [get_cells UART]
add_cells_to_pblock -pblock MY_UART -cells [get_cells DUT/CHCK]
# DUT Serial Output
set_property DRIVE 8            [get_ports S_OUT]
set_property SLEW FAST          [get_ports S_OUT]
set_property IOSTANDARD LVCMOS25 [get_ports S_OUT]
set_property LOC F6             [get_ports S_OUT]

##########################################################
###################
## Additional XDC Commands
##########################################################
###################
# Do not use DSP Blocks for the DCT
set_property USE_DSP48 no [get_cells DUT/FIR1/*]

# Generate EBD File
set_property BITSTREAM.SEU.ESSENTIALBITS yes
[current_design]

# For Flash Memory Programming
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4
[current_design]
set_property CONFIG_MODE SPIx4              [current_design]
```

Table 2 Constraint file example.



Sources × Netlist

- Design Sources (1)
  - my_design(Behavior) (my_design.vhd) (3)
    - SEM : sem_ip(xilinx) (sem_0_sem_example.vhd) (3)
      - example_controller : sem_0 (sem_0.xci)
      - example_cfg : sem_0_sem_cfg(xilinx) (sem_0_sem_
      - example_mon : sem_0_sem_mon(xilinx) (sem_0_se
    - DUT : top(Behavior) (top.vhd) (4)
    - UART : uart_tx6(low_level_definition) (uart_tx6.vhd)
- Constraints (1)
  - constrs_1 (1)
    - constraints.xdc
- Simulation Sources (2)

Figure 14 Constraint file added to the project.

13) *Run synthesis and implementation with Vivado. The implemented design is shown in Figure 15. If everything is correct, the bitstream can be generated via Generate Bitstream command (my_design.bit).*

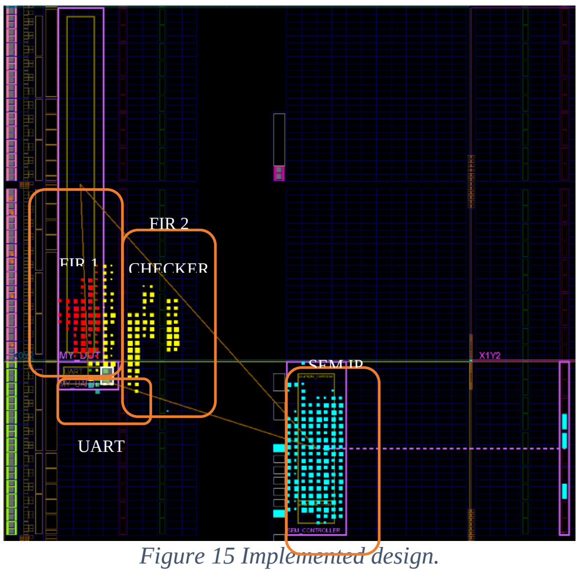
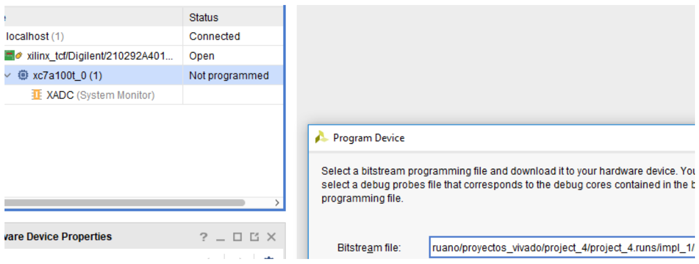


Figure 15 Implemented design.

14) *Hardware Manager can be opened in order to connect and program the board as it is exposed in the next two figures. It can be noticed how, in Figure 17, a bridge to send a reset signal is matched to automate the fault campaign.*



	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210292A401...	Open
xc7a100t_0 (1)	Not programmed
XADC (System Monitor)	

ware Device Properties

Program Device

Select a bitstream programming file and download it to your hardware device. You select a debug probes file that corresponds to the debug cores contained in the b programming file.

Bitstream file: ruano/proyectos_vivado/project_4/project_4.runs/impl_1/

Figure 16 Programming the board.

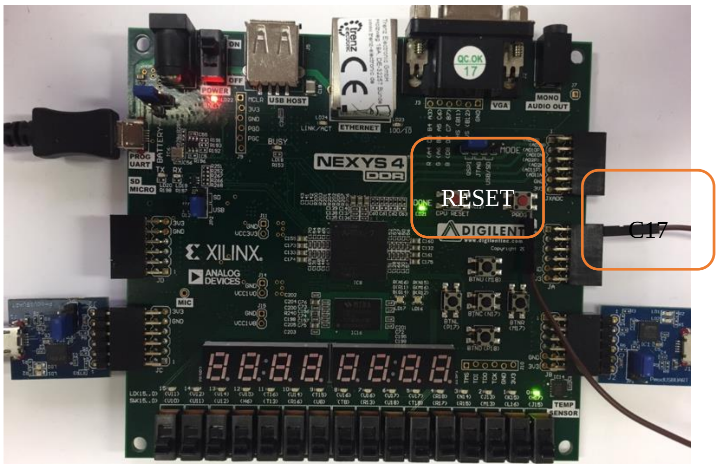
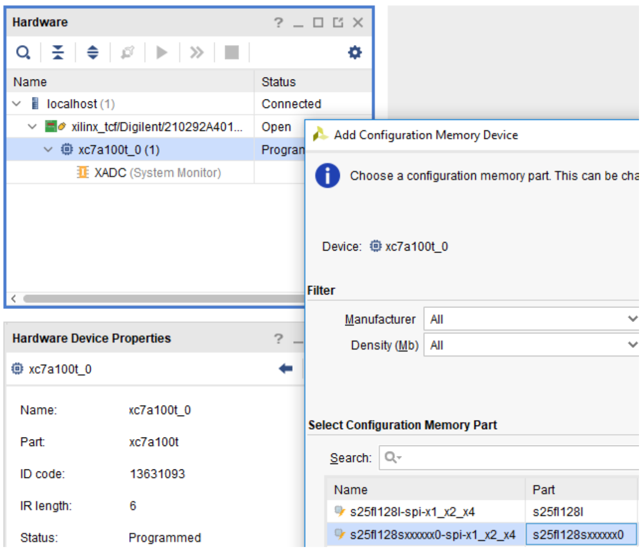


Figure 17 Board ready to work. Bridge for the reset after each fault injection.

15) *A flash memory is added to automate the reconfiguration process after each injected SEU.*



Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210292A401...	Open
xc7a100t_0 (1)	Program
XADC (System Monitor)	

Hardware Device Properties

xc7a100t_0

Name:	xc7a100t_0
Part:	xc7a100t
ID code:	13631093
IR length:	6
Status:	Programmed

Add Configuration Memory Device

Choose a configuration memory part. This can be cha

Device: xc7a100t_0

Filter

Manufacturer: All

Density (Mb): All

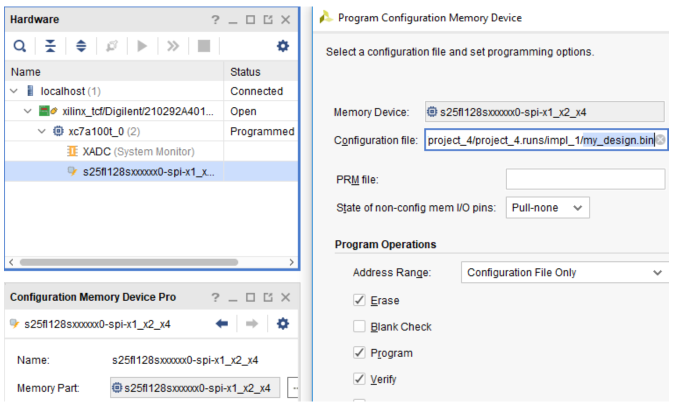
Select Configuration Memory Part

Search:

Name	Part
s25fl128l-spi-x1_x2_x4	s25fl128l
s25fl128sxxxxxx0-spi-x1_x2_x4	s25fl128sxxxxxx0

Figure 18 Flash memory selection.

16) *Program the flash memory with the .bin file stored in the impl_1 project's folder.*



Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210292A401...	Open
xc7a100t_0 (2)	Programmed
XADC (System Monitor)	
s25fl128sxxxxxx0-spi-x1_x...	

Configuration Memory Device Pro

s25fl128sxxxxxx0-spi-x1_x2_x4

Name:	s25fl128sxxxxxx0-spi-x1_x2_x4
Memory Part:	s25fl128sxxxxxx0-spi-x1_x2_x4

Program Configuration Memory Device

Select a configuration file and set programming options.

Memory Device: s25fl128sxxxxxx0-spi-x1_x2_x4

Configuration file: project_4/project_4.runs/impl_1/my_design.bin

PRM file:

State of non-config mem I/O pins: Pull-none

Program Operations

Address Range: Configuration File Only

- [x] Erase
- [ ] Blank Check
- [x] Program
- [x] Verify

Figure 19 Flash memory programmed (my_design.bin).

17) *Run the ACME tool in order to improve the fault injection in the configuration memory of the FPGA. It requires the EBD file, that is anticipated in the constraint file through,*

# Generate EBD File

set_property BITSTREAM.SEU.ESSENTIALBITS yes [current_design]

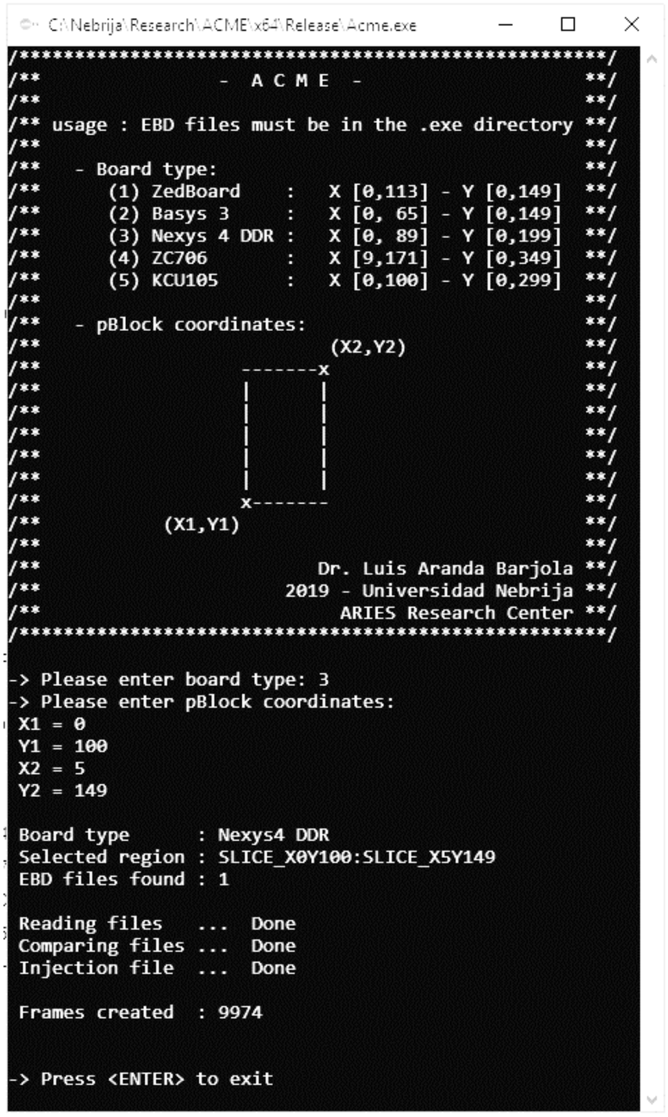
This file ought to be copied in the .exe directory: /ACME/x64/Release.
Based on both EBD file and the pBlocks coordinates where FIR1 has been placed,

create_pblock MY_DUT

*resize_pblock -pblock MY_DUT -add SLICE_**X0Y100**:SLICE_**X5Y149***

add_cells_to_pblock -pblock MY_DUT -cells [get_cells DUT/FIR1/]*

ACME returns a file named FrameRange.txt, including a subset of 9.974 essential bits which have been translated to injection addresses for the SEM IP controller (Figure 20). These bits belong to the FIR1 filter (FIR1 in the Figure 15).



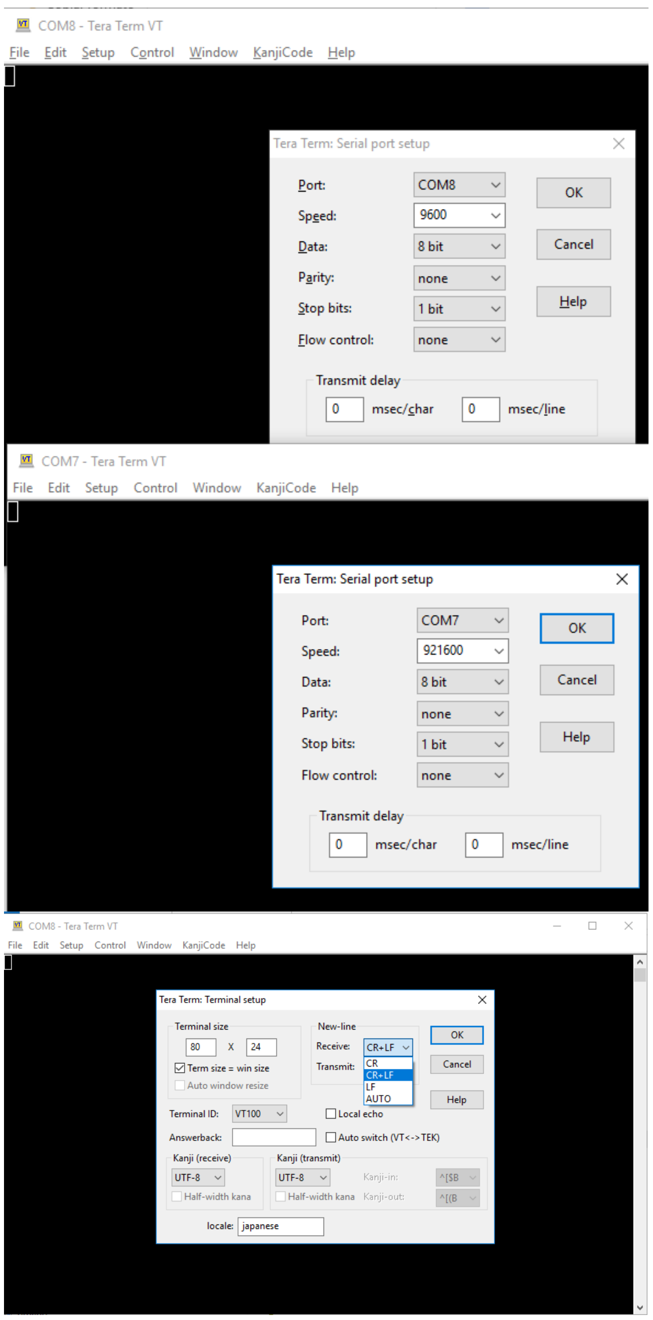
C:\Nebrija\Research\ACME\x64\Release\Acme.exe

```
/*****************************************************/
/**                   - A C M E -                   **/
/**                                                 **/
/** usage : EBD files must be in the .exe directory **/
/**                                                 **/
/**   - Board type:                                 **/
/**      (1) ZedBoard    :   X [0,113] - Y [0,149]  **/
/**      (2) Basys 3     :   X [0, 65] - Y [0,149]  **/
/**      (3) Nexys 4 DDR :   X [0, 89] - Y [0,199]  **/
/**      (4) ZC706       :   X [9,171] - Y [0,349]  **/
/**      (5) KCU105      :   X [0,100] - Y [0,299]  **/
/**                                                 **/
/**   - pBlock coordinates:                         **/
/**                             (X2,Y2)             **/
/**                     -------x                    **/
/**                     |      |                    **/
/**                     |      |                    **/
/**                     |      |                    **/
/**                     |      |                    **/
/**                     |      |                    **/
/**                     x-------                    **/
/**           (X1,Y1)                               **/
/**                                                 **/
/**                          Dr. Luis Aranda Barjola **/
/**                       2019 - Universidad Nebrija **/
/**                             ARIES Research Center **/
/*****************************************************/

-> Please enter board type: 3
-> Please enter pBlock coordinates:
 X1 = 0
 Y1 = 100
 X2 = 5
 Y2 = 149

 Board type      : Nexys4 DDR
 Selected region : SLICE_X0Y100:SLICE_X5Y149
 EBD files found : 1

 Reading files   ...  Done
 Comparing files ...  Done
 Injection file  ...  Done

 Frames created  : 9974


-> Press <ENTER> to exit
```

Figure 20 ACME tool chosen the number of injection.

18) *Connect the physical Pmods accordingly to the logical mapping interfaces defined in the constraint file (SEM IP in JB and UART in JC).*

19) *Test the connection between the host computer and the device (board), through the Tera Term serial port connection.*


COM8 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
OK
Cancel
Help
COM7 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
OK
Cancel
Help
COM8 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
OK
Cancel
Help

Tera Term: Serial port setup

Port:	COM8
Speed:	9600
Data:	8 bit
Parity:	none
Stop bits:	1 bit
Flow control:	none

Transmit delay: 0 msec/char 0 msec/line

Tera Term: Serial port setup

Port:	COM7
Speed:	921600
Data:	8 bit
Parity:	none
Stop bits:	1 bit
Flow control:	none

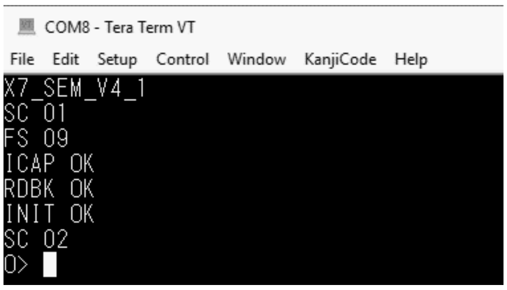
Transmit delay: 0 msec/char 0 msec/line

Tera Term: Terminal setup

Terminal size: 80 X 24
☑ Term size = win size
☐ Auto window resize
New-line: Receive: CR+LF (CR, CR+LF, LF, AUTO) Transmit:
Terminal ID: VT100
☐ Local echo
Answerback:
☐ Auto switch (VT<->TEK)
Kanji (receive): UTF-8 ☐ Half-width kana
Kanji (transmit): UTF-8 ☐ Half-width kana Kanji-in: ^[$B Kanji-out: ^[(B
locale: japanese

Figure 21 (Top) SEM IP serial connection (COM8, 9600 bauds for a V_ENABLETIME equal to 650). (Middle) DUT serial connection (COM7, 921600 bauds for UART Baud_Count equal to 6). (Bottom) Both Terminals: Receive new-line: CR-LF (carriadge return-line feed).

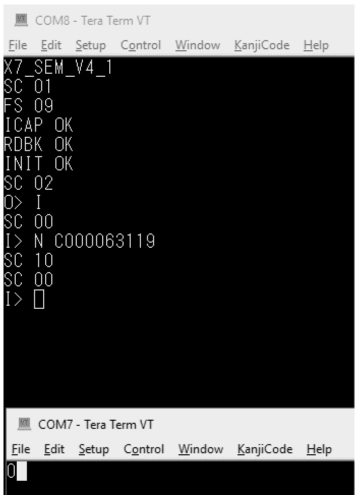
20) *Press the button reset on the board to initialize the emulation environment.*



```
X7_SEM_V4_1
SC 01
FS 09
ICAP OK
RDBK OK
INIT OK
SC 02
O>
```

Figure 22 SEM IP controller ready for the injection.

21) *Injecting a SEU in an arbitrary essential bit address like C000063119, and receiving the result through the COM7 (DUT's UART):*


COM8 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
COM7 - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
X7_SEM_V4_1
SC 01
FS 09
ICAP OK
RDBK OK
INIT OK
SC 02
O> I
SC 00
I> N C000063119
SC 10
SC 00
I>
```

```
0
```

Figure 23 A performance for a single fault injection.

22) *Finally, in order to carry out an exhaustive fault campaign based on the 9,974 essential bits identified by the ACME tool, it is advisable to automatize the process through a Matlab script (Figure 24) given as a result a report similar to the one in Table 3.*

```
%% Injection Loop
for i = 1 to FrameRange % Number of iterations equal to number of
SEUs

   % IDLE
  fprintf(sp,'I'); % SP full duplex serial communication for the SEM IP
  fscanf(sp);

  % INJECTION
  fprintf(sp, ['N ' FrameRange (i,:)]);
  fscanf(sp);

  % OBSERVATION
  fprintf(sp,'O');
```

Figure 24 Example of fault injection loop in Matlab.

```
-            ERROR REPORT            -
· Total Number of Injections: 9974
…
C000048155 ->  48
C000048156 ->  49
C000048157 ->  48
C000048159 ->  48
C00004815A ->  49
```

```
C00004815B ->  48
C00004815D ->  48
C00004815E ->  49
C000048160 ->  48
C000048161 ->  48
```

Table 3 Error report: ASCII code 48 (0) no error; ASCII code 49 (1) error.

References

1. P. Reviriego, C. Bleakley, J.A. Maestro, "Structural DMR: a Technique for Implementation of Soft Error Tolerant FIR Filters", IEEE Transactions on Circuits and Systems II (ISSN: 1549-7747), Vol. 58, No 8, August 2011, pp. 512-516.
2. A. V. Oppenheim and R. W. Schafer, “Discrete Time Signal Processing”, Prentice Hall, 1999.