A NOVEL HANDS-ON APPROACH TO LEARNING FPGA DESIGN

Tuesday, December 6th, 2016

1:00 PM PST

UBM

ESC EMBEDDED SYSTEMS CONFERENCE '16

#ESCsv

Pre-Requisites

You Will Need:

- **Laptop with Windows or Linux (***Tools not supported on Mac***)**

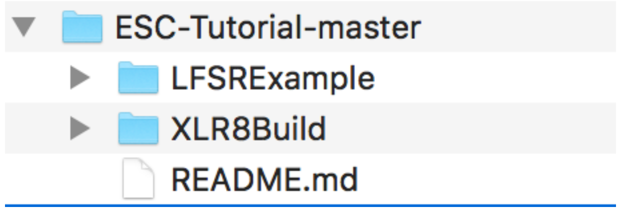
- **Installed Tools:**
  - Arduino IDE
  - Intel Quartus Prime Lite Edition
    - *Includes Modelsim-Intel FPGA Edition and Max 10 FPGA support*

- **A USB Mini cable for connecting XLR8 board to laptop**

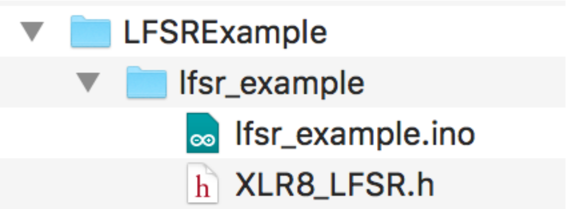
Follow the instructions here:
http://www.aloriumtech.com/openxlr8/

Tutorial Download

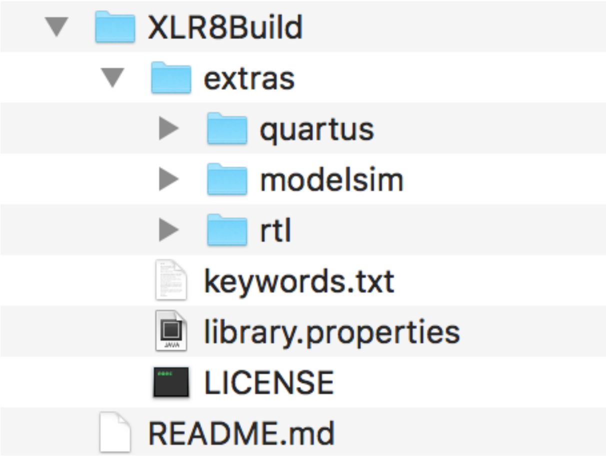
- **LFSR Code Package:** *https://github.com/AloriumTechnology/ESC-Tutorial*



```
▼ ESC-Tutorial-master
    ▶ LFSRExample
    ▶ XLR8Build
      README.md
```



```
▼ LFSRExample
    ▼ lfsr_example
          lfsr_example.ino
          XLR8_LFSR.h
```

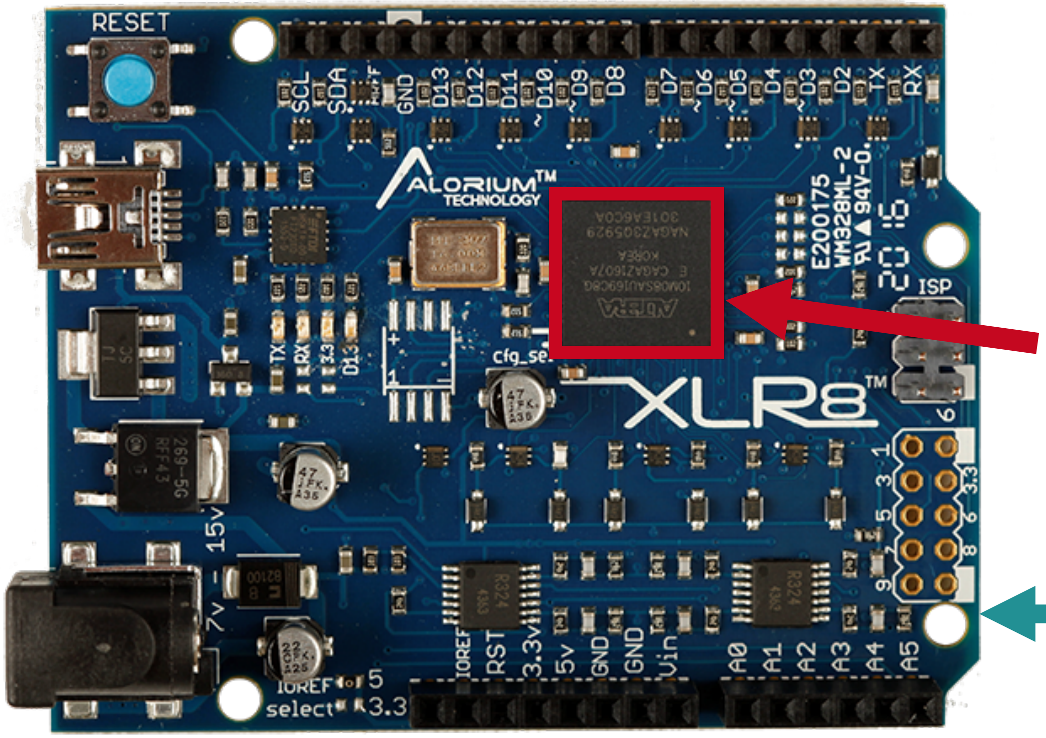


```
▼ XLR8Build
    ▼ extras
        ▶ quartus
        ▶ modelsim
        ▶ rtl
      keywords.txt
      library.properties
      LICENSE
  README.md
```

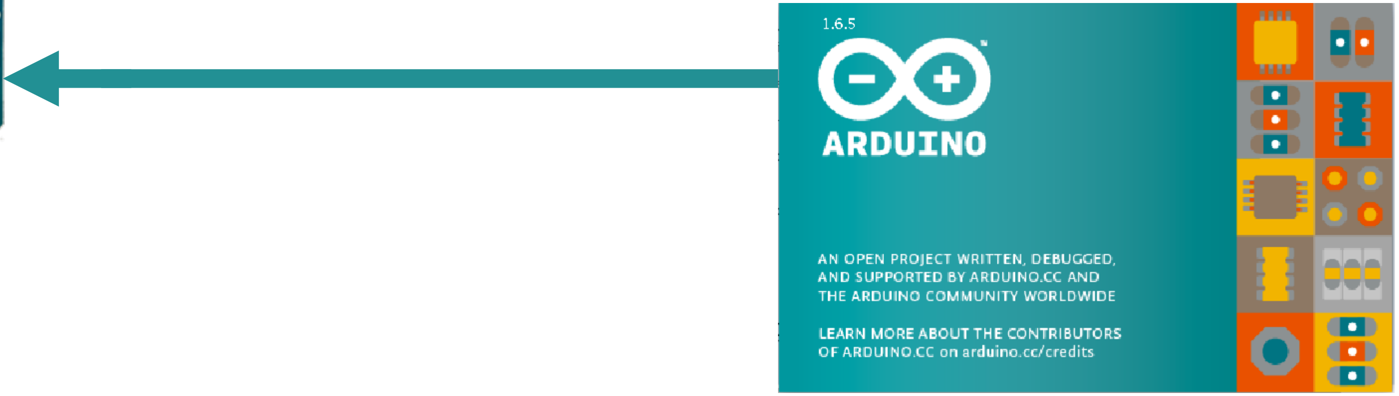
- **Arduino Board Library URL:**

https://raw.githubusercontent.com/AloriumTechnology/Arduino_Boards/master/package_aloriumtech_index.json

XLR8 Development Platform



- Application Accelerator & Development Board
- Designed for Arduino Developer Community
- Based on Intel MAX 10 FPGA
- Programmable with Arduino IDE



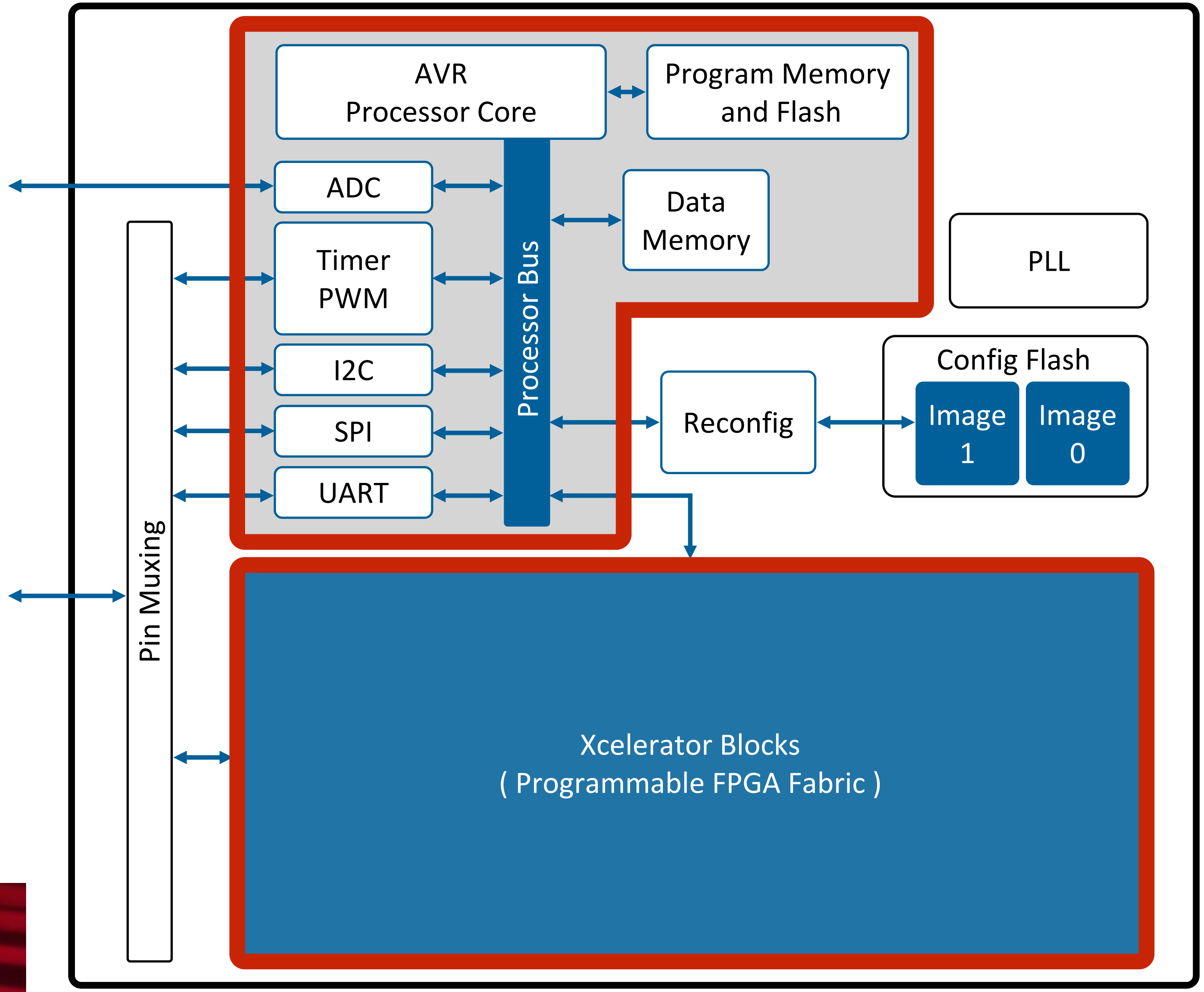
1.6.5

ARDUINO

AN OPEN PROJECT WRITTEN, DEBUGGED,
AND SUPPORTED BY ARDUINO.CC AND
THE ARDUINO COMMUNITY WORLDWIDE

LEARN MORE ABOUT THE CONTRIBUTORS
OF ARDUINO.CC on arduino.cc/credits

FPGA Block Diagram



Xcelerator Blocks

An **Xcelerator Block (XB)** is an optimized hardware implementation of a specific processor-intensive function.

Custom hardware implemented on the same FPGA fabric

Tightly integrated with the microcontroller

XBs can access the same register space

Integrate with the instructions of the microcontroller

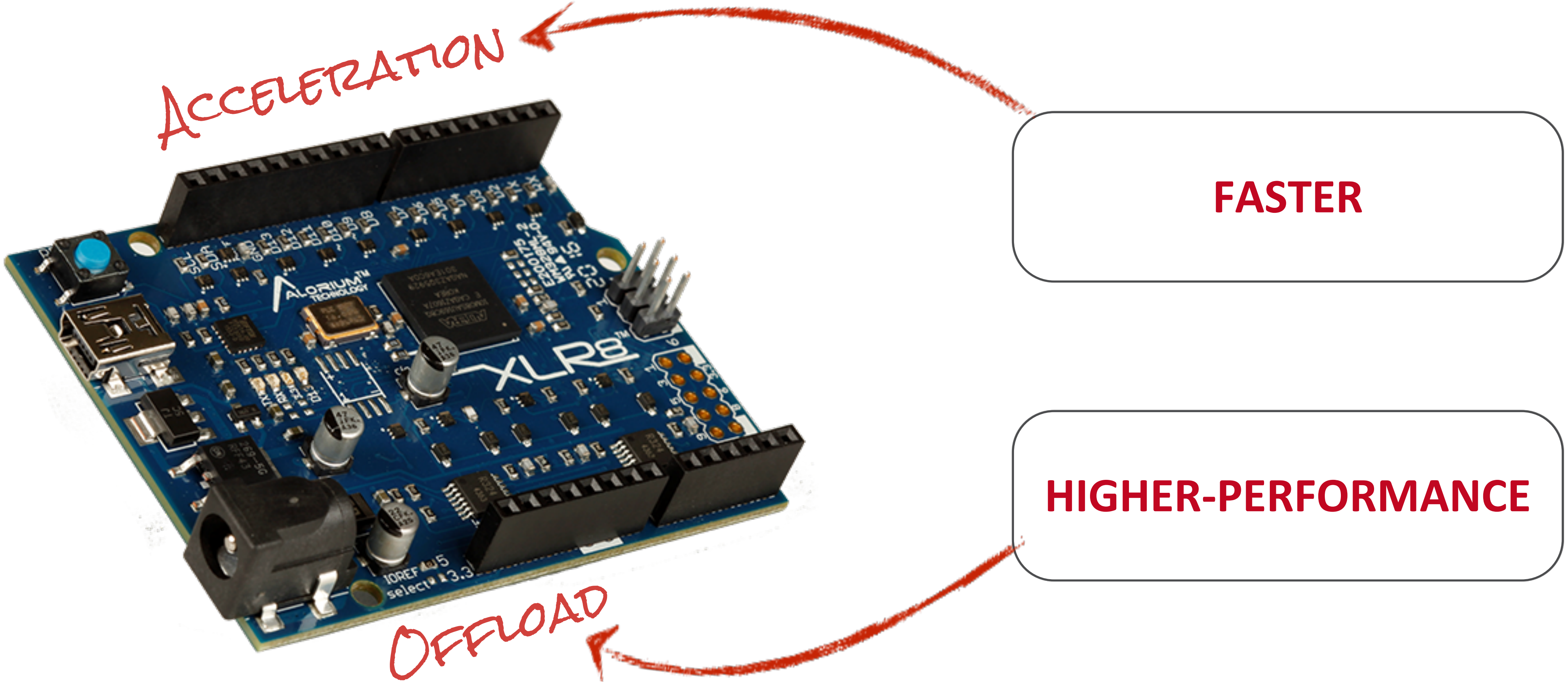
Available XBs

- Floating Point Math
- Servo Control
- NeoPixel Control
- Enhanced Analog-to-Digital Functionality
- Multiple SPI

XB Roadmap

- Proportional-Integral-Derivative (PID) control
- Event Counters and Timers
- Quadrature Encoders/Decoders
- Pulse Width Modulation (PWM)
- Multiple UARTS

Why use FPGA?



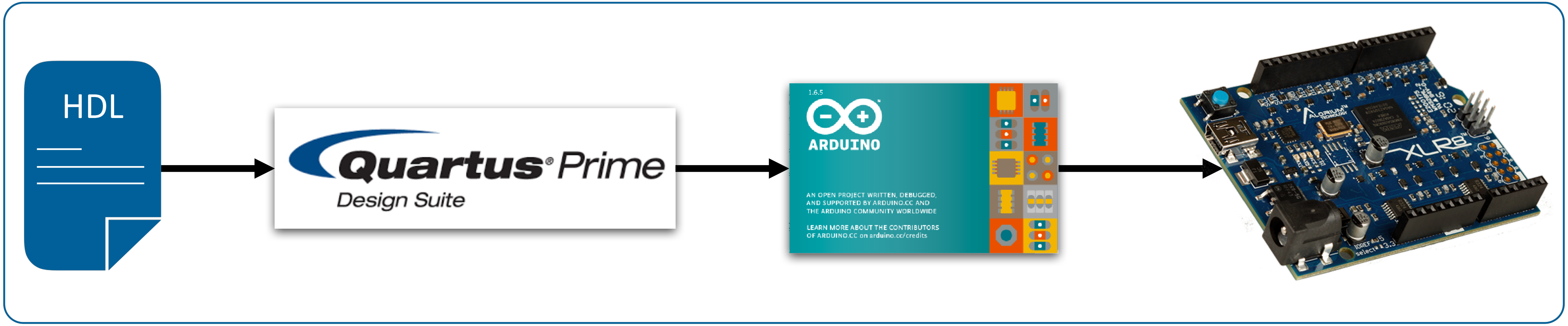
FASTER

HIGHER-PERFORMANCE

Overview of Tutorial

OpenXLR8

Methodology that allows XLR8 users to develop their own Xcelerator Blocks and upload them to the FPGA.

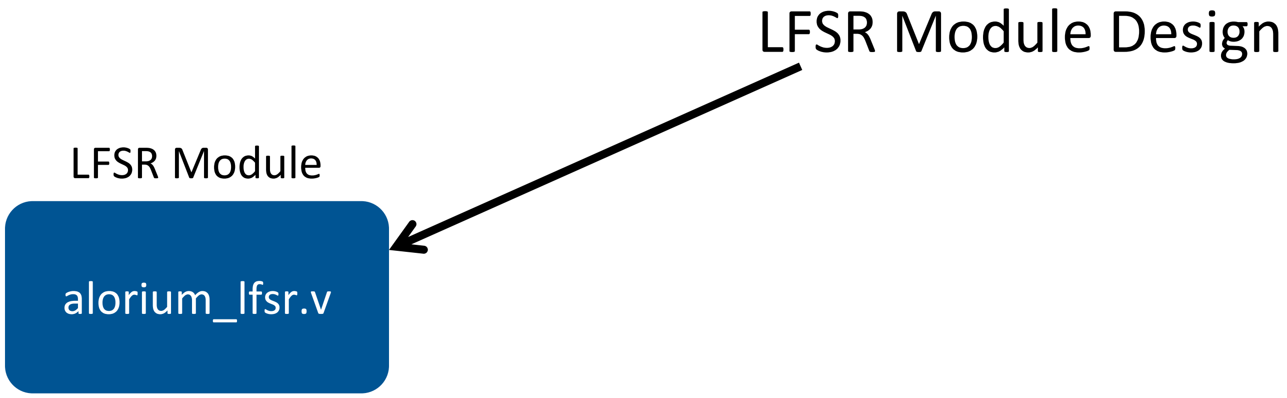


AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE

LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

Module-Level Design and Simulation

- **Pseudorandom Number Generator**
  - Using a Linear Feedback Shift Register (LFSR)
  - 8-bit
  - 4-tap

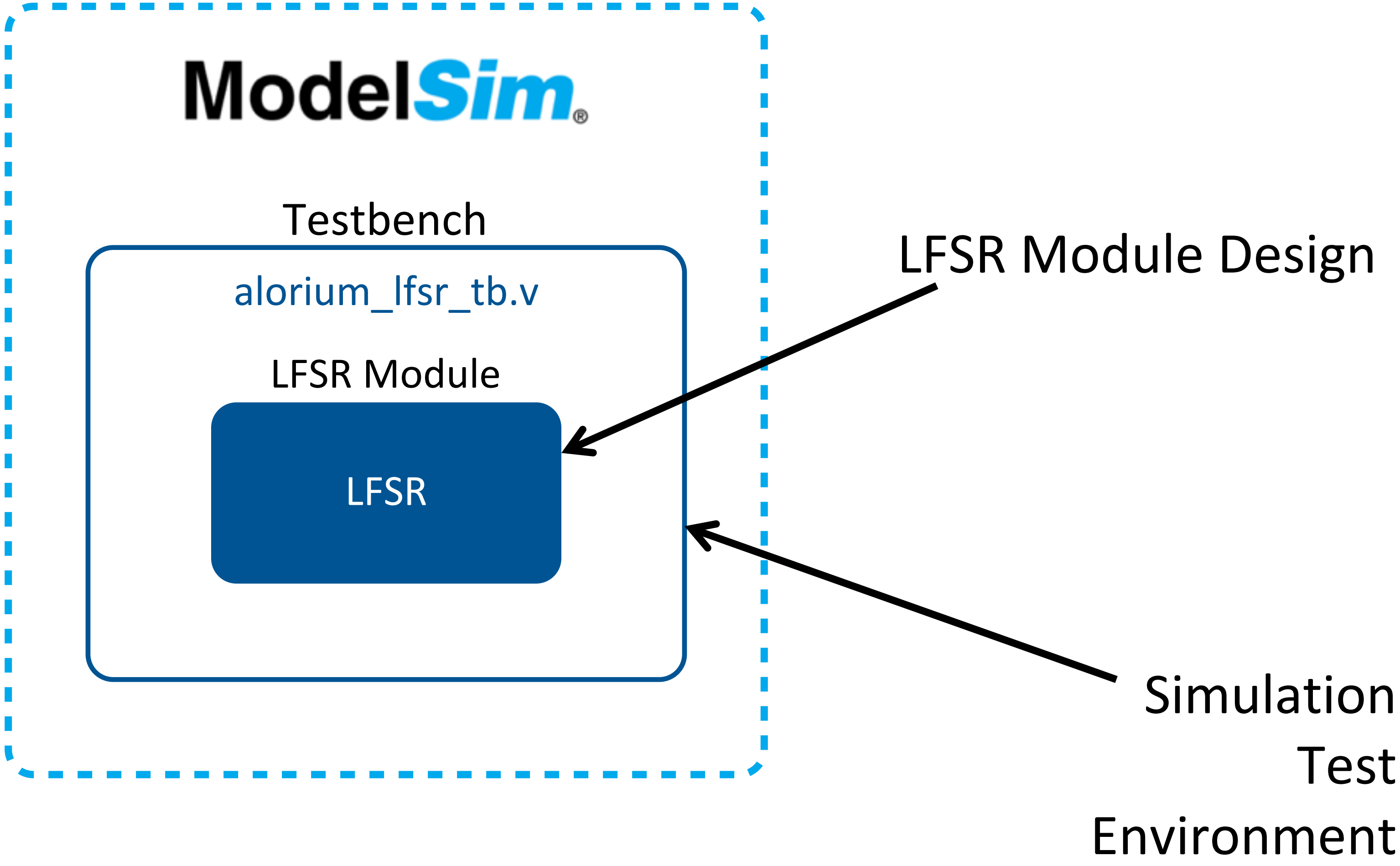


LFSR Module Design

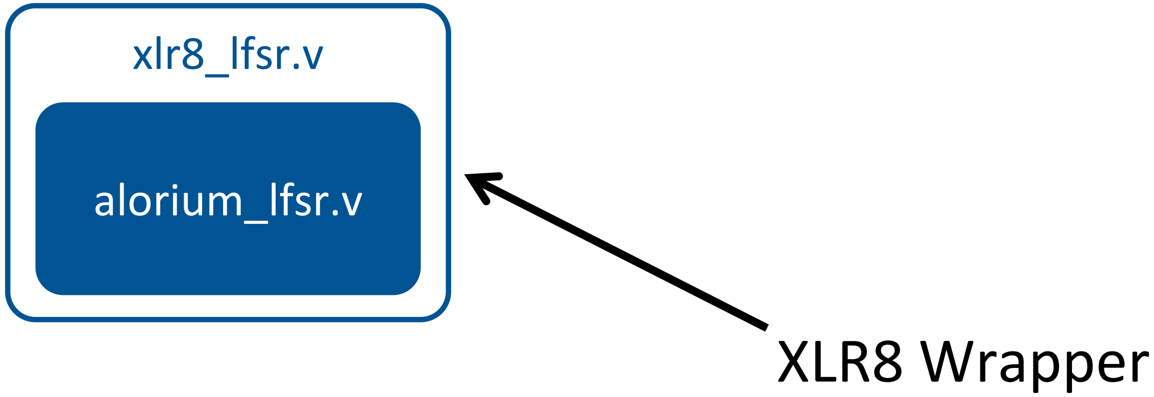
LFSR Module

alorium_lfsr.v

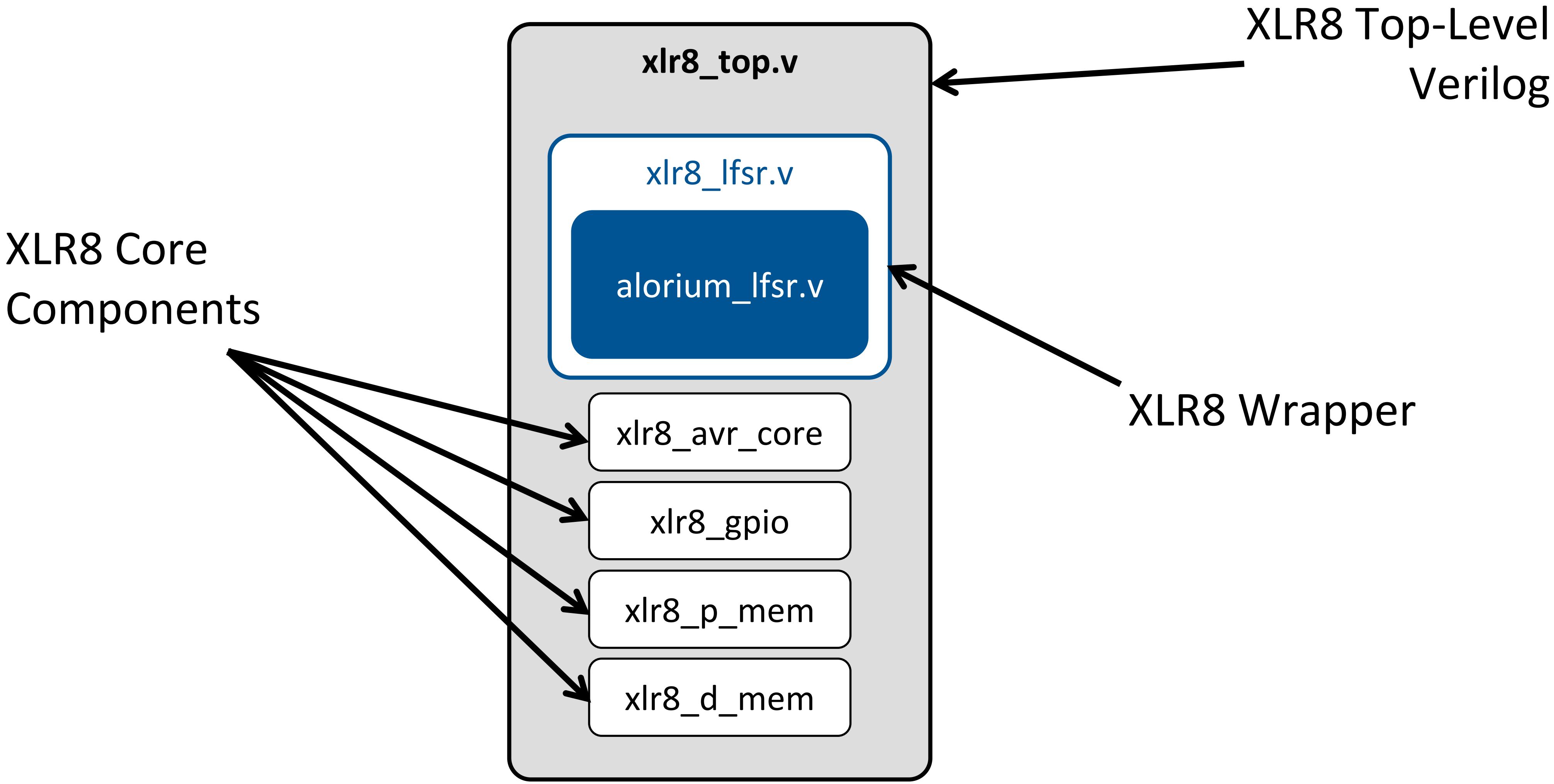
Module-Level Design and Simulation



Integration into XLR8: LFSR Wrapper



Integration into XLR8: XLR8 Top Module

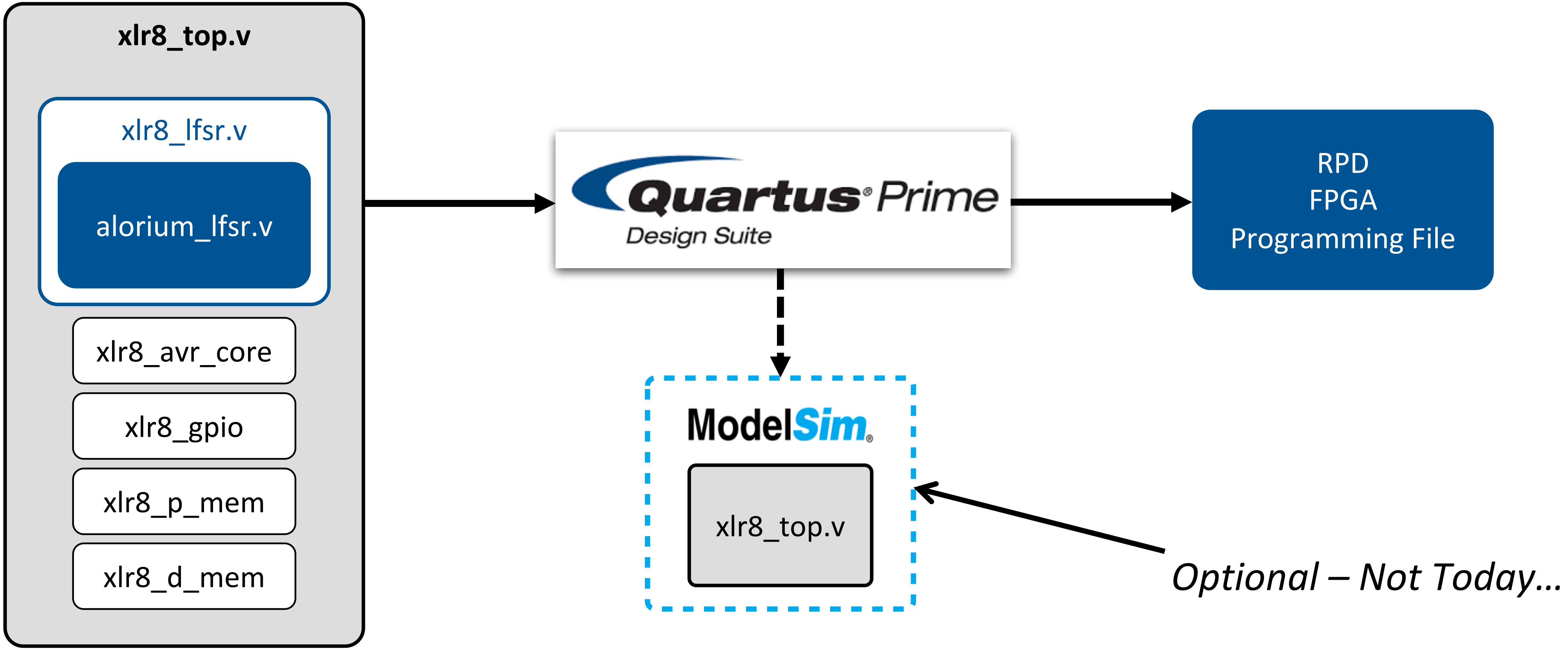


XLR8 Top-Level Verilog

XLR8 Core Components

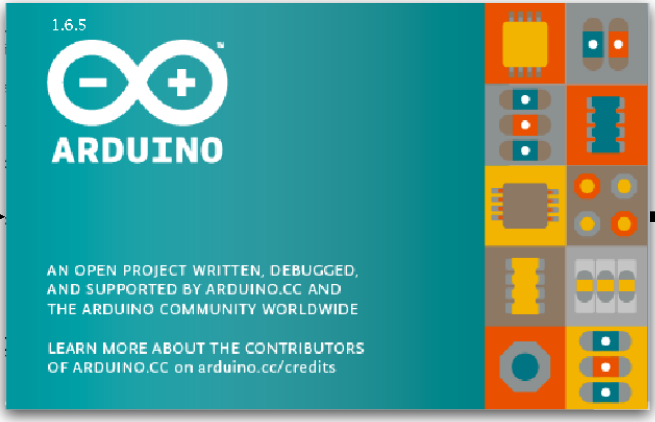
XLR8 Wrapper

Design Synthesis



Upload to FPGA

XLR8
RPD

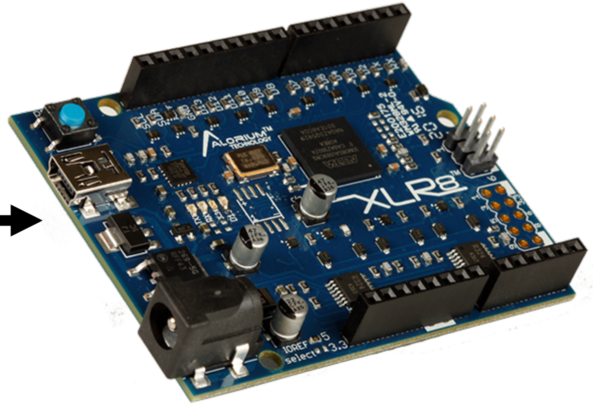


1.6.5

ARDUINO

AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE

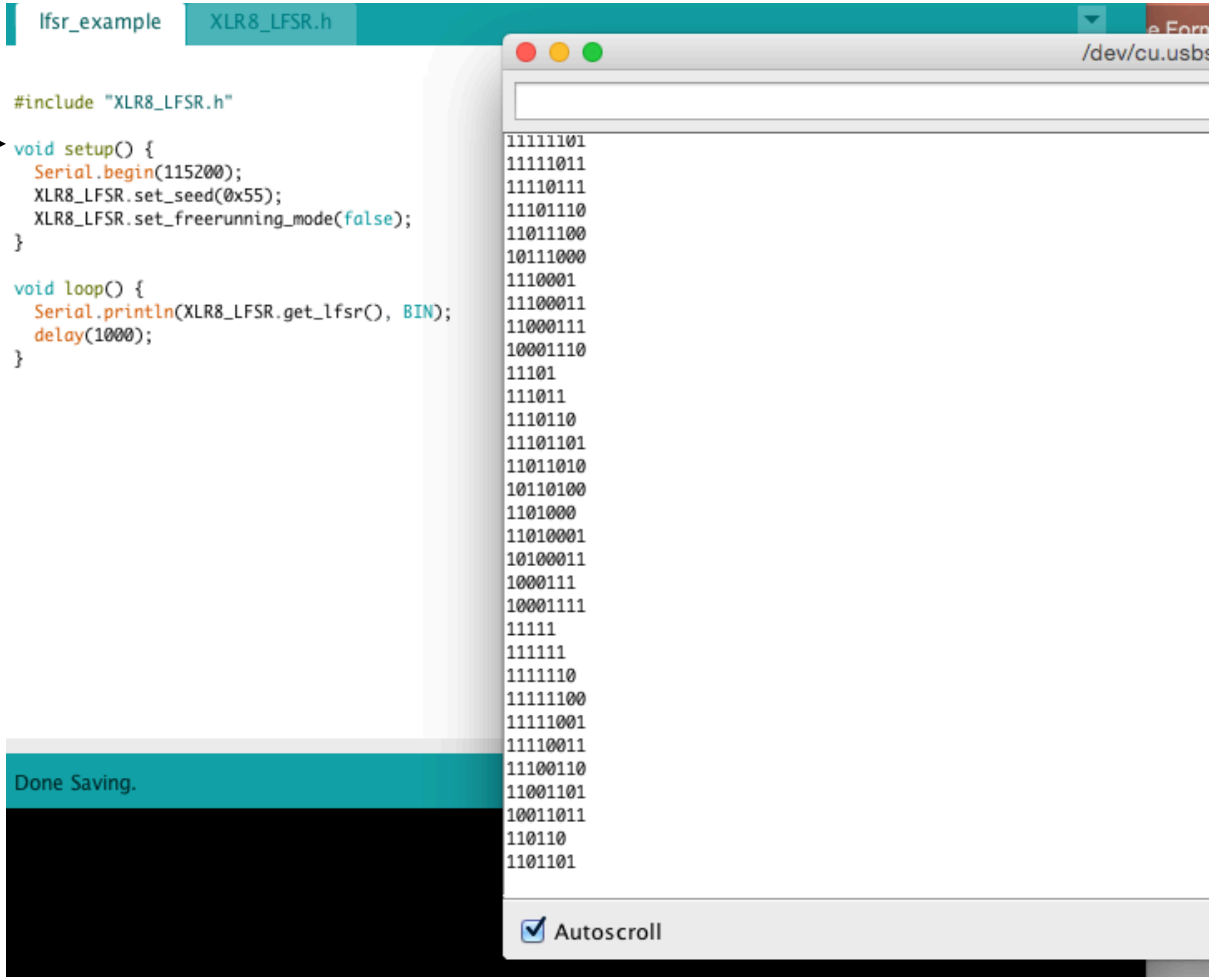
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits



Run Sketch

AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE

LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits



```
#include "XLR8_LFSR.h"

void setup() {
  Serial.begin(115200);
  XLR8_LFSR.set_seed(0x55);
  XLR8_LFSR.set_freerunning_mode(false);
}

void loop() {
  Serial.println(XLR8_LFSR.get_lfsr(), BIN);
  delay(1000);
}
```

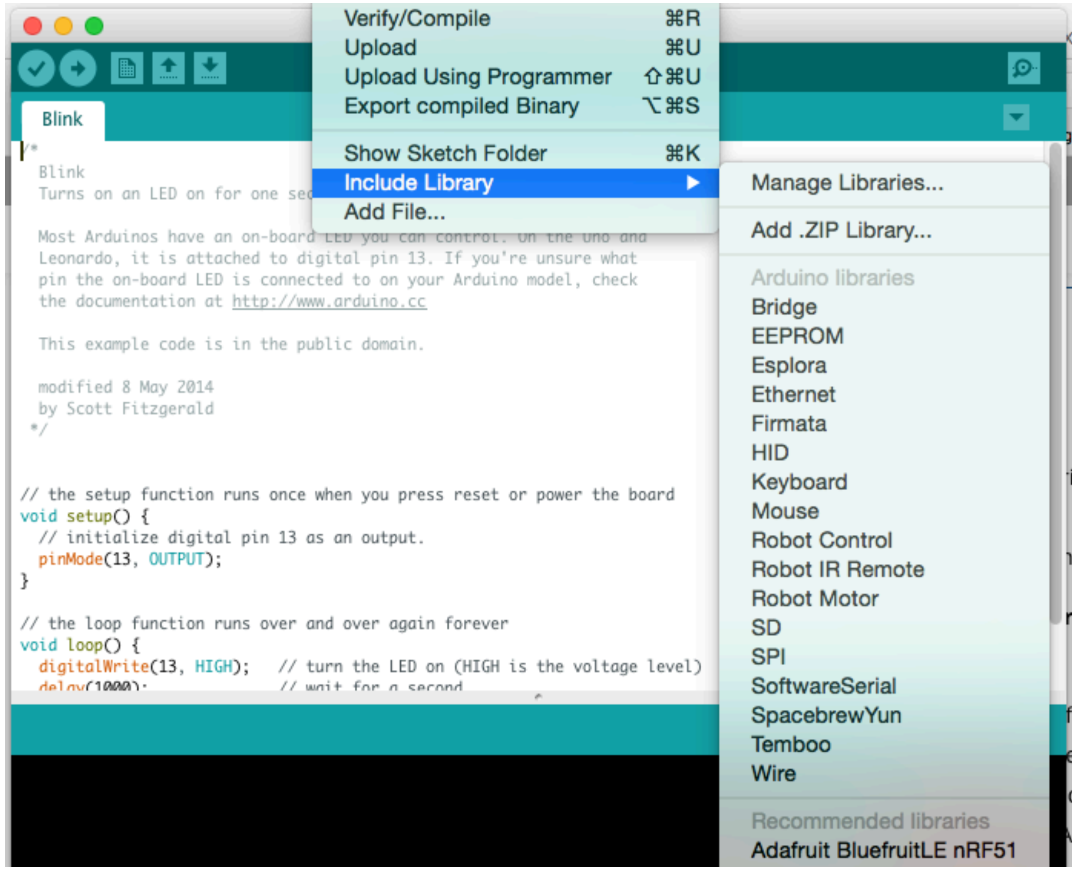
Done Saving.

```
11111101
11111011
11110111
11101110
11011100
10111000
1110001
11100011
11000111
10001110
11101
111011
1110110
11101101
11011010
10110100
1101000
11010001
10100011
1000111
10001111
11111
111111
1111110
11111100
11111001
11110011
11100110
11001101
10011011
110110
1101101
```

Let’s Dive In!

Arduino IDE Setup

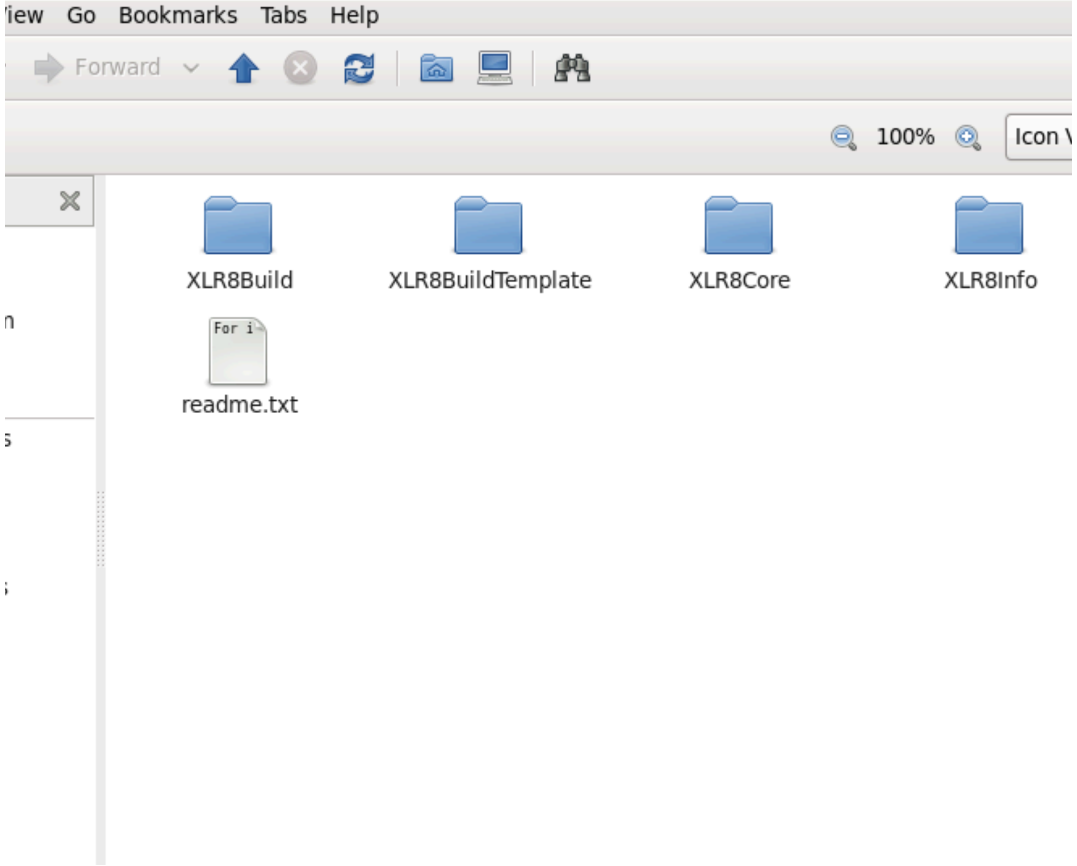
- Go to Sketch -> Include Library -> Manage Libraries…
- Search for “XLR8” and install XLR8Core and XLR8BuildTemplate
- Go to Tools -> Board -> Boards Manager…
- Search for “XLR8” and install Alorium XLR8 Boards



```
/*
  Blink
  Turns on an LED on for one sec

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
```

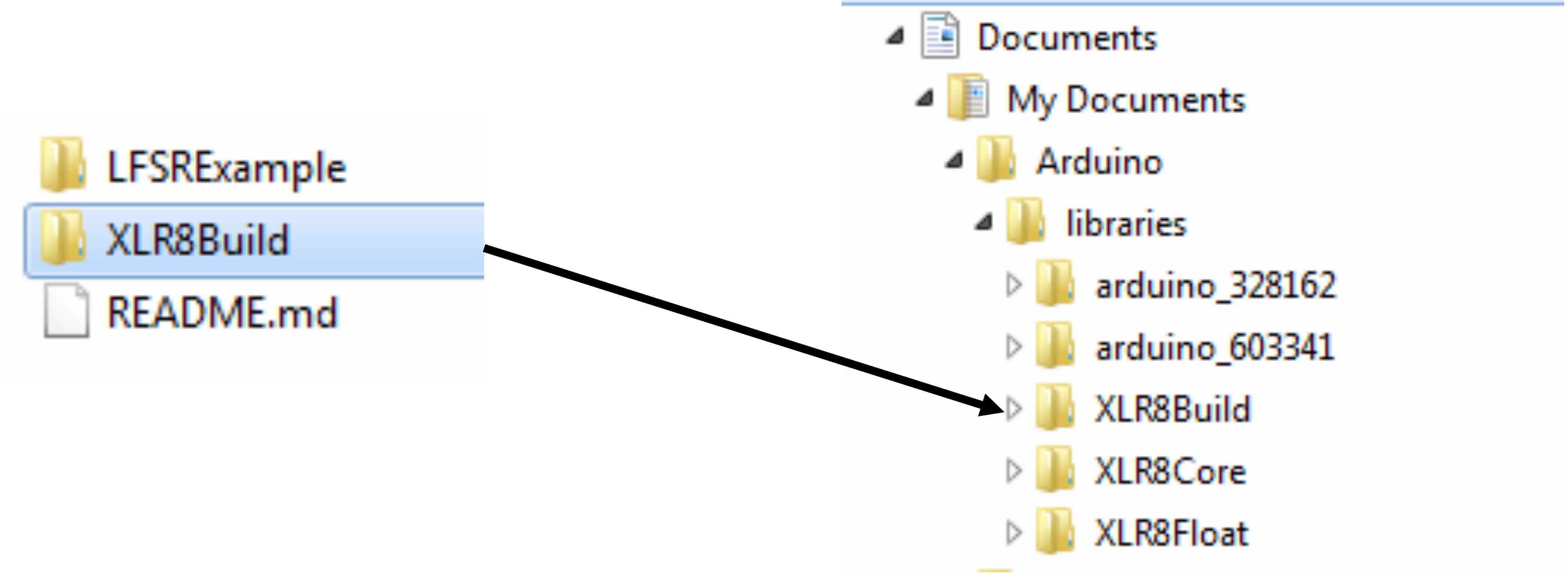
Project Directory Setup

- In your operating system's file explorer, go into your Documents directory, then Arduino/libraries
- Copy the "XLR8BuildTemplate" directory you just downloaded to a new directory named "XLR8Build"
- This will be where we build our project



But for today at ESC…

- Copy downloaded XLR8Build directory to Arduino libraries directory

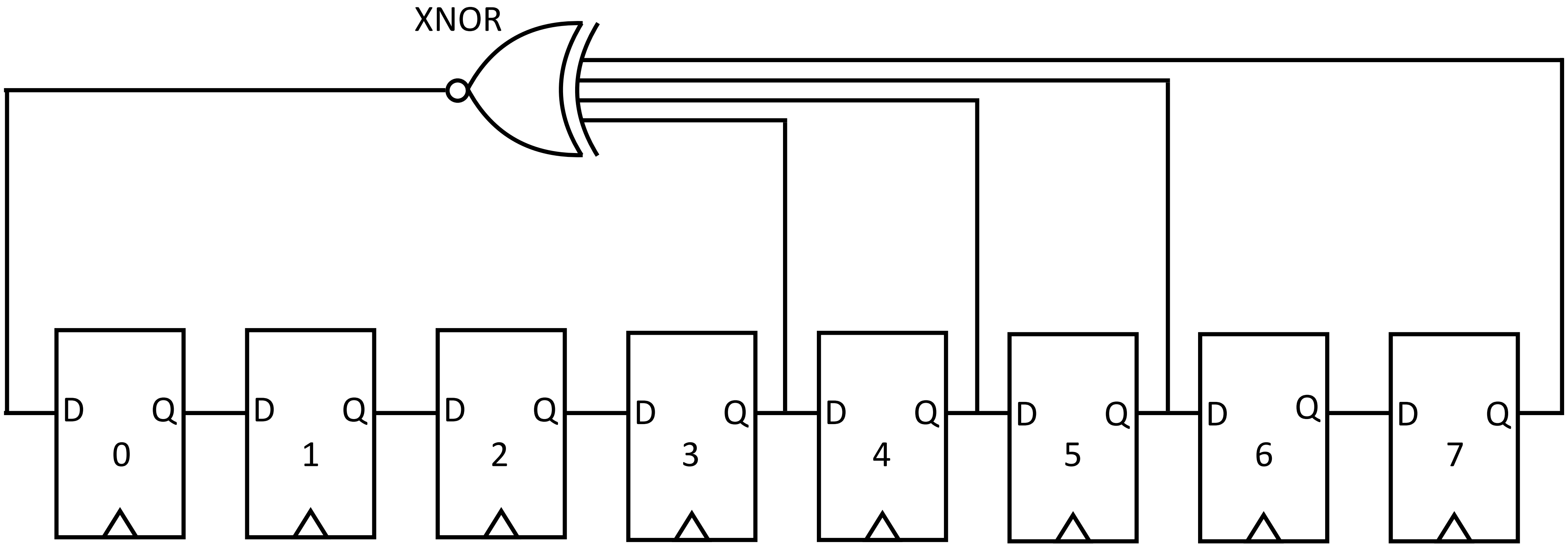


LFSRExample
XLR8Build
README.md

Documents
My Documents
Arduino
libraries
arduino_328162
arduino_603341
XLR8Build
XLR8Core
XLR8Float

Linear Feedback Shift Register

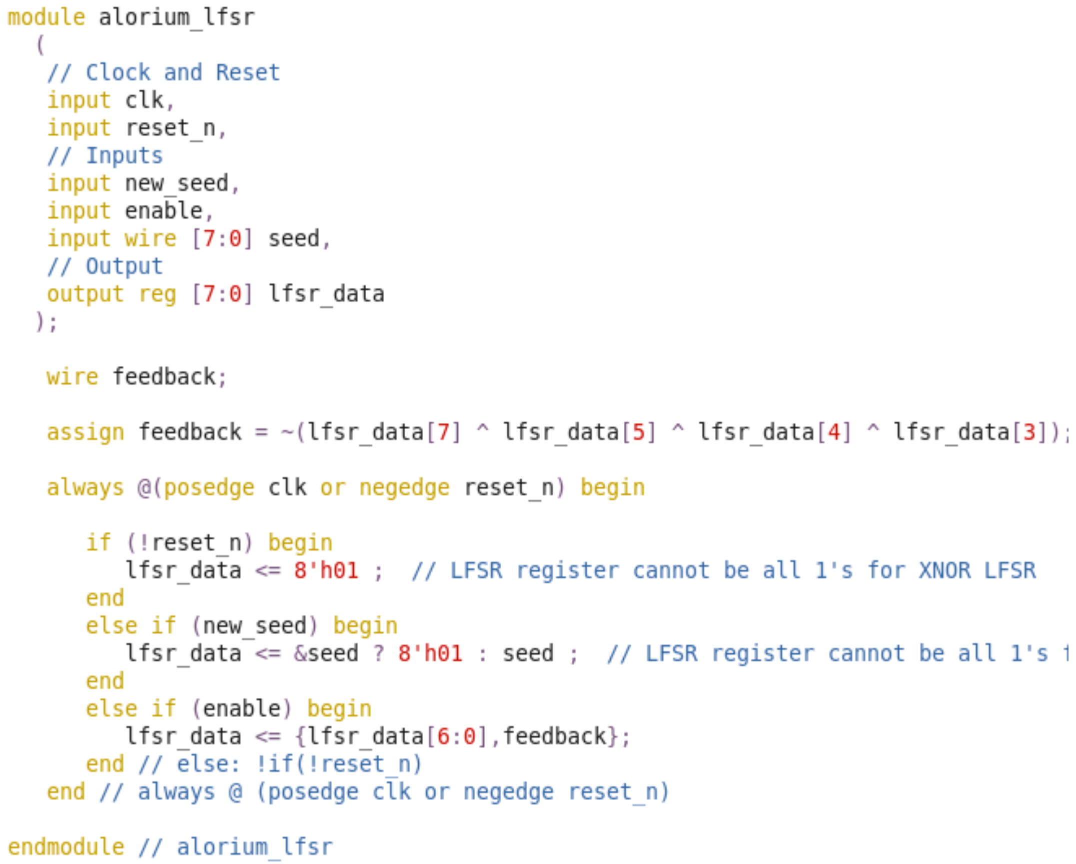
(LFSR)



```
assign feedback = ~(lfsr_data[7] ^ lfsr_data[5] ^ lfsr_data[4] ^ lfsr_data[3]);
```

RTL for the LFSR

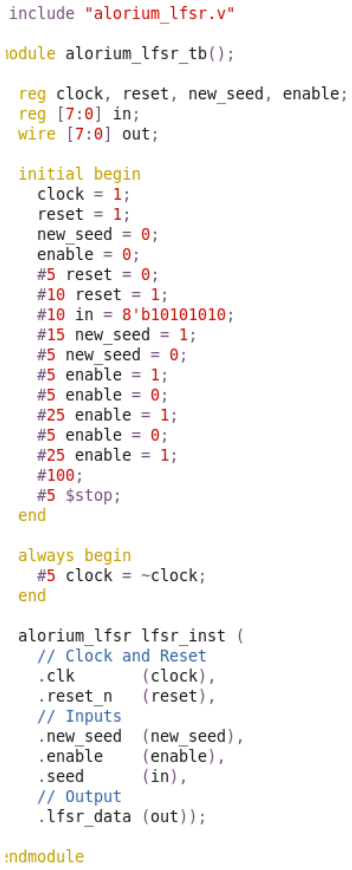
- RTL = Register-Transfer Level
  - HDL code
  - Verilog/SystemVerilog
  - VHDL
- Located in XLR8Build/extras/rtl
- The LFSR module, alorium_lfsr.v



```
module alorium_lfsr
  (
   // Clock and Reset
   input clk,
   input reset_n,
   // Inputs
   input new_seed,
   input enable,
   input wire [7:0] seed,
   // Output
   output reg [7:0] lfsr_data
  );

   wire feedback;

   assign feedback = ~(lfsr_data[7] ^ lfsr_data[5] ^ lfsr_data[4] ^ lfsr_data[3]);

   always @(posedge clk or negedge reset_n) begin

      if (!reset_n) begin
         lfsr_data <= 8'h01 ;  // LFSR register cannot be all 1's for XNOR LFSR
      end
      else if (new_seed) begin
         lfsr_data <= &seed ? 8'h01 : seed ;  // LFSR register cannot be all 1's f
      end
      else if (enable) begin
         lfsr_data <= {lfsr_data[6:0],feedback};
      end // else: !if(!reset_n)
   end // always @ (posedge clk or negedge reset_n)

endmodule // alorium_lfsr
```

Testbench

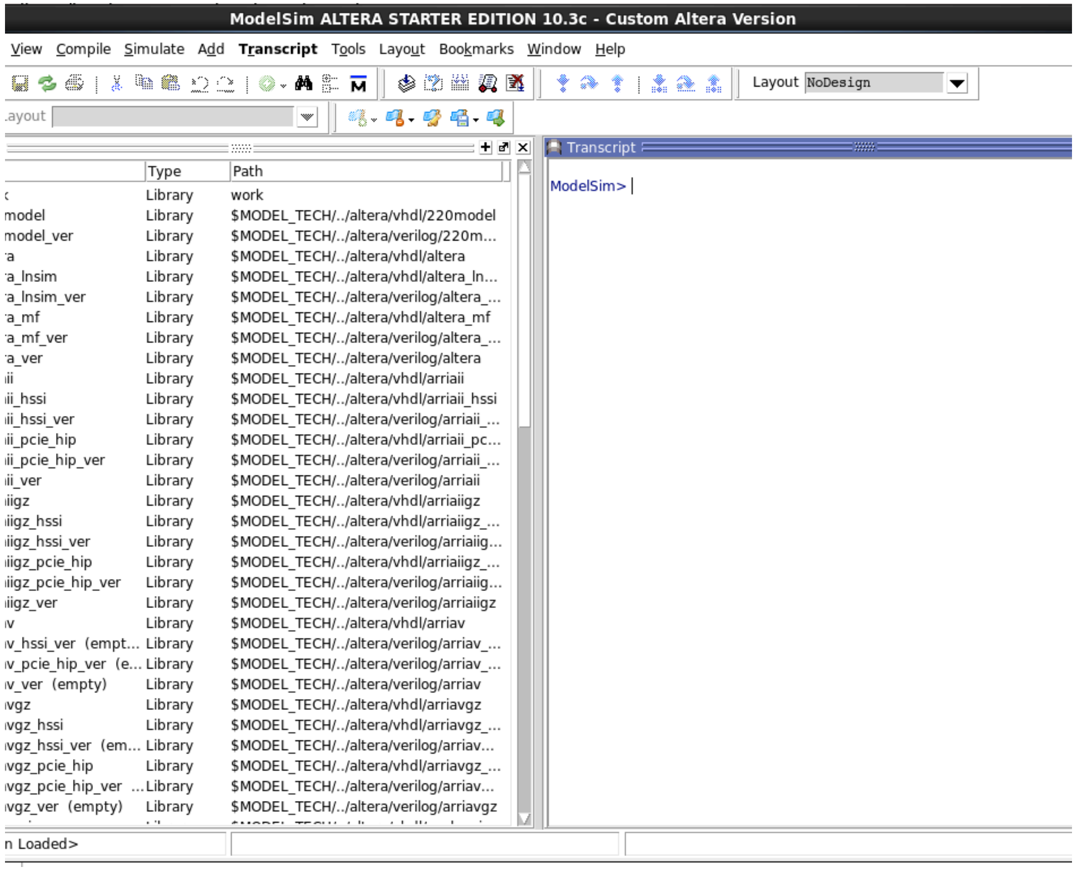
- The testbench, alorium_lfsr_tb.v



```
include "alorium_lfsr.v"

iodule alorium_lfsr_tb();

 reg clock, reset, new_seed, enable;
 reg [7:0] in;
 wire [7:0] out;

 initial begin
   clock = 1;
   reset = 1;
   new_seed = 0;
   enable = 0;
   #5 reset = 0;
   #10 reset = 1;
   #10 in = 8'b10101010;
   #15 new_seed = 1;
   #5 new_seed = 0;
   #5 enable = 1;
   #5 enable = 0;
   #25 enable = 1;
   #5 enable = 0;
   #25 enable = 1;
   #100;
   #5 $stop;
 end

 always begin
   #5 clock = ~clock;
 end

 alorium_lfsr lfsr_inst (
   // Clock and Reset
   .clk        (clock),
   .reset_n    (reset),
   // Inputs
   .new_seed   (new_seed),
   .enable     (enable),
   .seed       (in),
   // Output
   .lfsr_data  (out));

:ndmodule
```

Simulating the Testbench

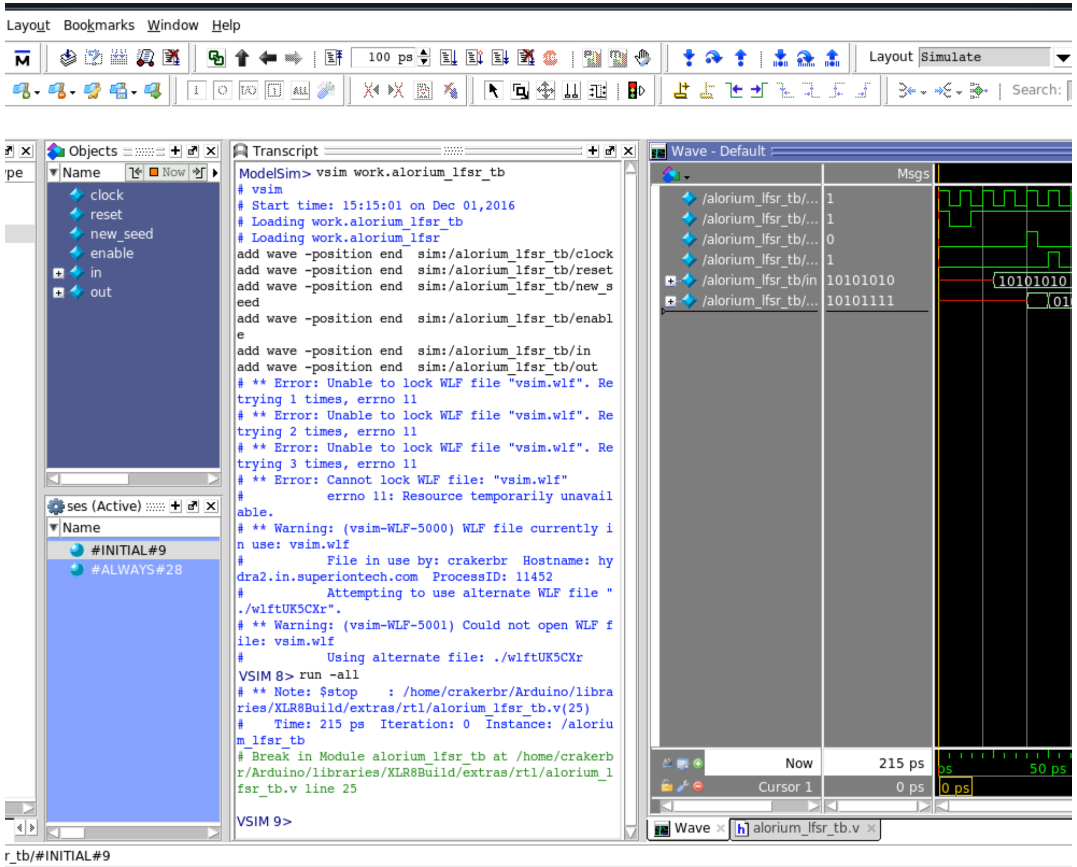
- Start Modelsim
- File -> New -> Library…
- Create the default “work” library inside of our project RTL directory
- Compile -> Compile…
- Select alorium_lfsr.v and alorium_lfsr_tb.v
- “Compile” and then “Done”
- Open the testbench in the work area



	Type	Path
(	Library	work
model	Library	$MODEL_TECH/../altera/vhdl/220model
model_ver	Library	$MODEL_TECH/../altera/verilog/220m...
'a	Library	$MODEL_TECH/../altera/vhdl/altera
'a_lnsim	Library	$MODEL_TECH/../altera/vhdl/altera_ln...
'a_lnsim_ver	Library	$MODEL_TECH/../altera/verilog/altera_...
'a_mf	Library	$MODEL_TECH/../altera/vhdl/altera_mf
'a_mf_ver	Library	$MODEL_TECH/../altera/verilog/altera_...
'a_ver	Library	$MODEL_TECH/../altera/verilog/altera
ii	Library	$MODEL_TECH/../altera/vhdl/arriaii
ii_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaii_hssi
ii_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_...
ii_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaii_pc...
ii_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog/arriaii_...
ii_ver	Library	$MODEL_TECH/../altera/verilog/arriaii
iigz	Library	$MODEL_TECH/../altera/vhdl/arriaiigz
iigz_hssi	Library	$MODEL_TECH/../altera/vhdl/arriaiigz_...
iigz_hssi_ver	Library	$MODEL_TECH/../altera/verilog/arriaiig...
iigz_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriaiigz_...
iigz_pcie_hip_ver	Library	$MODEL_TECH/../altera/verilog/arriaiig...
iigz_ver	Library	$MODEL_TECH/../altera/verilog/arriaiigz
v	Library	$MODEL_TECH/../altera/vhdl/arriav
v_hssi_ver (empt...	Library	$MODEL_TECH/../altera/verilog/arriav_...
v_pcie_hip_ver (e...	Library	$MODEL_TECH/../altera/verilog/arriav_...
v_ver (empty)	Library	$MODEL_TECH/../altera/verilog/arriav
vgz	Library	$MODEL_TECH/../altera/vhdl/arriavgz
vgz_hssi	Library	$MODEL_TECH/../altera/vhdl/arriavgz_...
vgz_hssi_ver (em...	Library	$MODEL_TECH/../altera/verilog/arriav...
vgz_pcie_hip	Library	$MODEL_TECH/../altera/vhdl/arriavgz_...
vgz_pcie_hip_ver ...	Library	$MODEL_TECH/../altera/verilog/arriav...
vgz_ver (empty)	Library	$MODEL_TECH/../altera/verilog/arriavgz

Simulating the Testbench Continued

- Select our testbench signals and bring them into a waves window
- Hit the “Run –all” button

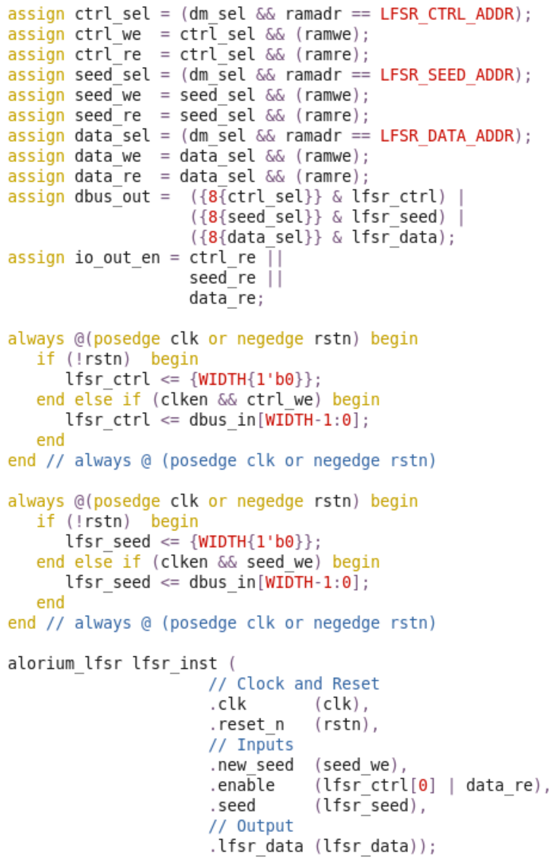


```
ModelSim> vsim work.alorium_lfsr_tb
# vsim
# Start time: 15:15:01 on Dec 01,2016
# Loading work.alorium_lfsr_tb
# Loading work.alorium_lfsr
add wave -position end  sim:/alorium_lfsr_tb/clock
add wave -position end  sim:/alorium_lfsr_tb/reset
add wave -position end  sim:/alorium_lfsr_tb/new_s
eed
add wave -position end  sim:/alorium_lfsr_tb/enabl
e
add wave -position end  sim:/alorium_lfsr_tb/in
add wave -position end  sim:/alorium_lfsr_tb/out
# ** Error: Unable to lock WLF file "vsim.wlf". Re
trying 1 times, errno 11
# ** Error: Unable to lock WLF file "vsim.wlf". Re
trying 2 times, errno 11
# ** Error: Unable to lock WLF file "vsim.wlf". Re
trying 3 times, errno 11
# ** Error: Cannot lock WLF file: "vsim.wlf"
#          errno 11: Resource temporarily unavail
able.
# ** Warning: (vsim-WLF-5000) WLF file currently i
n use: vsim.wlf
#          File in use by: crakerbr  Hostname: hy
dra2.in.superiontech.com  ProcessID: 11452
#          Attempting to use alternate WLF file "
./wlftUK5CXr".
# ** Warning: (vsim-WLF-5001) Could not open WLF f
ile: vsim.wlf
#          Using alternate file: ./wlftUK5CXr
VSIM 8> run -all
# ** Note: $stop    : /home/crakerbr/Arduino/libra
ries/XLR8Build/extras/rtl/alorium_lfsr_tb.v(25)
#    Time: 215 ps  Iteration: 0  Instance: /aloriu
m_lfsr_tb
# Break in Module alorium_lfsr_tb at /home/crakerb
r/Arduino/libraries/XLR8Build/extras/rtl/alorium_l
fsr_tb.v line 25

VSIM 9>
```

```
/alorium_lfsr_tb/...  1
/alorium_lfsr_tb/...  1
/alorium_lfsr_tb/...  0
/alorium_lfsr_tb/...  1
/alorium_lfsr_tb/in   10101010
/alorium_lfsr_tb/...  10101111

Now       215 ps
Cursor 1  0 ps
```

XLR8 Module

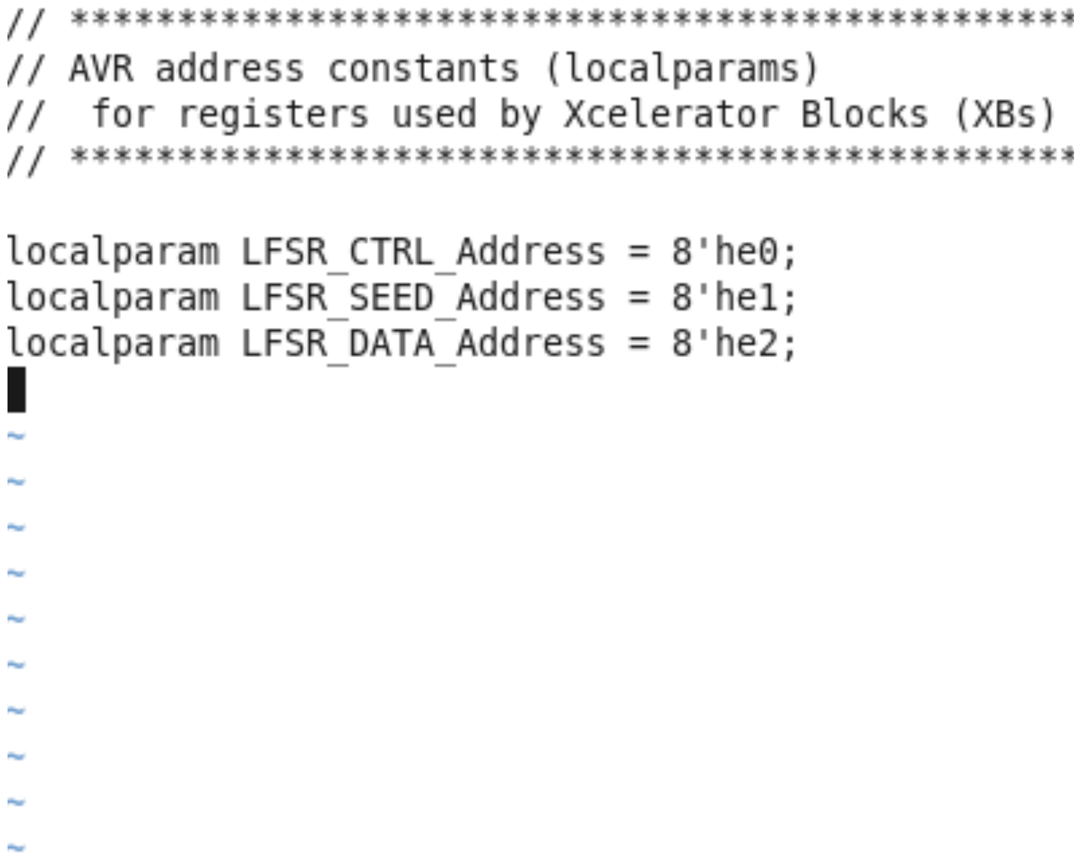
- xlr8_lfsr.v
- Connects the signals from the XLR8 core to the LFSR module
- Instantiates the alorium_lfsr module
- Controls register access



```verilog
assign ctrl_sel = (dm_sel && ramadr == LFSR_CTRL_ADDR);
assign ctrl_we  = ctrl_sel && (ramwe);
assign ctrl_re  = ctrl_sel && (ramre);
assign seed_sel = (dm_sel && ramadr == LFSR_SEED_ADDR);
assign seed_we  = seed_sel && (ramwe);
assign seed_re  = seed_sel && (ramre);
assign data_sel = (dm_sel && ramadr == LFSR_DATA_ADDR);
assign data_we  = data_sel && (ramwe);
assign data_re  = data_sel && (ramre);
assign dbus_out =  ({8{ctrl_sel}} & lfsr_ctrl) |
                   ({8{seed_sel}} & lfsr_seed) |
                   ({8{data_sel}} & lfsr_data);
assign io_out_en = ctrl_re ||
                   seed_re ||
                   data_re;

always @(posedge clk or negedge rstn) begin
   if (!rstn)  begin
      lfsr_ctrl <= {WIDTH{1'b0}};
   end else if (clken && ctrl_we) begin
      lfsr_ctrl <= dbus_in[WIDTH-1:0];
   end
end // always @ (posedge clk or negedge rstn)

always @(posedge clk or negedge rstn) begin
   if (!rstn)  begin
      lfsr_seed <= {WIDTH{1'b0}};
   end else if (clken && seed_we) begin
      lfsr_seed <= dbus_in[WIDTH-1:0];
   end
end // always @ (posedge clk or negedge rstn)

alorium_lfsr lfsr_inst (
                     // Clock and Reset
                     .clk       (clk),
                     .reset_n   (rstn),
                     // Inputs
                     .new_seed  (seed_we),
                     .enable    (lfsr_ctrl[0] | data_re),
                     .seed      (lfsr_seed),
                     // Output
                     .lfsr_data (lfsr_data));
```

XB Addresses

- xb_adr_pack.vh
- Declare the address locations of your registers
- Refer to the XLR8 User Manual to find open register space



```
// ***********************************************
// AVR address constants (localparams)
//  for registers used by Xcelerator Blocks (XBs)
// ***********************************************

localparam LFSR_CTRL_Address = 8'he0;
localparam LFSR_SEED_Address = 8'he1;
localparam LFSR_DATA_Address = 8'he2;

~
~
~
~
~
~
~
~
~
~
```

LFSR Register Descriptions

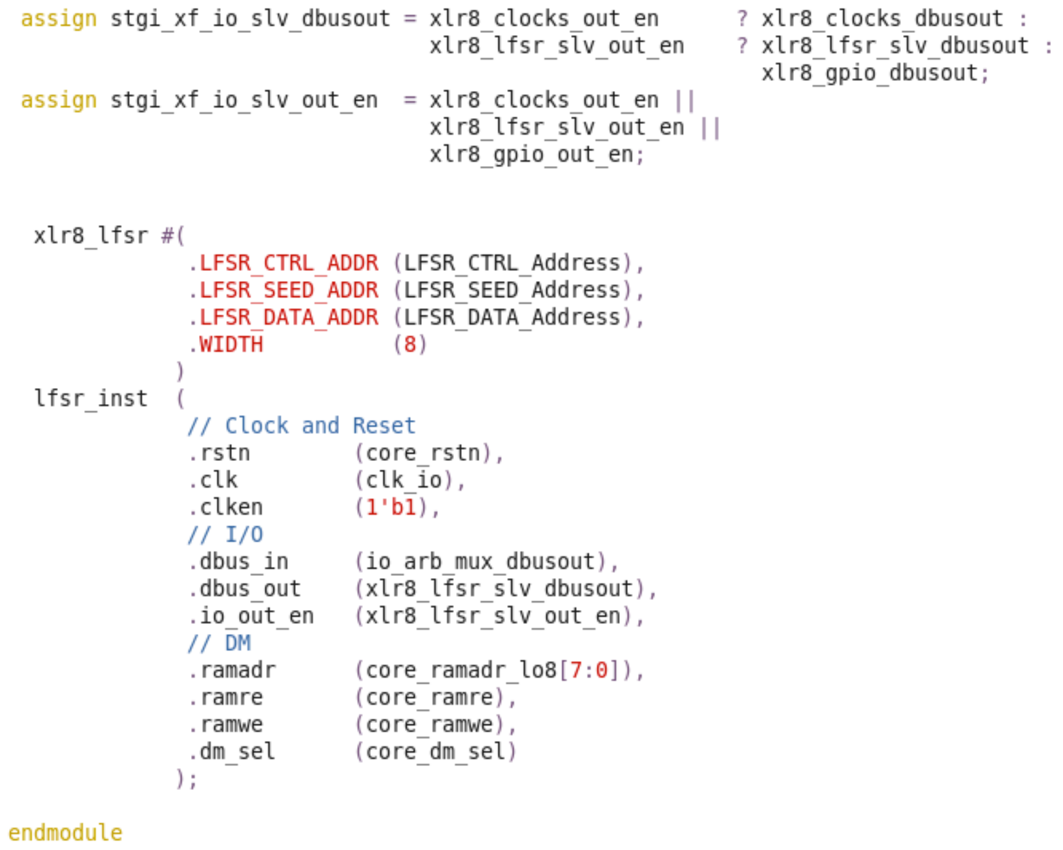
LFSR Control					Address 0xE0			
Bit	7	6	5	4	3	2	1	0
Function	Unused							Freerunning Mode
R/W	R	R	R	R	R	R	R	R/W
Initial	0	0	0	0	0	0	0	0

LFSR Seed					Address 0xE1			
Bit	7	6	5	4	3	2	1	0
Function	LFSR Seed Data							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial	0	0	0	0	0	0	0	0

LFSR Data					Address 0xE2			
Bit	7	6	5	4	3	2	1	0
Function	LFSR Result Data							
R/W	R	R	R	R	R	R	R	R
Initial	0	0	0	0	0	0	0	0

XLR8 Top

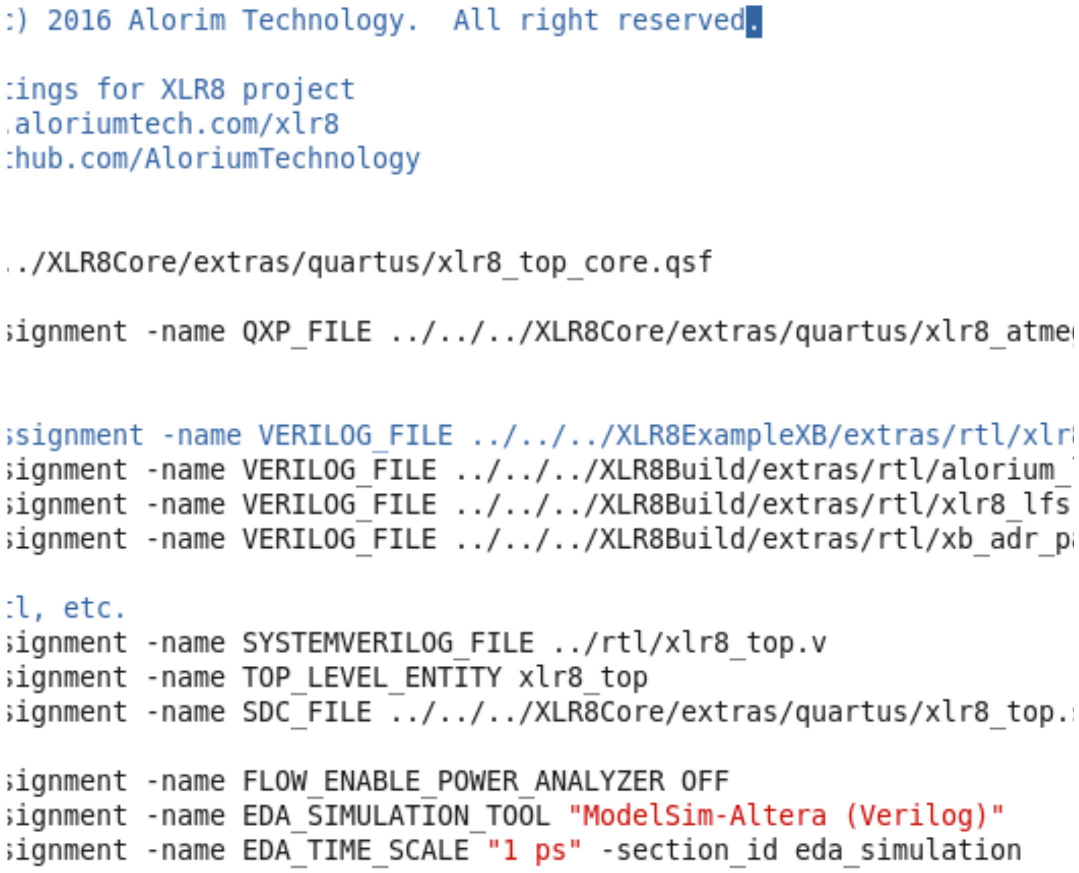
- xlr8_top.v
- Instantiate the xlr8_lfsr module
- Add the control signals to “stgi_xf_io_slv_dbusout” and “stgi_xf_io_slv_out_en”
- Don’t forget to declare your control signals



```
 assign stgi_xf_io_slv_dbusout = xlr8_clocks_out_en      ? xlr8_clocks_dbusout :
                                 xlr8_lfsr_slv_out_en    ? xlr8_lfsr_slv_dbusout :
                                                           xlr8_gpio_dbusout;

 assign stgi_xf_io_slv_out_en  = xlr8_clocks_out_en ||
                                 xlr8_lfsr_slv_out_en ||
                                 xlr8_gpio_out_en;


  xlr8_lfsr #(
             .LFSR_CTRL_ADDR (LFSR_CTRL_Address),
             .LFSR_SEED_ADDR (LFSR_SEED_Address),
             .LFSR_DATA_ADDR (LFSR_DATA_Address),
             .WIDTH          (8)
            )
  lfsr_inst (
             // Clock and Reset
             .rstn        (core_rstn),
             .clk         (clk_io),
             .clken       (1'b1),
             // I/O
             .dbus_in     (io_arb_mux_dbusout),
             .dbus_out    (xlr8_lfsr_slv_dbusout),
             .io_out_en   (xlr8_lfsr_slv_out_en),
             // DM
             .ramadr      (core_ramadr_lo8[7:0]),
             .ramre       (core_ramre),
             .ramwe       (core_ramwe),
             .dm_sel      (core_dm_sel)
            );

endmodule
```

Modify the Project QSF File

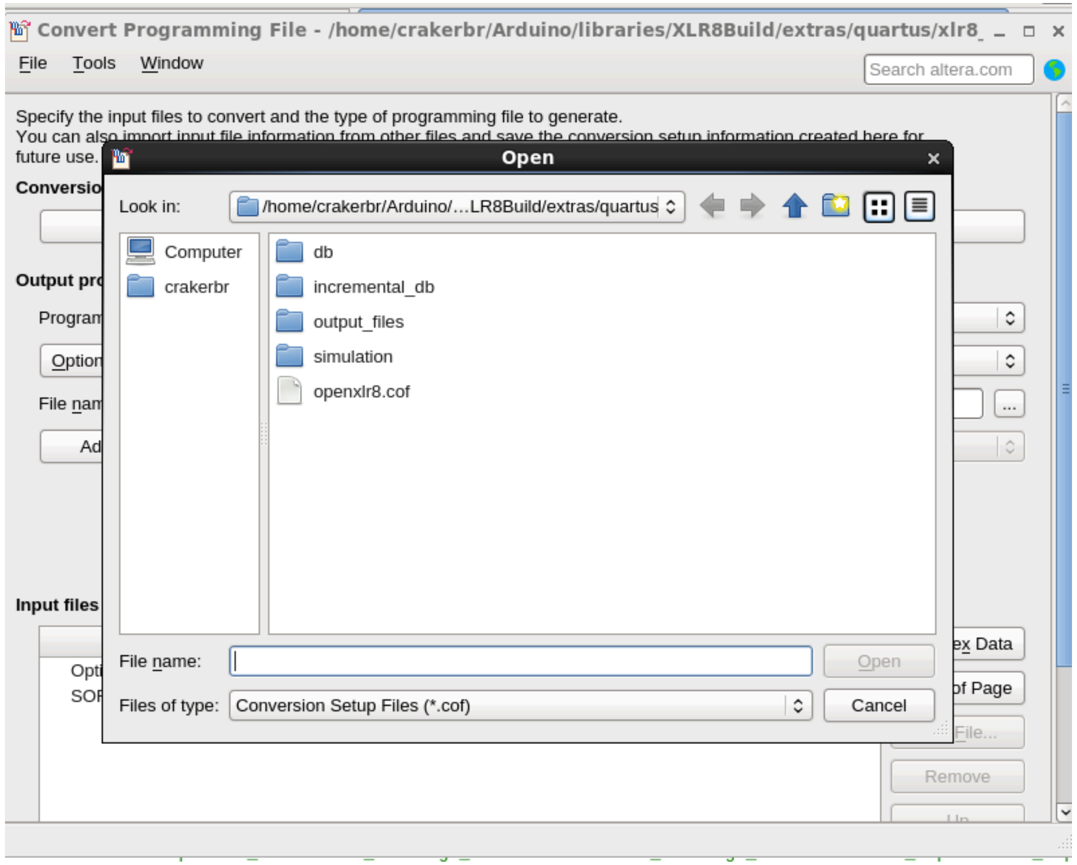
- xlr8_top.qsf under the “quartus” directory
- Add in our module files and the register address file



```
c) 2016 Alorim Technology.  All right reserved.

tings for XLR8 project
.aloriumtech.com/xlr8
thub.com/AloriumTechnology


../XLR8Core/extras/quartus/xlr8_top_core.qsf

signment -name QXP_FILE ../../../XLR8Core/extras/quartus/xlr8_atme


ssignment -name VERILOG_FILE ../../../XLR8ExampleXB/extras/rtl/xlr
signment -name VERILOG_FILE ../../../XLR8Build/extras/rtl/alorium_
signment -name VERILOG_FILE ../../../XLR8Build/extras/rtl/xlr8_lfs
signment -name VERILOG_FILE ../../../XLR8Build/extras/rtl/xb_adr_p

tl, etc.
signment -name SYSTEMVERILOG_FILE ../rtl/xlr8_top.v
signment -name TOP_LEVEL_ENTITY xlr8_top
signment -name SDC_FILE ../../../XLR8Core/extras/quartus/xlr8_top.

signment -name FLOW_ENABLE_POWER_ANALYZER OFF
signment -name EDA_SIMULATION_TOOL "ModelSim-Altera (Verilog)"
signment -name EDA_TIME_SCALE "1 ps" -section_id eda_simulation
```

Compile the Project in Quartus

- Open Quartus and open our project QSF file with File -> Open Project…
- Begin the compile with Processing -> Start Compilation
- After compilation is completed, File -> Convert Programming Files…
- Open Conversion Setup Data, open “openxlr8.cof,” and Generate



Convert Programming File - /home/crakerbr/Arduino/libraries/XLR8Build/extras/quartus/xlr8_

Specify the input files to convert and the type of programming file to generate. You can also import input file information from other files and save the conversion setup information created here for future use.

Open

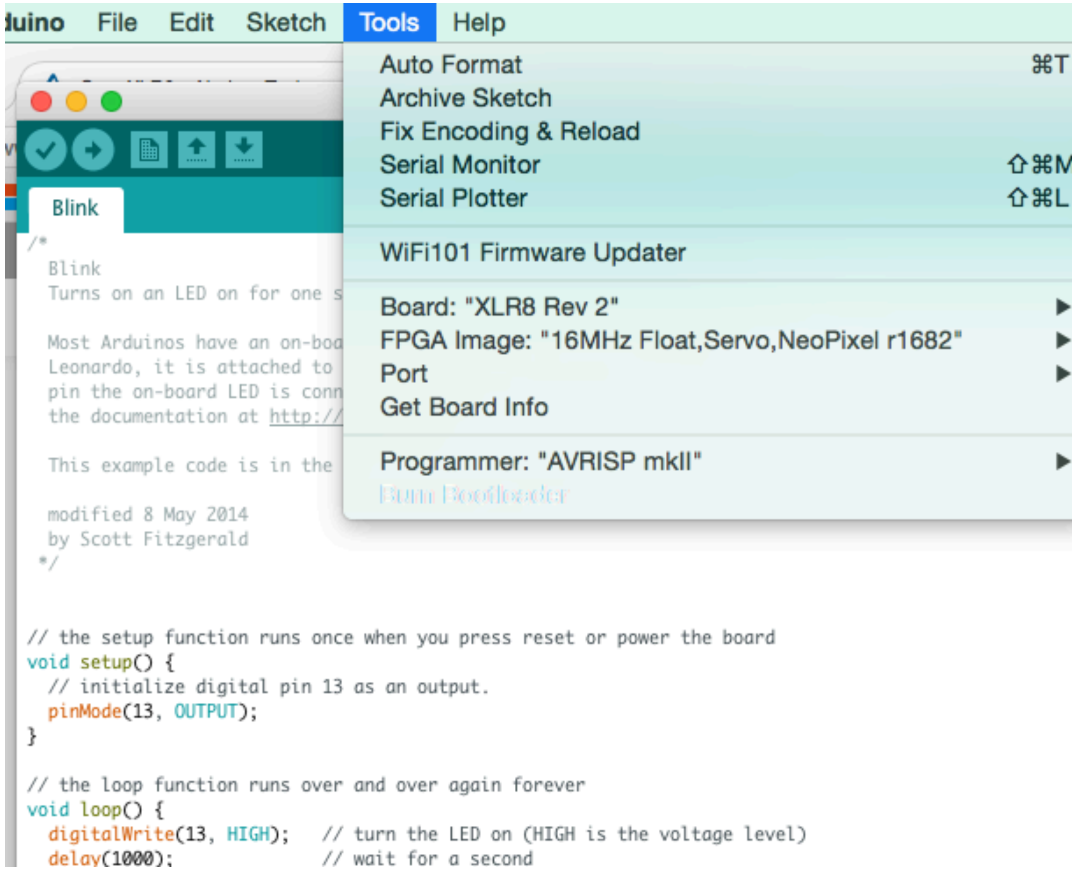
Look in: /home/crakerbr/Arduino/…LR8Build/extras/quartus

File name:

Files of type: Conversion Setup Files (*.cof)

Burn the FPGA Image

- Open the Arduino IDE
- Under Tools -> Board select OpenXLR8
- Connect your board via USB and make sure it is selected in Arduino under Tools -> Port
- Tools -> Burn Bootloader



Board: "XLR8 Rev 2" ▶

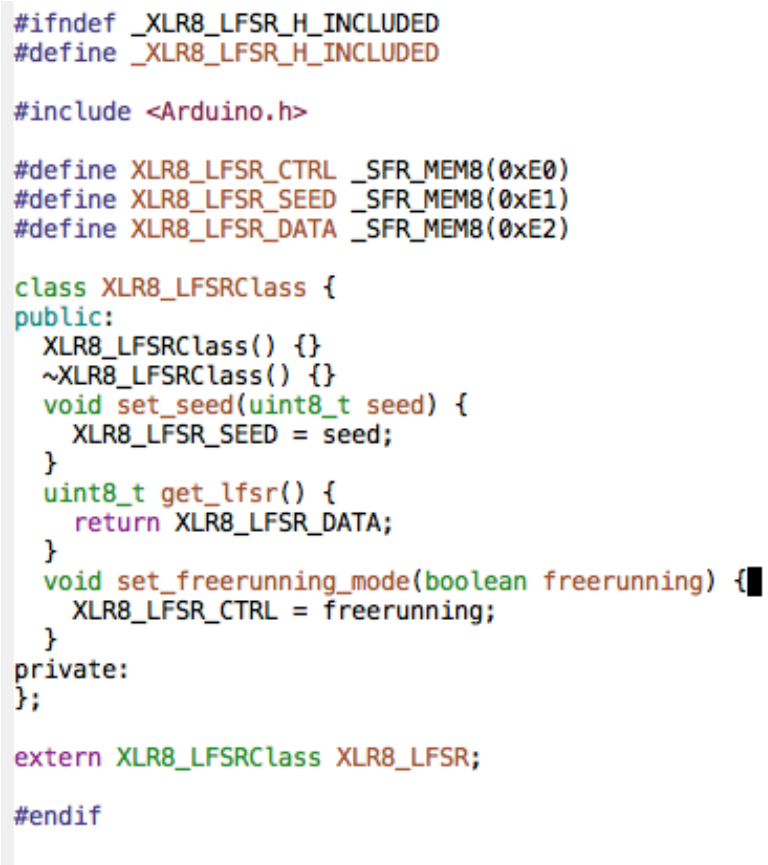
FPGA Image: "16MHz Float,Servo,NeoPixel r1682" ▶

Programmer: "AVRISP mkII" ▶

```
/*
  Blink
  Turns on an LED on for one s

  Most Arduinos have an on-boa
  Leonardo, it is attached to
  pin the on-board LED is conn
  the documentation at http://

  This example code is in the

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
```

Arduino Library for the LFSR

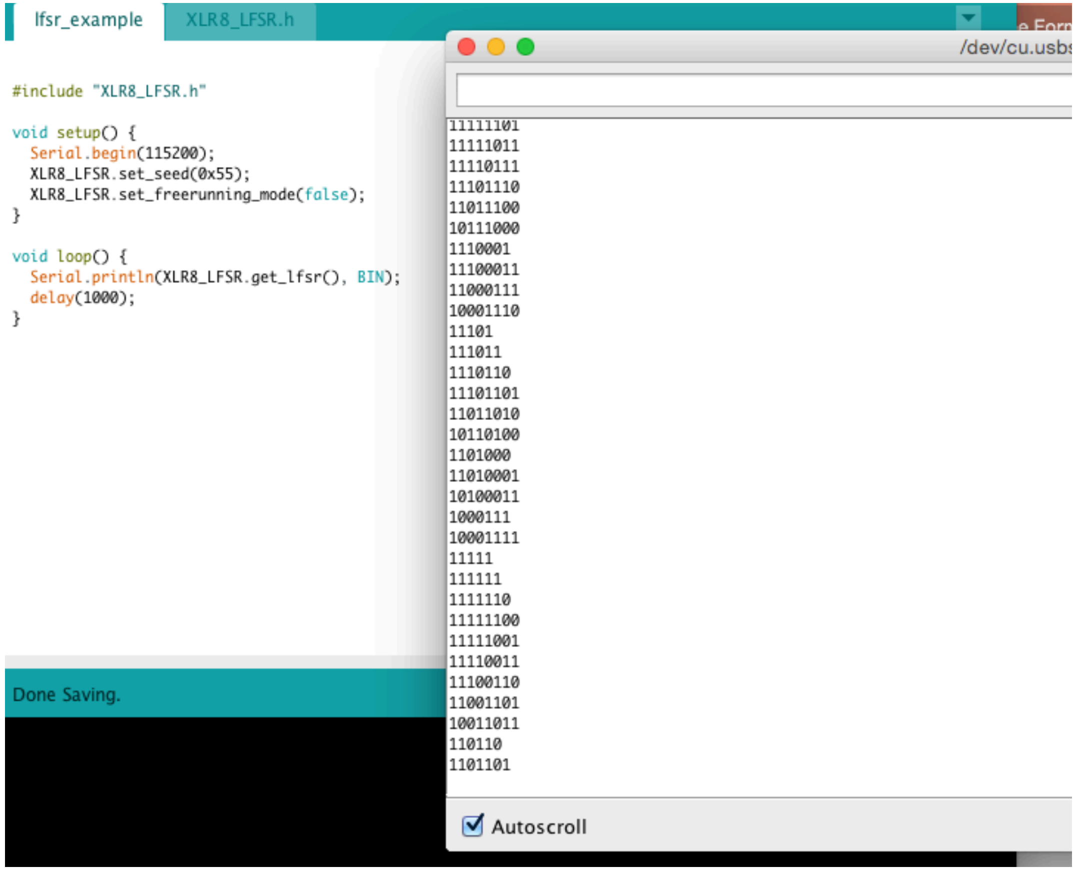
- XLR8_LFSR.h
- Defines the same register addresses as in the RTL
- Sets and reads the LFSR registers



```
#ifndef _XLR8_LFSR_H_INCLUDED
#define _XLR8_LFSR_H_INCLUDED

#include <Arduino.h>

#define XLR8_LFSR_CTRL _SFR_MEM8(0xE0)
#define XLR8_LFSR_SEED _SFR_MEM8(0xE1)
#define XLR8_LFSR_DATA _SFR_MEM8(0xE2)

class XLR8_LFSRClass {
public:
  XLR8_LFSRClass() {}
  ~XLR8_LFSRClass() {}
  void set_seed(uint8_t seed) {
    XLR8_LFSR_SEED = seed;
  }
  uint8_t get_lfsr() {
    return XLR8_LFSR_DATA;
  }
  void set_freerunning_mode(boolean freerunning) {
    XLR8_LFSR_CTRL = freerunning;
  }
private:
};

extern XLR8_LFSRClass XLR8_LFSR;

#endif
```

Arduino LFSR Example

- Include the XLR8_LFSR.h
- Set the seed, enter a loop to print the result of the LFSR to serial output
- Compile and run on the board



```
#include "XLR8_LFSR.h"

void setup() {
  Serial.begin(115200);
  XLR8_LFSR.set_seed(0x55);
  XLR8_LFSR.set_freerunning_mode(false);
}

void loop() {
  Serial.println(XLR8_LFSR.get_lfsr(), BIN);
  delay(1000);
}
```

```
11111101
11111011
11110111
11101110
11011100
10111000
1110001
11100011
11000111
10001110
11101
111011
1110110
11101101
11011010
10110100
1101000
11010001
10100011
1000111
10001111
11111
111111
1111110
11111100
11111001
11110011
11100110
11001101
10011011
110110
1101101
```

Thank You!

Questions?