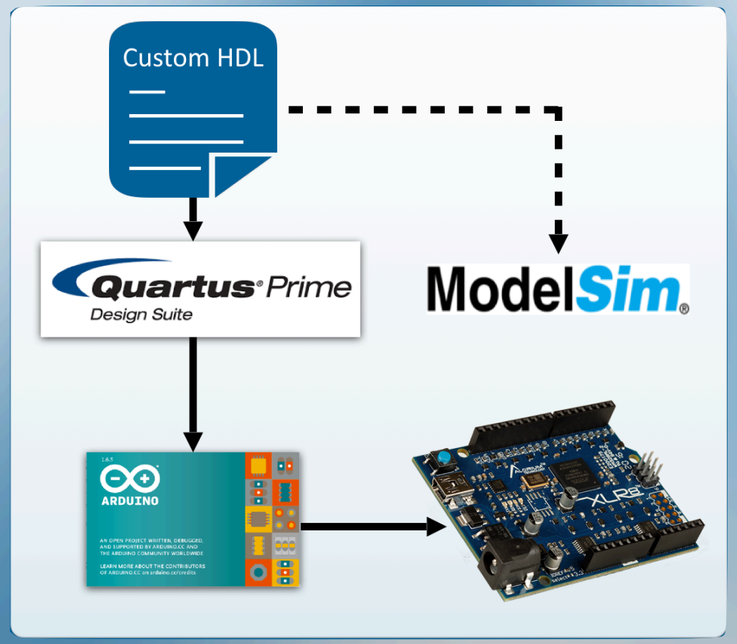
XLR8™ Webinars



OpenXLR8:
How to Load Custom FPGA Blocks

Webinar Breakdown:

- Introduction to pseudorandom number generator (LFSR) code
- Review of Verilog wrapper interface to microcontroller
- Simulation with Mentor Graphics® ModelSim®
- Synthesis using Intel® Quartus® Prime Lite
- Upload to FPGA via the Arduino IDE
- Overview software library
- Run simple sketch to demonstrate new FPGA hardware

Webinar Replay
from
January 12, 2017

Presenters

Jason Pecor

Harlie Juedes

Bryan Craker

ALORIUM TECHNOLOGY

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**

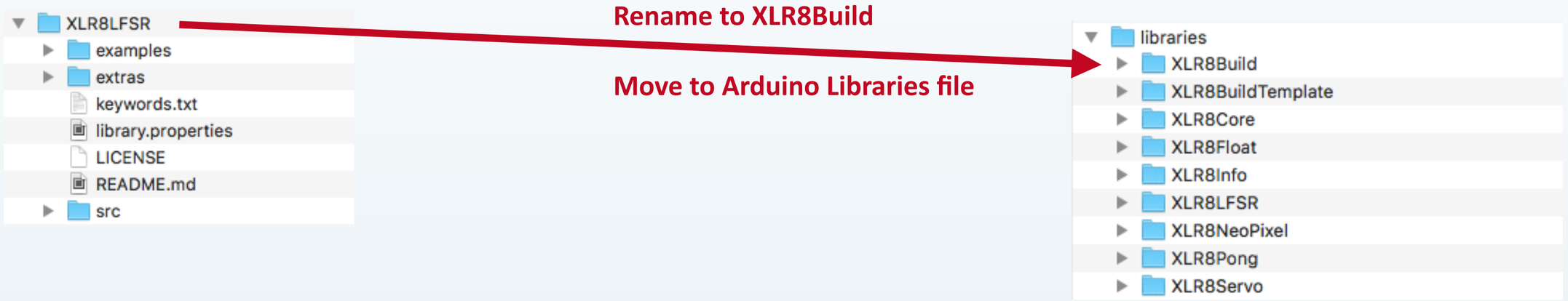
Follow the instructions here:
http://www.aloriumtech.com/openxlr8/

ALORIUM TECHNOLOGY

LFSR and Board Library URLs

- **LFSR Code Package:**

  https://github.com/AloriumTechnology/XLR8LFSR



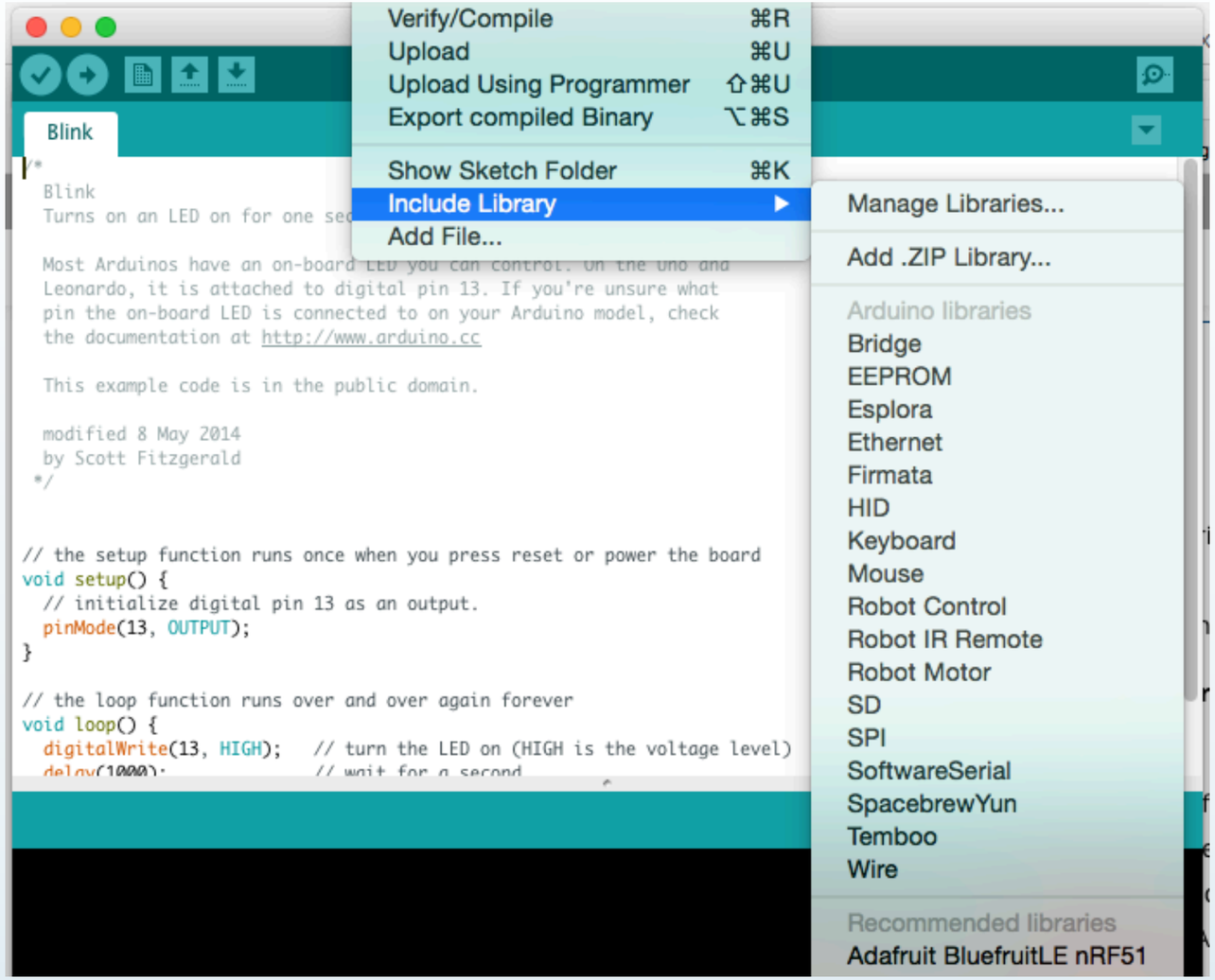
- **Arduino Board Library URL:**

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

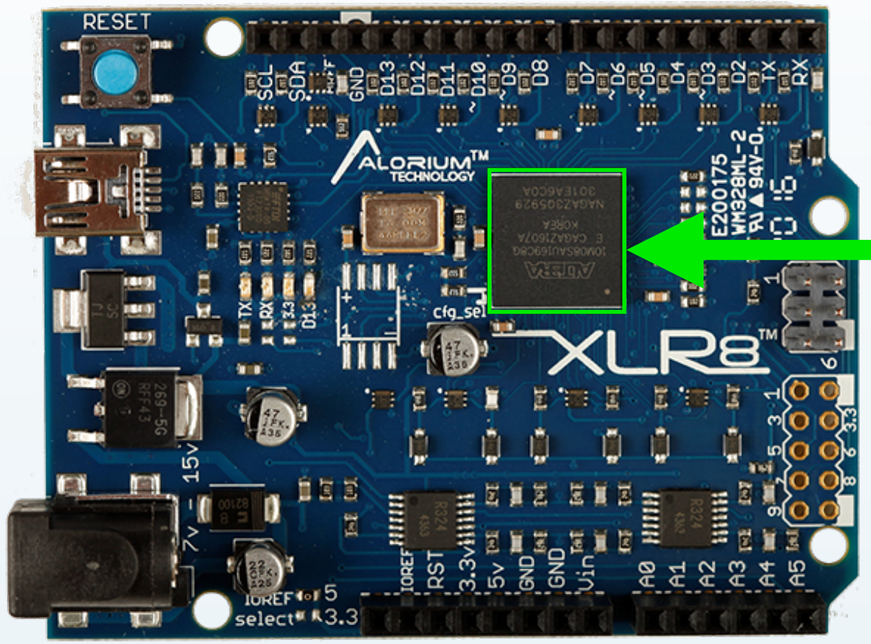
ALORIUM TECHNOLOGY

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



What is XLR8?



Application Accelerator & Development Board

Designed for Arduino Developer Community

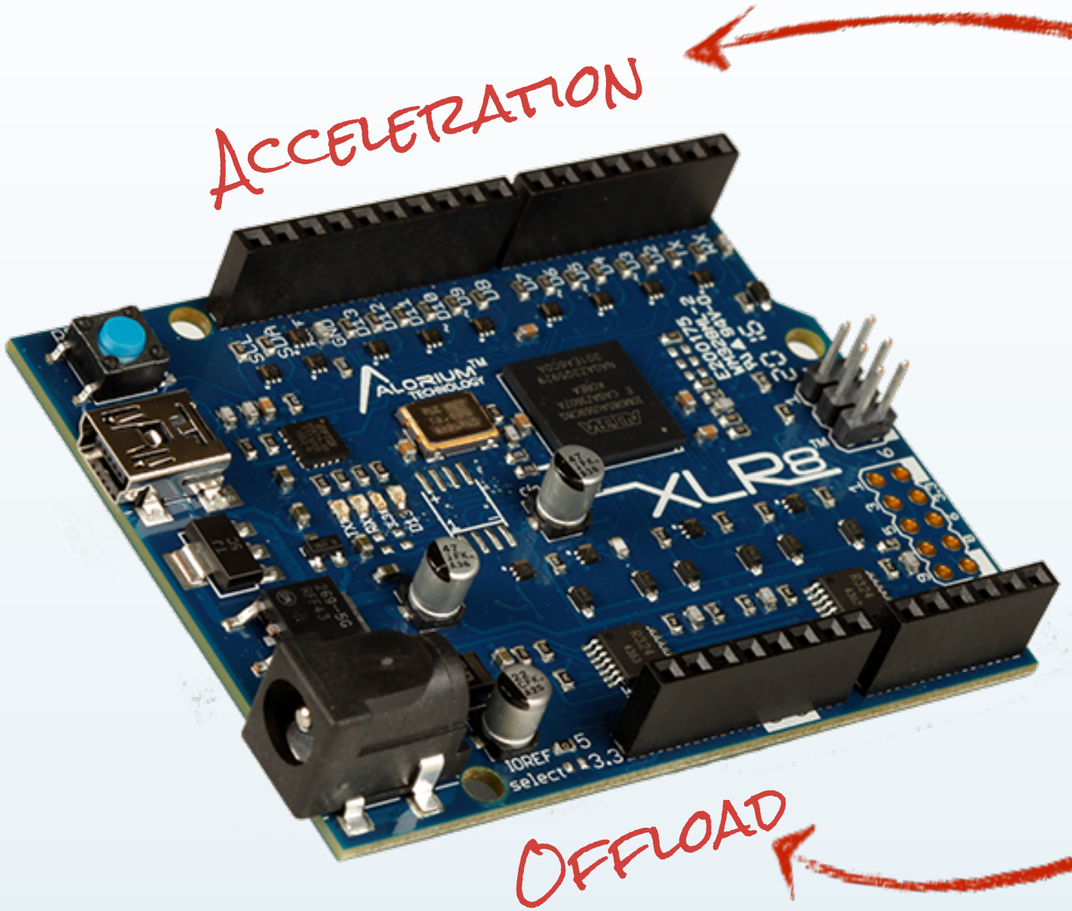
Based on Intel® MAX® 10 FPGA

Programmable with Arduino IDE

Field-Programmable Gate Array

ALORIUM TECHNOLOGY

Why use FPGA?

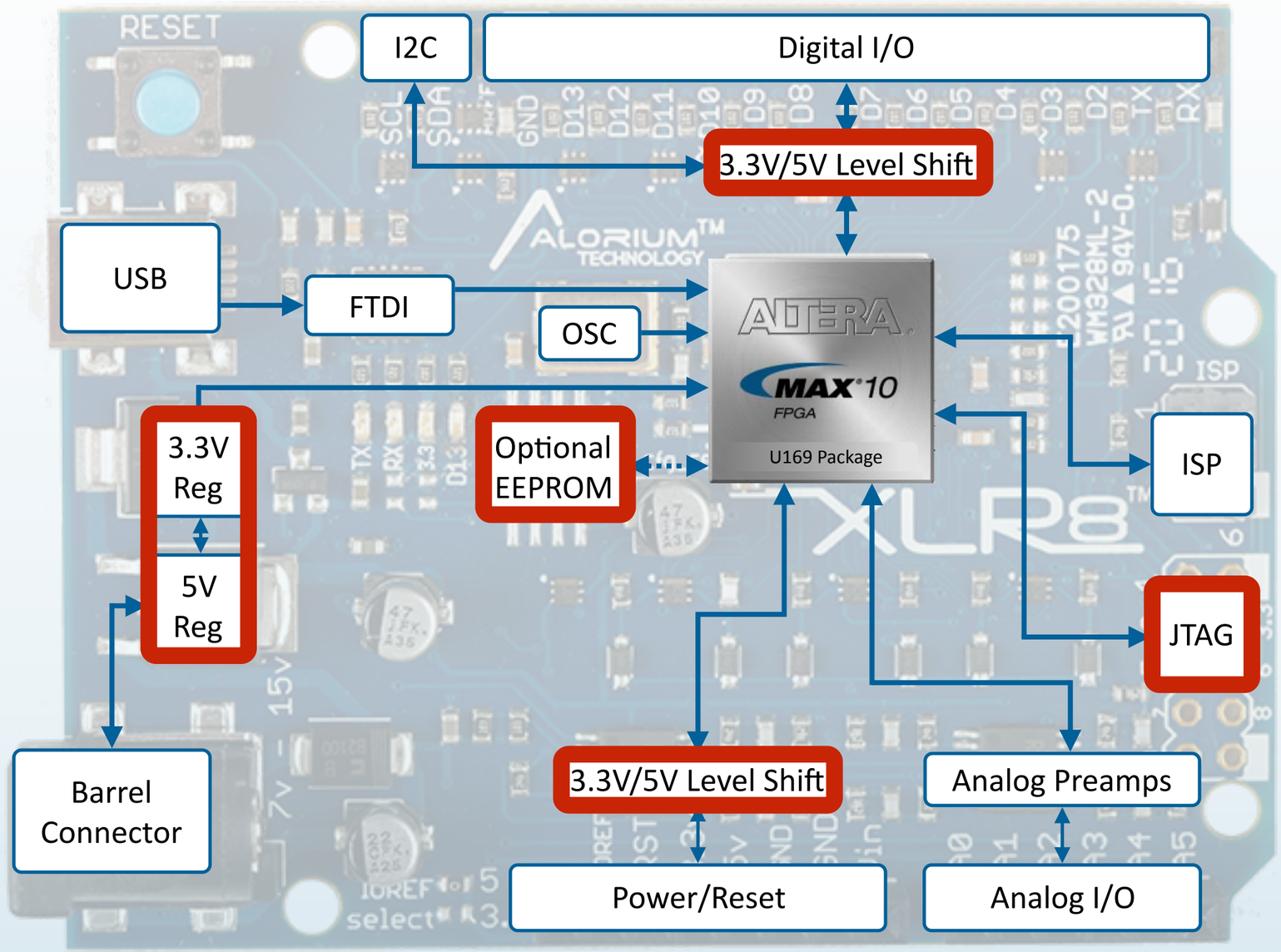


FASTER

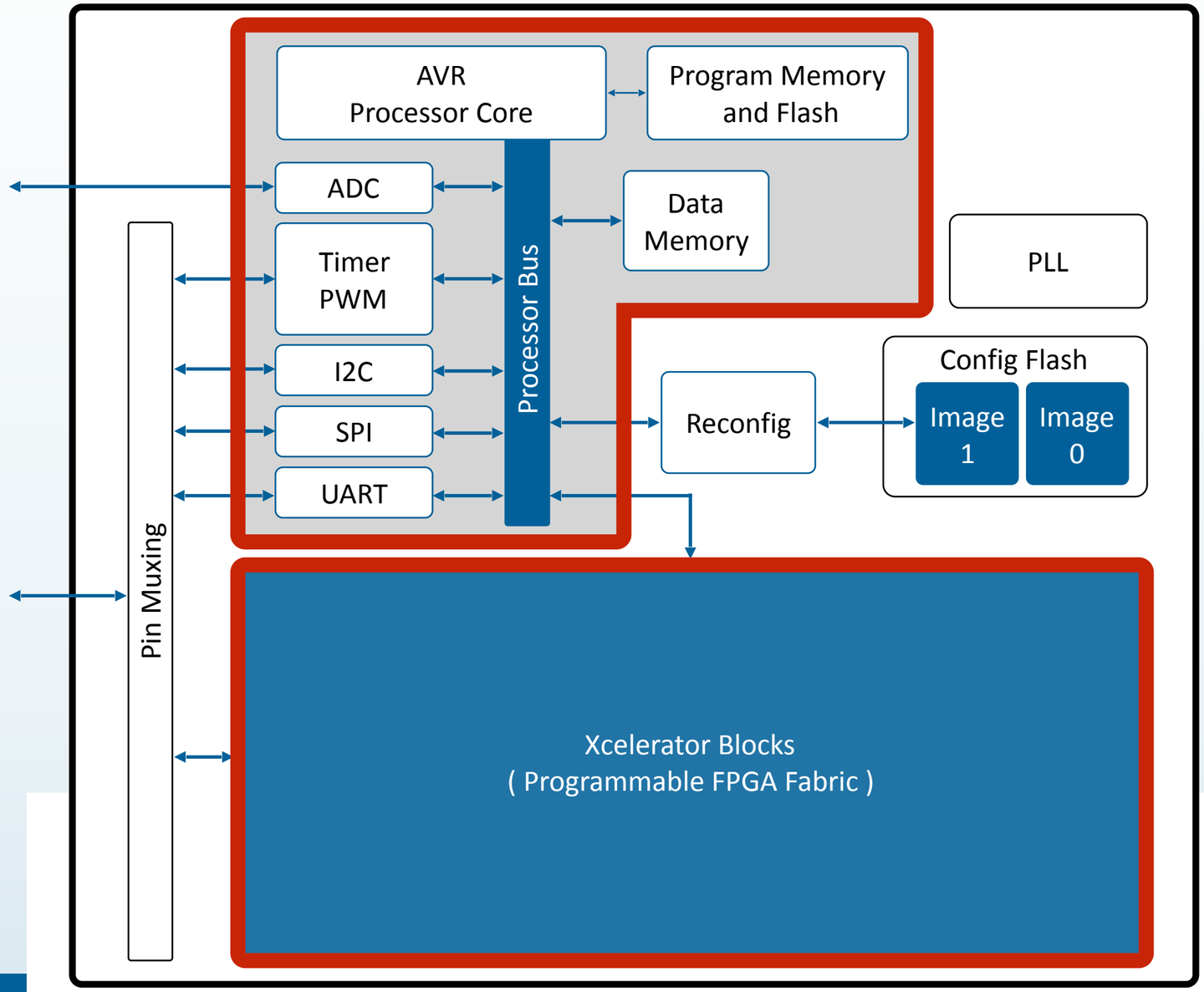
HIGHER-PERFORMANCE

ALORIUM TECHNOLOGY

Board Level Block Diagram



FPGA Block Diagram



ALORIUM TECHNOLOGY

Xcelerator Blocks

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

Custom hardware implemented on the same chip

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

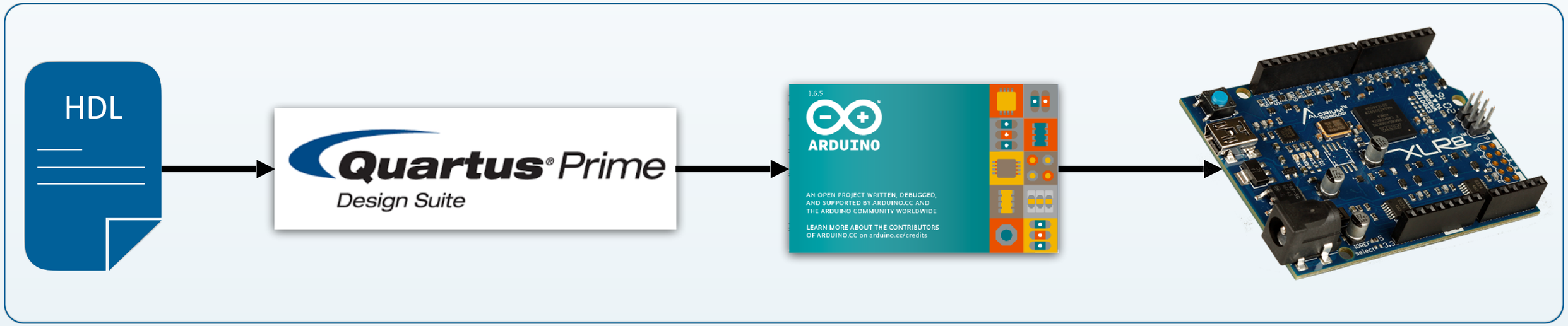
XB Roadmap

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

ALORIUM TECHNOLOGY

OpenXLR8

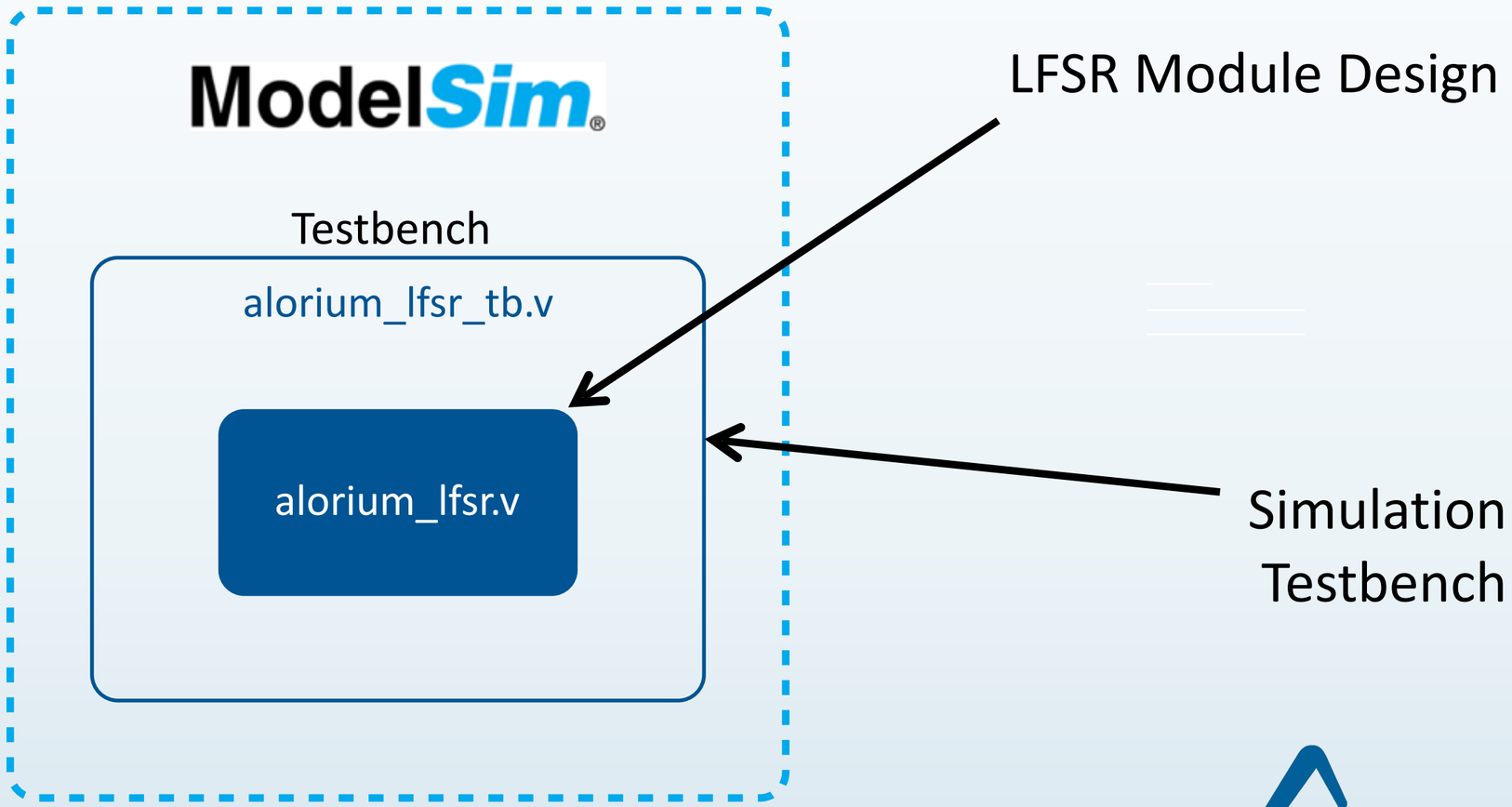
Methodology that allows XLR8 users to develop their own Xcelerator Blocks and upload them to the FPGA.



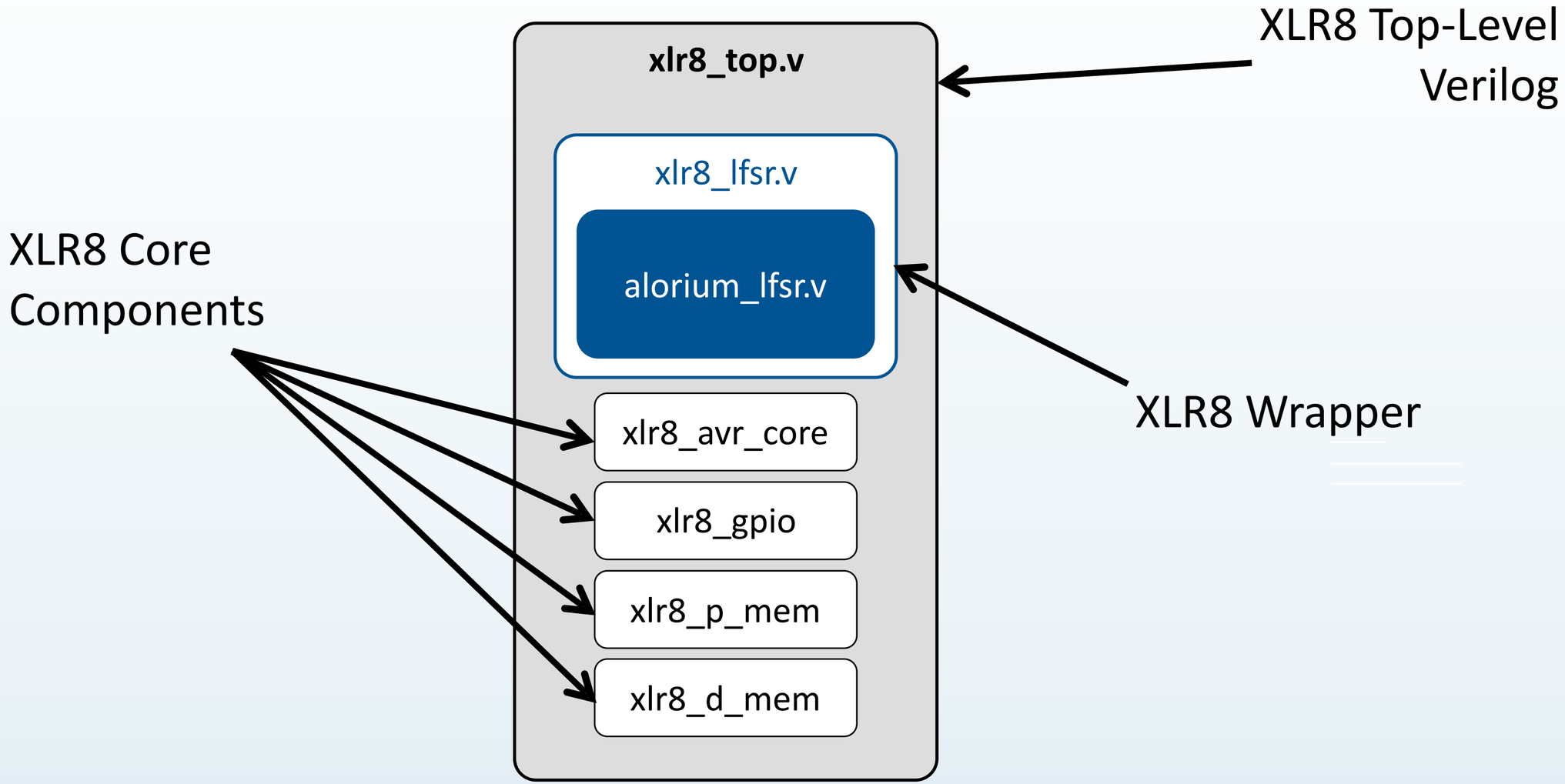
ALORIUM TECHNOLOGY

Module-Level Design and Simulation

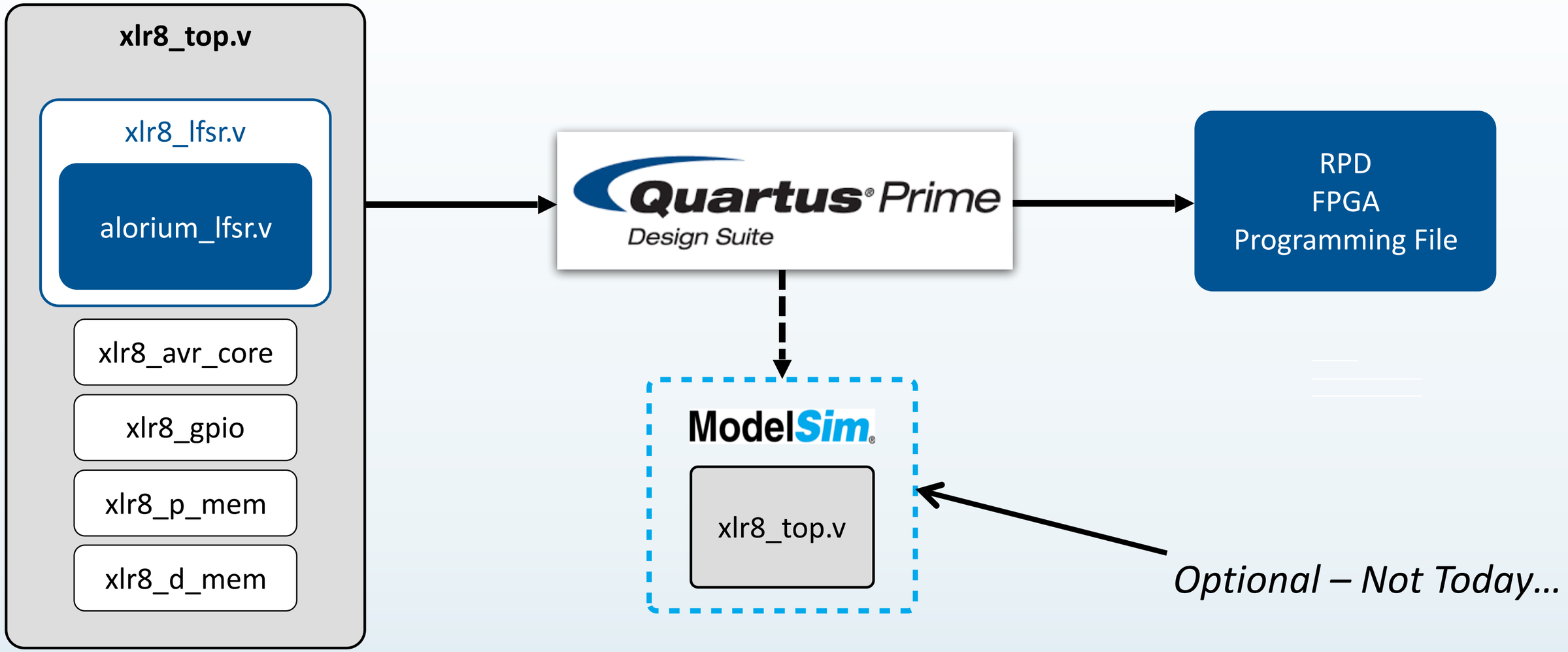
- **Pseudorandom Number Generator**
  - Using a Linear Feedback Shift Register (LFSR)
  - 8-bit
  - 4-tap



Integration into XLR8

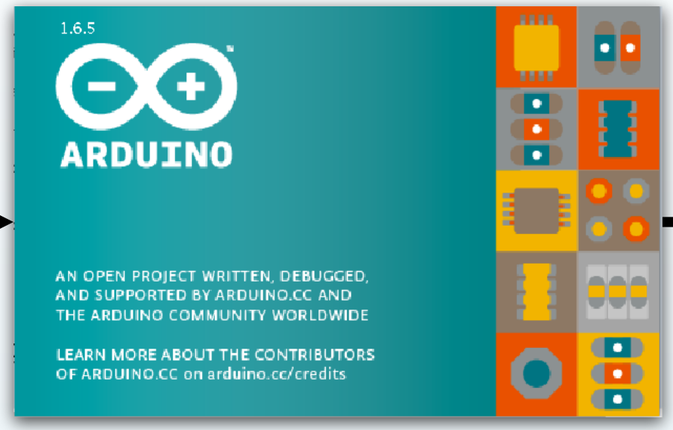


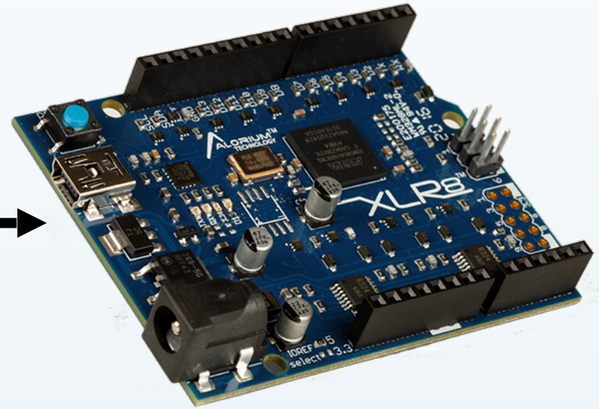
Synthesis


xlr8_top.v
xlr8_lfsr.v
alorium_lfsr.v
xlr8_avr_core
xlr8_gpio
xlr8_p_mem
xlr8_d_mem
Quartus Prime
Design Suite
RPD
FPGA
Programming File
ModelSim
xlr8_top.v
Optional – Not Today…

Upload to FPGA

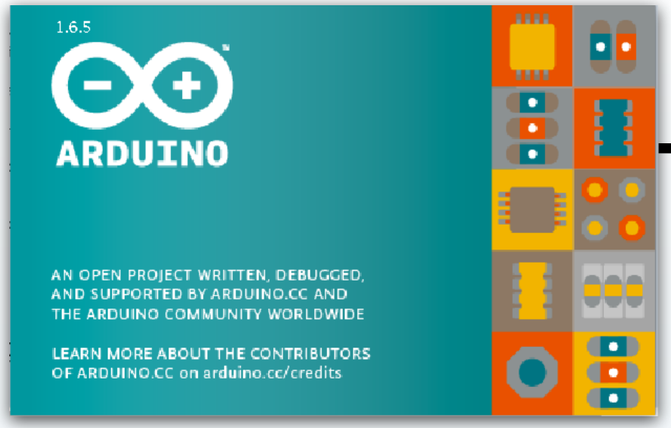
XLR8
RPD

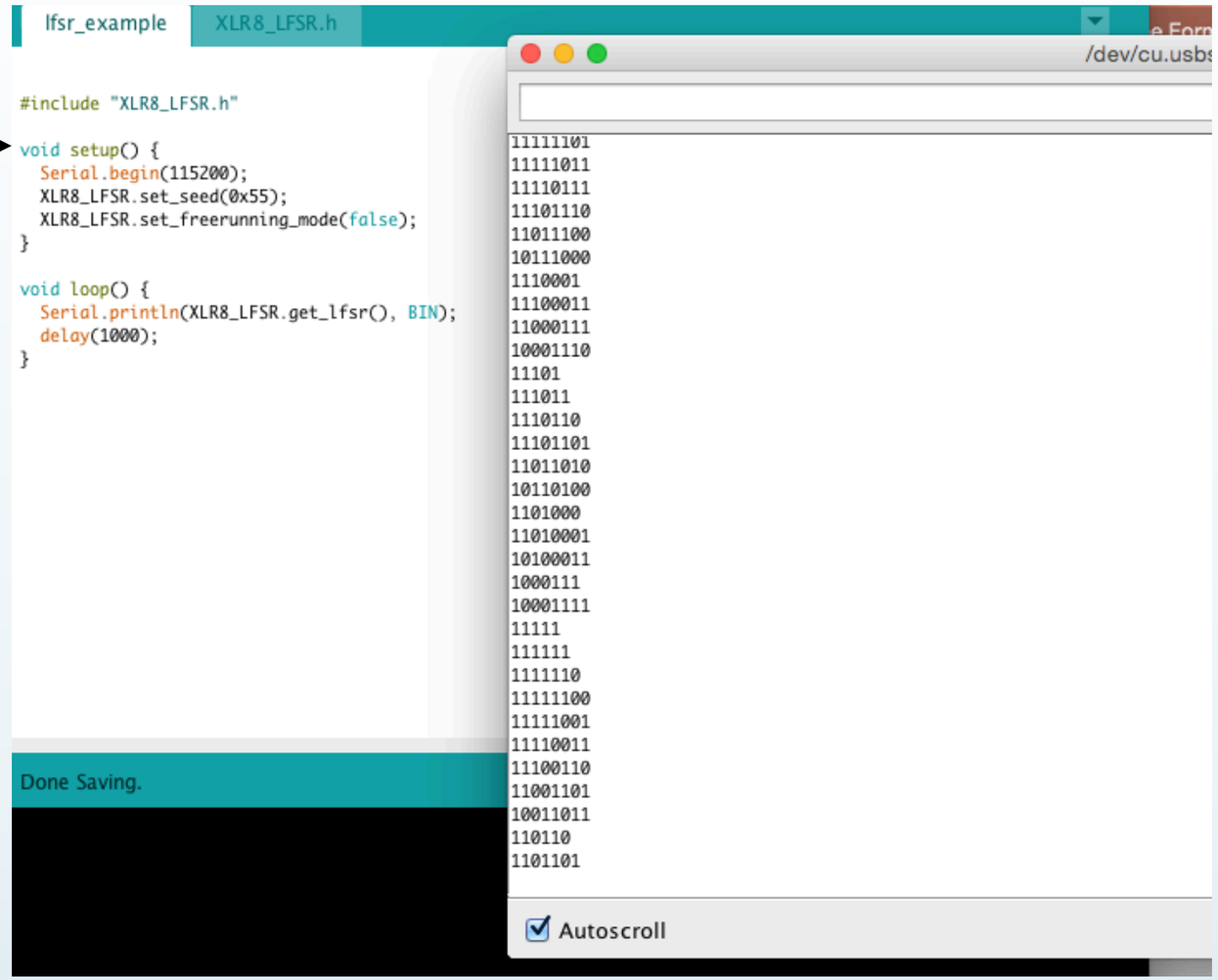




ALORIUM TECHNOLOGY

Run Sketch



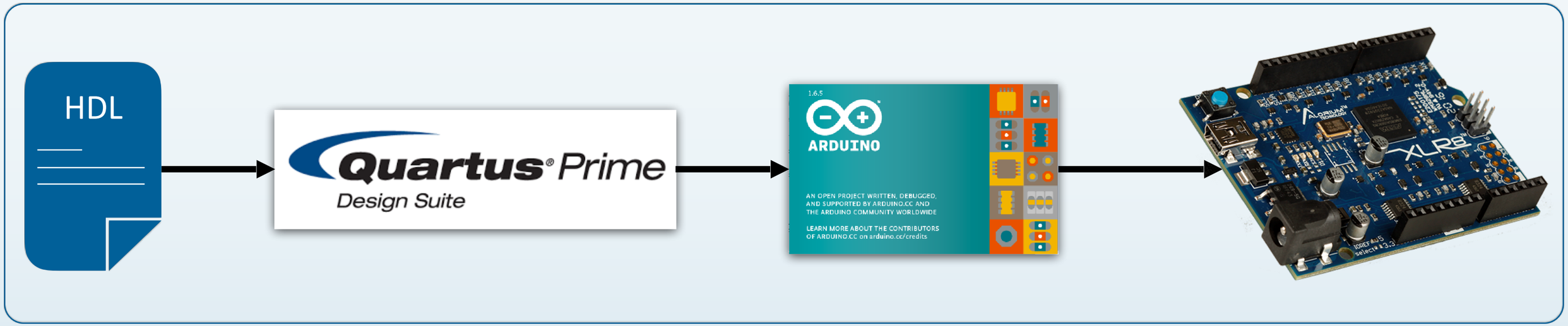


ALORIUM TECHNOLOGY

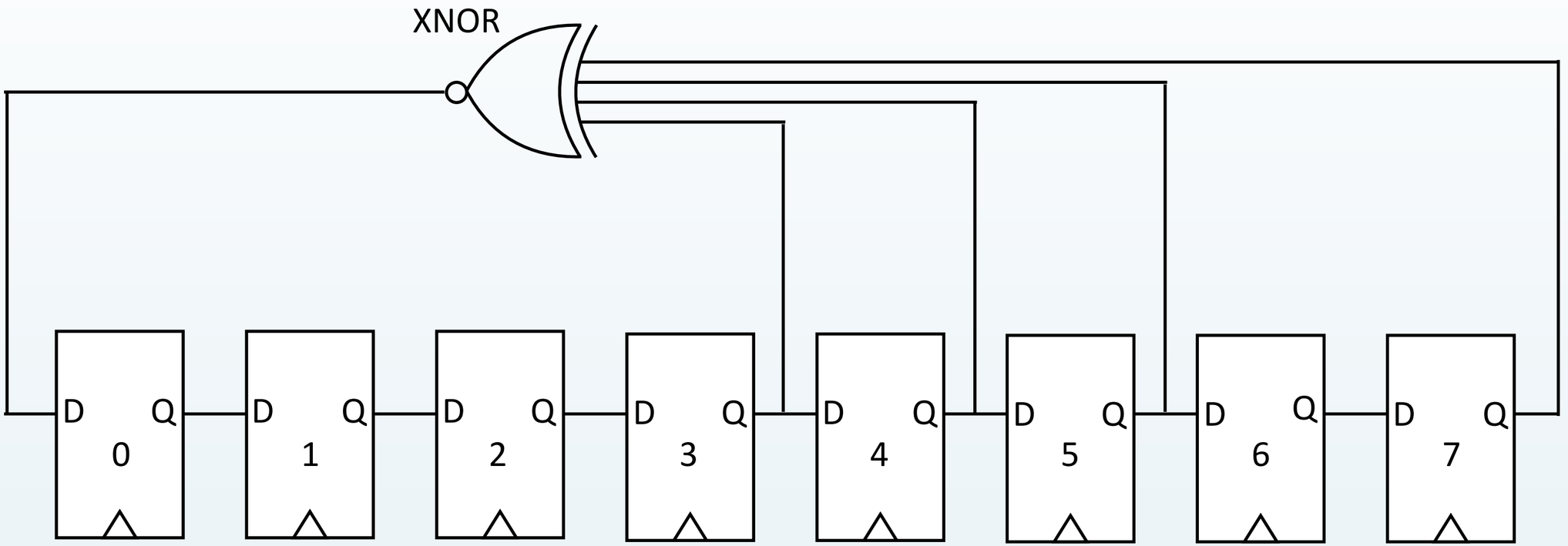
Let’s Dive In!

ALORIUM TECHNOLOGY

Building an LFSR on an FPGA



Linear Feedback Shift Register *(LFSR)*




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

Software Function vs Generated Assembly Code

```
uint8_t __attribute__ ((noinline)) advance_lfsr (uint8_t seed) {
  uint8_t result = 0;
  uint8_t feedback = 0;
  feedback = ~( ( ((seed & 0x80) >> 7) ^
                  ((seed & 0x20) >> 5) ^
                  ((seed & 0x10) >> 4) ^
                  ((seed & 0x08) >> 3) ) | 0xFE );
  result = seed << 1;
  result |= feedback;
  return result;
}
```

```
000002f6 <_Z12advance_lfsrh>:
 2f6:   38 2f           mov     r19, r24
 2f8:   33 0f           add     r19, r19
 2fa:   85 fb           bst     r24, 5
 2fc:   22 27           eor     r18, r18
 2fe:   20 f9           bld     r18, 0
 300:   84 fb           bst     r24, 4
 302:   99 27           eor     r25, r25
 304:   90 f9           bld     r25, 0
 306:   29 27           eor     r18, r25
 308:   98 2f           mov     r25, r24
 30a:   99 1f           adc     r25, r25
 30c:   99 27           eor     r25, r25
 30e:   99 1f           adc     r25, r25
 310:   29 27           eor     r18, r25
 312:   83 fb           bst     r24, 3
 314:   99 27           eor     r25, r25
 316:   90 f9           bld     r25, 0
 318:   82 2f           mov     r24, r18
 31a:   89 27           eor     r24, r25
 31c:   8e 6f           ori     r24, 0xFE       ; 254
 31e:   80 95           com     r24
 320:   83 2b           or      r24, r19
 322:   08 95           ret
```

ALORIUM TECHNOLOGY

RTL for the LFSR

- RTL = Register-Transfer Level
  - HDL code
  - Verilog/SystemVerilog
  - VHDL
- 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

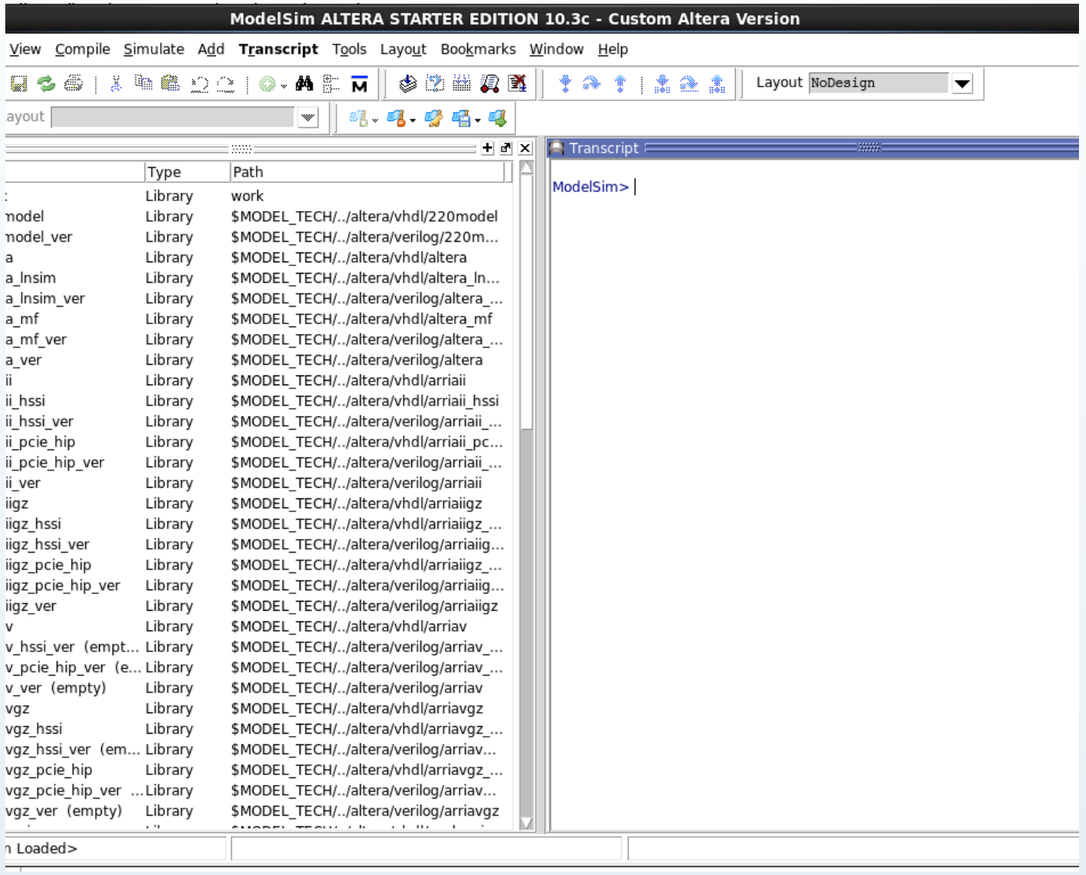
- The testbench, alorium_lfsr_tb.v

```
include "alorium_lfsr.v"

odule 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
```

ALORIUM TECHNOLOGY

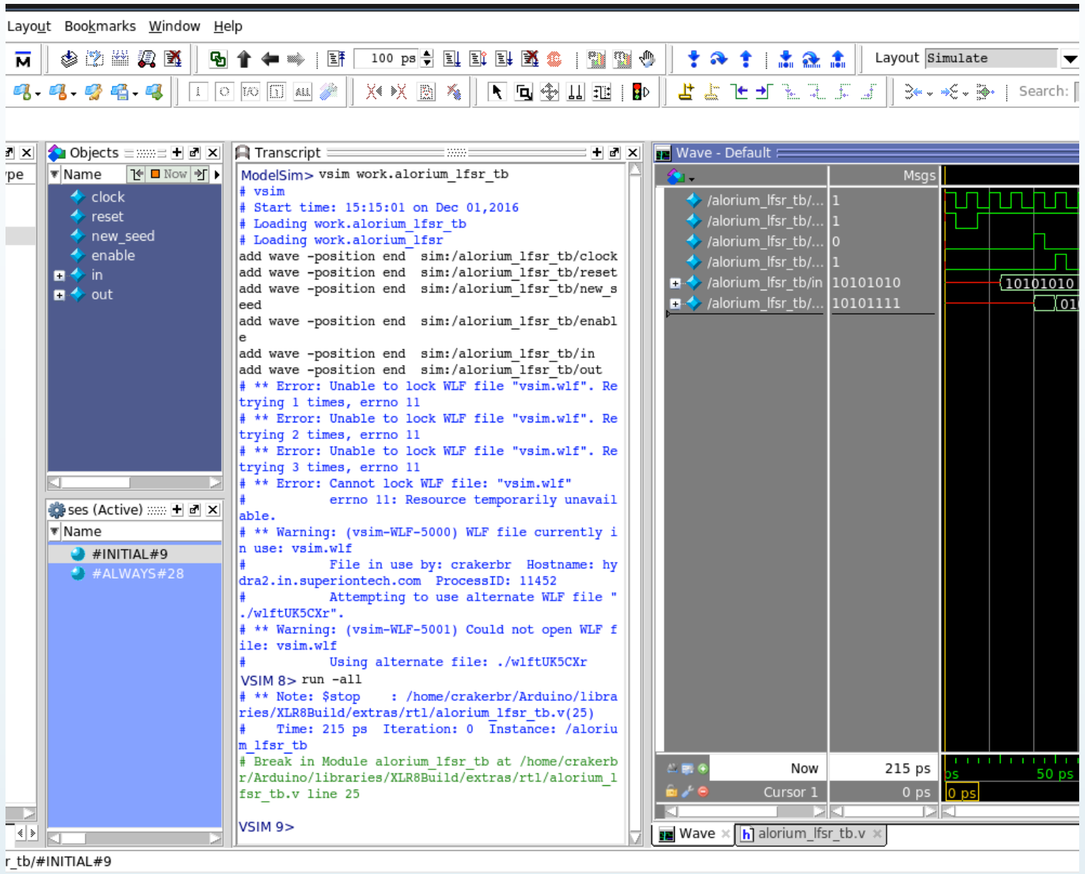
Simulating the Testbench

- 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



Simulating the Testbench Continued

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



XLR8 Module

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

```
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));
```

ALORIUM TECHNOLOGY

Register Definitions

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

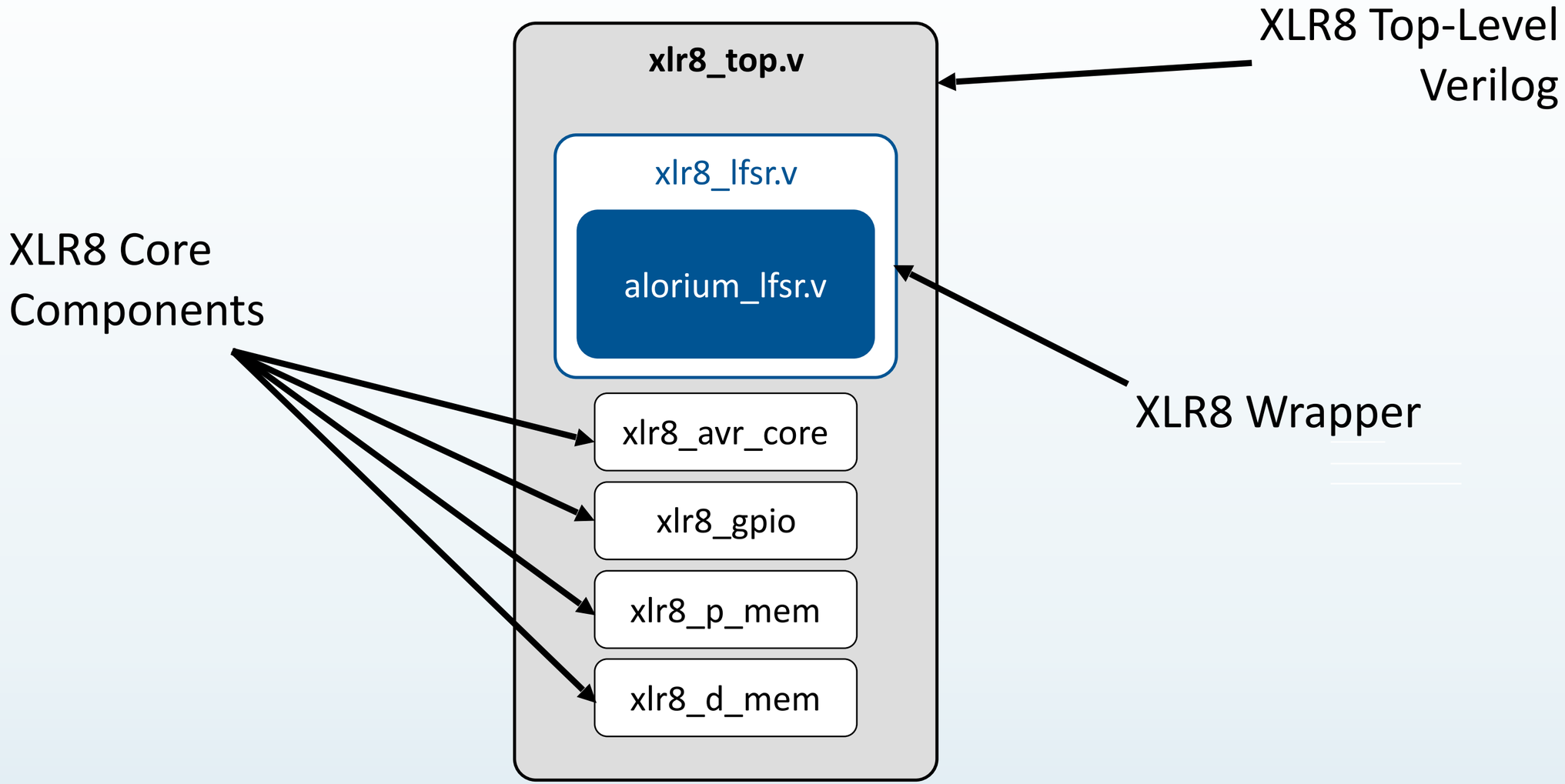
LFSR Data								Address 0xE2
Bit	7	6	5	4	3	2	1	0
Function	LFSR Result Data							
R/W	R	R	R	R	R	R	R	R
Initial	0	0	0	0	0	0	0	0

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;
```

Integration into XLR8



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”

```
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
```

ALORIUM TECHNOLOGY

Modify the Project QSF File

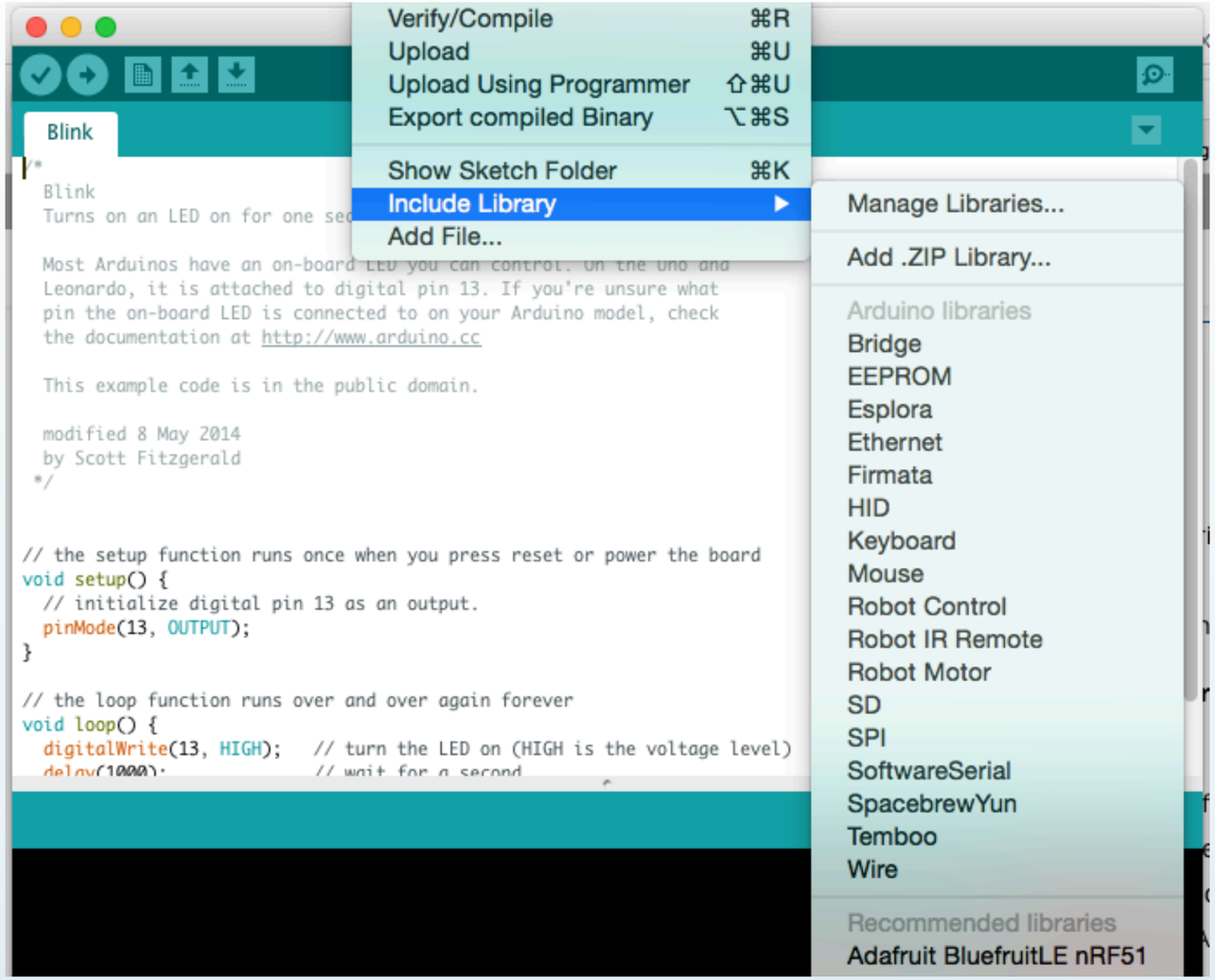
- xlr8_top.qsf under the “quartus” directory
- Add in our module files and the register address file

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

:ings for XLR8 project
.aloriumtech.com/xlr8
:hub.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

:l, 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
```

ALORIUM TECHNOLOGY

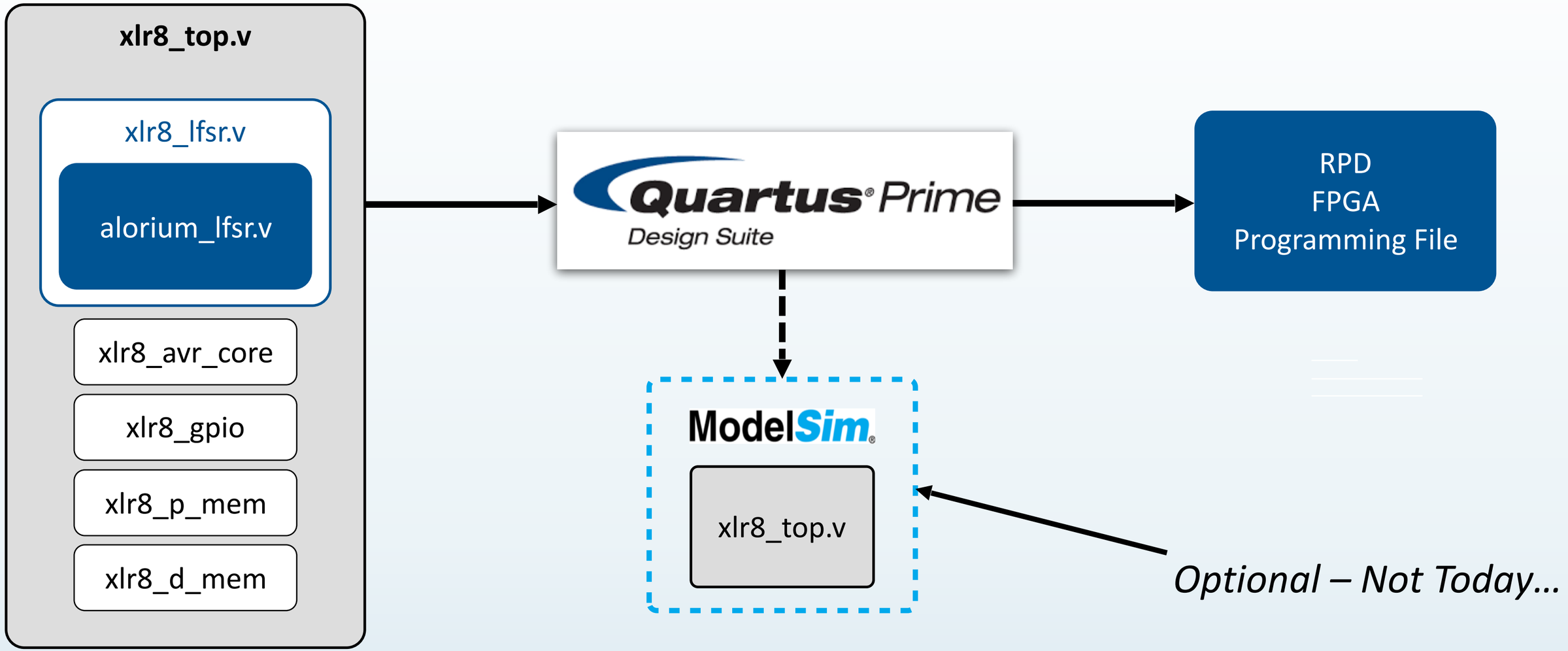
Arduino IDE Setup

- 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



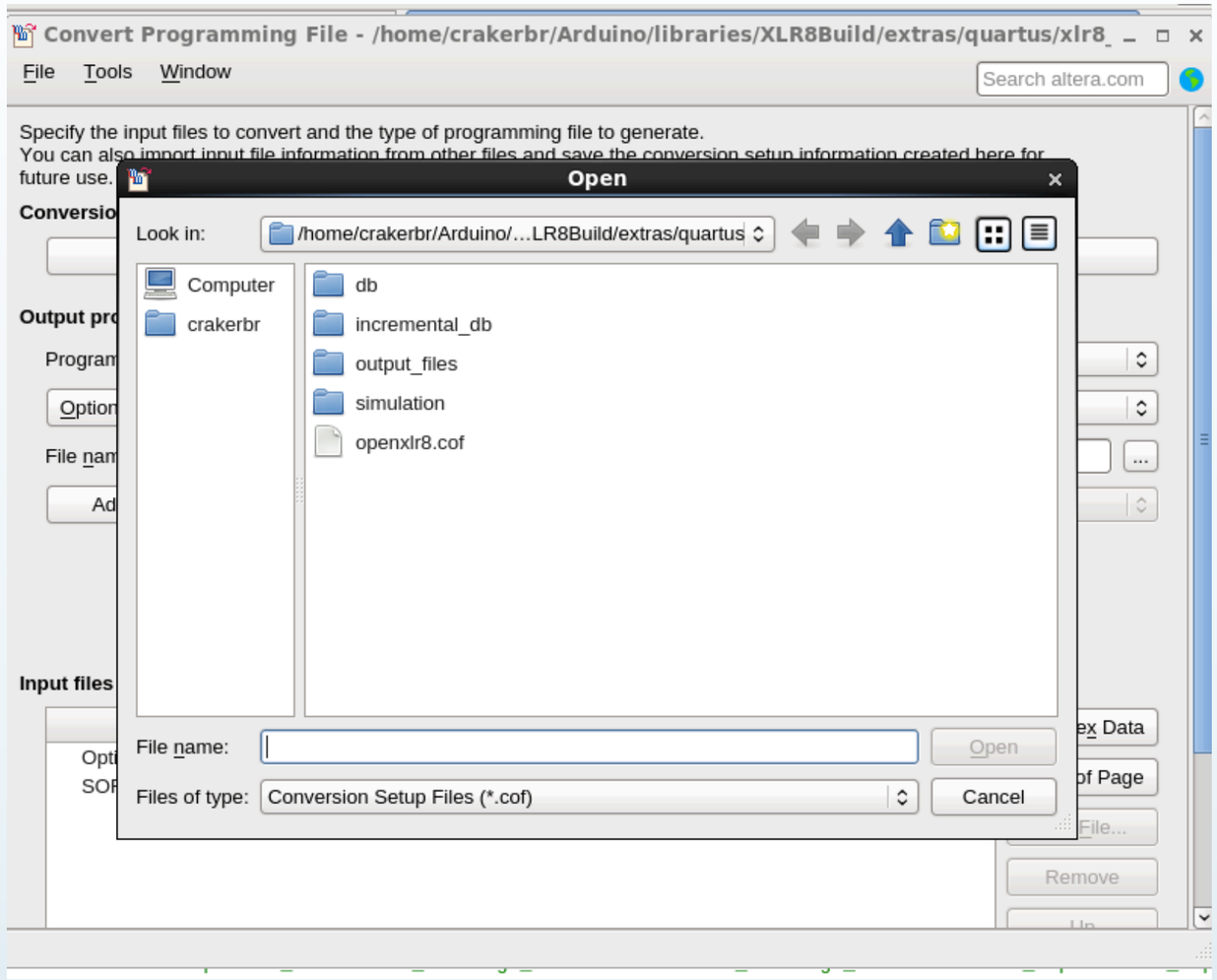
ALORIUM TECHNOLOGY

Synthesis


xlr8_top.v
xlr8_lfsr.v
alorium_lfsr.v
xlr8_avr_core
xlr8_gpio
xlr8_p_mem
xlr8_d_mem
Quartus Prime
Design Suite
RPD
FPGA
Programming File
ModelSim
xlr8_top.v
Optional – Not Today…

Compile the Project in Quartus

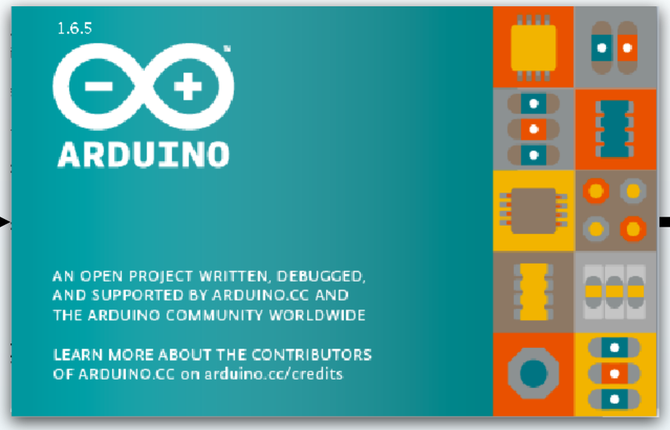
- Open Quartus and open our project QPF file with File -> Open Project…
- Begin the compile with Processing -> Start Compilation
- After compilation is completed, File -> Convert Programming Files…
- Open Conversion Setup Data, open “openxlr8.cof,” and Generate

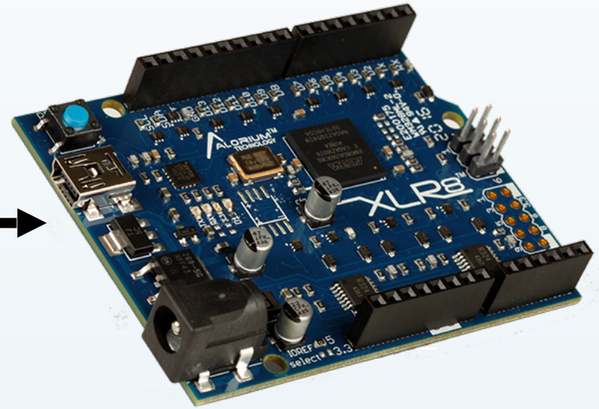


ALORIUM TECHNOLOGY

Upload to FPGA

XLR8
RPD

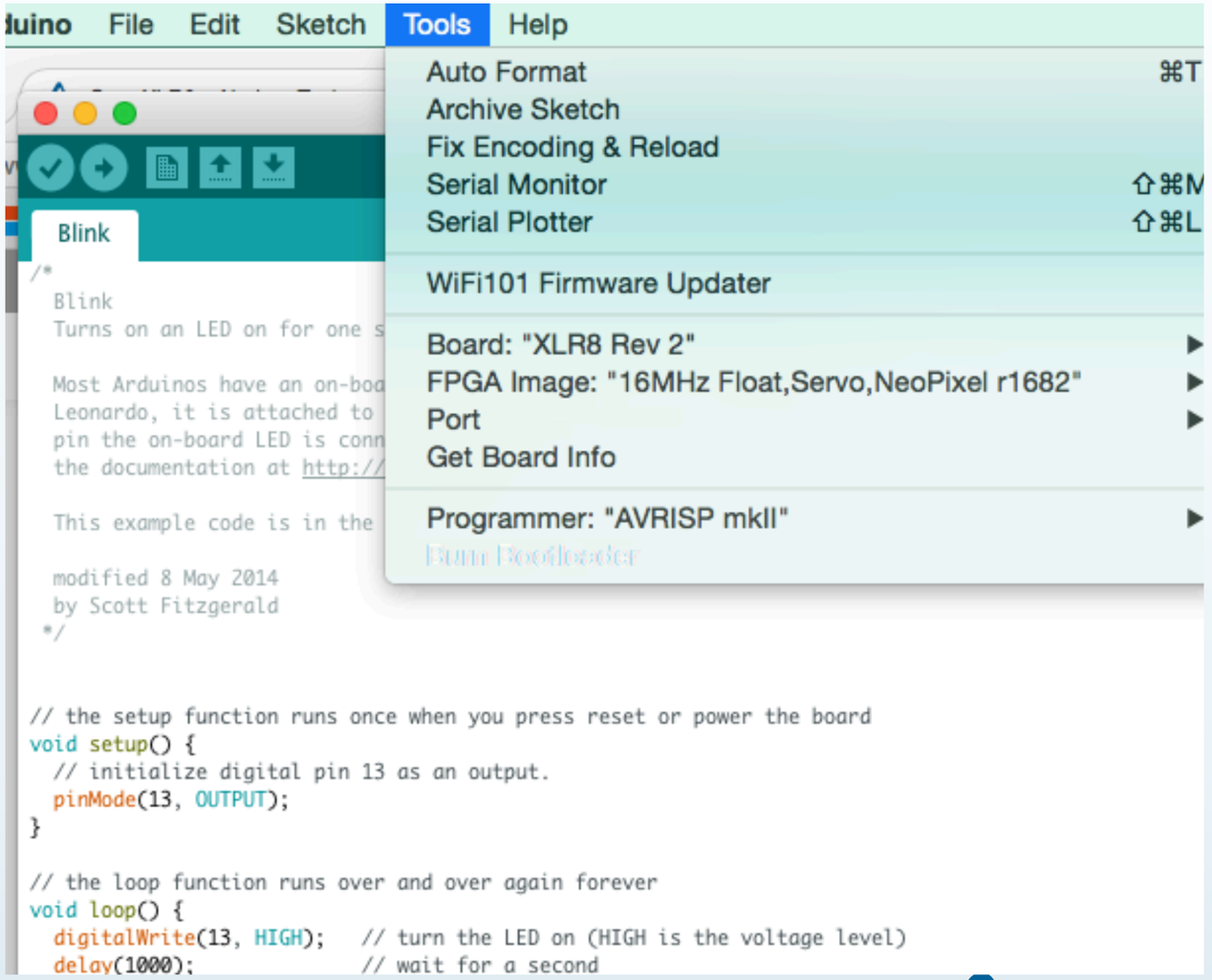




ALORIUM TECHNOLOGY

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



ALORIUM TECHNOLOGY

Arduino Library for the LFSR

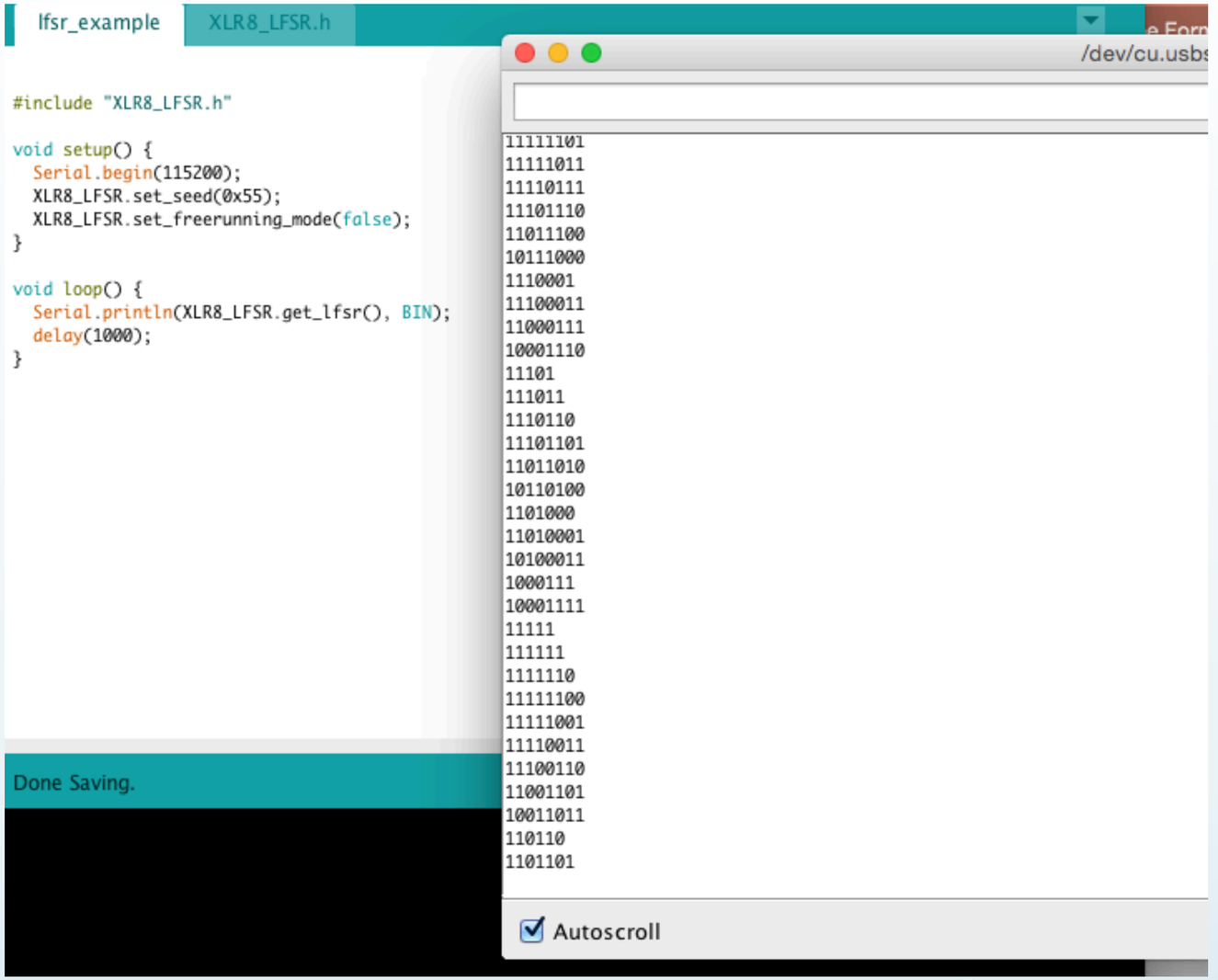
- XLR8_LFSR.h
- Defines the same register addresses as in the RTL
- Sets and reads the LFSR registers

```
#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
```

ALORIUM TECHNOLOGY

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



Assembly Code: Software vs FPGA

```
000002f6 <_Z12advance_lfsrh>:
 2f6:   38 2f           mov     r19, r24
 2f8:   33 0f           add     r19, r19
 2fa:   85 fb           bst     r24, 5
 2fc:   22 27           eor     r18, r18
 2fe:   20 f9           bld     r18, 0
 300:   84 fb           bst     r24, 4
 302:   99 27           eor     r25, r25
 304:   90 f9           bld     r25, 0
 306:   29 27           eor     r18, r25
 308:   98 2f           mov     r25, r24
 30a:   99 1f           adc     r25, r25
 30c:   99 27           eor     r25, r25
 30e:   99 1f           adc     r25, r25
 310:   29 27           eor     r18, r25
 312:   83 fb           bst     r24, 3
 314:   99 27           eor     r25, r25
 316:   90 f9           bld     r25, 0
 318:   82 2f           mov     r24, r18
 31a:   89 27           eor     r24, r25
 31c:   8e 6f           ori     r24, 0xFE       ; 254
 31e:   80 95           com     r24
 320:   83 2b           or      r24, r19
 322:   08 95           ret
```

```
000000be <_ZN14XLR8_LFSRClass8set_seedEh.isra.0.constprop.12>:
  be:   85 e5           ldi     r24, 0x55       ; 85
  c0:   80 93 e1 00     sts     0x00E1, r24
  c4:   08 95           ret
```

```
000002fe <_ZN14XLR8_LFSRClass8get_lfsrEv.isra.1>:
 2fe:   80 91 e2 00     lds     r24, 0x00E2
 302:   08 95           ret
```

ALORIUM TECHNOLOGY

Q&A