

A build system for DAMC-FMC25 Board Support Package

Jan Marjanovic

CAEN ELS s.r.l.

December 8, 2016
5th MicroTCA Workshop for
Industry and Research

CAEN ELS s.r.l.

- Spin-off company of CAEN SpA, founded in 2009 in Slovenia, transferred to Italy in 2016
- Oriented and dedicated to particle accelerator facilities
- Know-how and hands-on large installations and maintenance
- Customization and dedicated support
- Product families:
 - MicroTCA and FMC instrumentation
 - Power Supply Systems
 - Beamline Electronic Instrumentation
 - Precision Current Transducers

DAMC-FMC25 FMC carrier board

FMC carrier is a base for modular system



FMC 4-channel Picoammeter



4 channel picoammeter

Two input ranges: 1mA, 1uA

10 kHz input bandwidth

Customizable on request
(full-scale value, bandwidth)

FMC SFP Adapter



4 slot FMC carrier

Data rate up-to 10 Gbps

Optical or copper (RJ45) SFP modules

Developed by DESY

FMC Motor Driver



2 channel stepper motor driver

1.8A coil current max

256 micro-steps for full step

Developed by DESY

FMC 16-channel ADC board



FMC116 from 4DSP

16-channels, 14-bit ADCs

125 MHz max sampling rate

FMC GPIO board



68-pin GPIO board
from Alpha Data

FMC ADC evaluation board



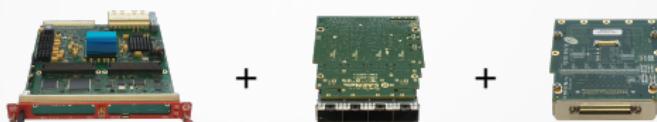
Single channel 500 MSPS ADC
12-bit resolution
from Analog Devices

Some application ideas

8-ch picoammeter =



Data aggregator =



Beam pos sys =

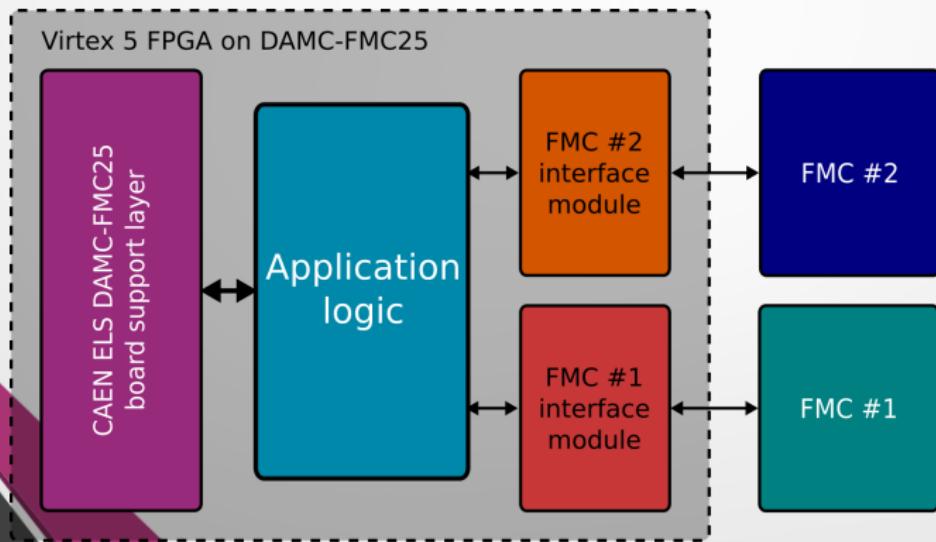


Machine prot sys =



DAMC-FMC25 Board Support Package

Board Support Package hides all the "boring" parts of the development → more time for application logic



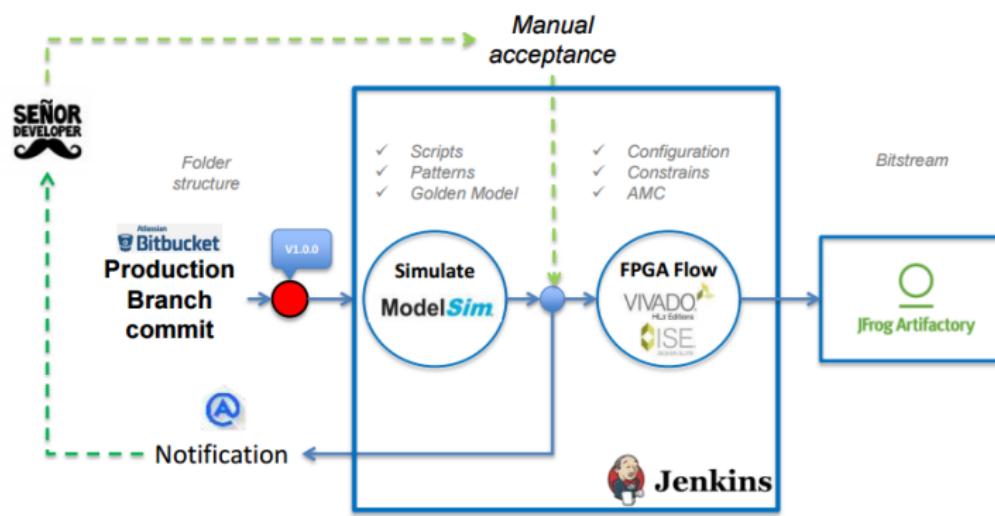
Build system

What composes a build system?

- ① Tcl script to recompile project from source code
- ② Python script to run all tests
- ③ Doxygen configuration file to generate low-level documentation from comments

Build procedure

From Maurizio Donna's (ESS) talk
(<https://indico.esss.lu.se/event/642/contribution/27>):



The Joel Test: 12 Steps to Better Code

(from <http://www.joelonsoftware.com/articles/fog0000000043.html>)

- ① Do you use source control?
- ② **Can you make a build in one step?**
- ③ Do you make daily builds?
- ④ Do you have a bug database?
- ⑤ Do you fix bugs before writing new code?
- ⑥ Do you have an up-to-date schedule?
- ⑦ Do you have a spec?
- ⑧ Do programmers have quiet working conditions?
- ⑨ Do you use the best tools money can buy?
- ⑩ Do you have testers?
- ⑪ Do new candidates write code during their interview?
- ⑫ Do you do hallway usability testing?

recreate_project.tcl

Steps to get from the source code to final binary:

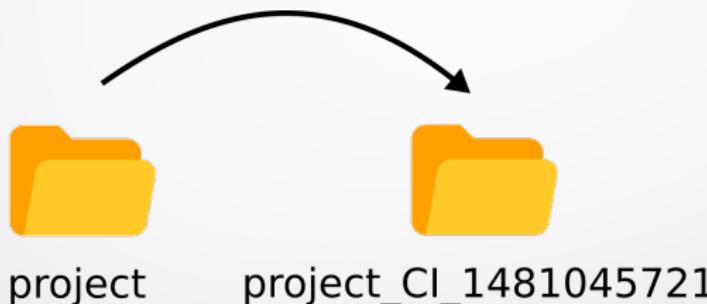
- ① Recreate IP cores outputs
- ② Embed the timestamp and Git commit SHA1 value in header file

```
localparam REG_TIMESTAMP = 32'h0000_0000;  
localparam REG_VERSION    = 32'h00_01_00_04;  
localparam REG_GIT_SHA1   = 32'h0000_0000;
```

- ③ Add all source files to the project
- ④ Set compilation options
- ⑤ Compile project

Local build

A script will git-clone a project locally and rebuild it in a separated folder.



Jenkins

A Continuous Integration server compiles a code from Git repository and stores the output product (binary).

All	+	Name ↓	Last Success	Last Failure	Last Duration	Fav
		amc-pico8-virtex5	2 mo 11 days - #6	N/A	21 min	
		ESS-BLEDP-interface	21 days - #5	22 days - #2	19 min	
		ESS-nBLM-500M	1 mo 27 days - #27	1 mo 28 days - #24	28 min	
		Fast-PS	15 days - #1	N/A	13 sec	
		test	2 mo 17 days - #3	N/A	2.4 sec	
		test_damc	2 mo 16 days - #1	N/A	3.9 sec	

Icon:

Legend: RSS for all RSS for failures RSS for just latest builds

run_all_tests.py

A Python script runs all the test in the project and checks for the successful completion.

```
TESTS = [
    '../hdl/AD9434_capture/test/AD9434_data_edge_detect/sim.do',
    '../hdl/AD9434_capture/test/AD9434_data_level_meas/sim.do',
    '../hdl/AD9434_clock_calibration/test/sim.do',
    '../hdl/AD9434_PN9_checker/test/sim.do',
    '../hdl/AD9434_SPI_interface/test/sim.do',
    '../hdl/axi_stream_monitor/test/sim.do',
    '../hdl/clk_freq_meas/test/xilinx_sim.do',
    '../hdl/edge_detection/test_kernel/sim.do',
    '../hdl/edge_detection/test_module/sim.do',
    '../hdl/median_filter/test_filter/sim.do',
    '../hdl/median_filter/test_kernel/sim.do',
    '../hdl/nblm_buffer_system/nblm_buffer_unit/test/sim.do',
    '../hdl/nblm_buffer_system/test/sim.do',
```

Testbenches

Functionality of BSP components is tested with 5 extensive unit tests (number of lines of code shown):

```
205 .../hdl/DDR2_AXI/test/DDR2_AXI_slave_tb.sv  
66 .../hdl/DDR2_AXI/test/ddr2_model.sv  
55 .../hdl/DDR2_AXI/test/sim.do  
326 total
```

```
404 .../hdl/pcie_axi_sys/test/pcie_axi/pcie_axi_tb.sv  
263 .../hdl/pcie_axi_sys/test/pcie_axi/pcie_axi_tb_tasks.h  
65 .../hdl/pcie_axi_sys/test/pcie_axi/sim.do  
732 total
```

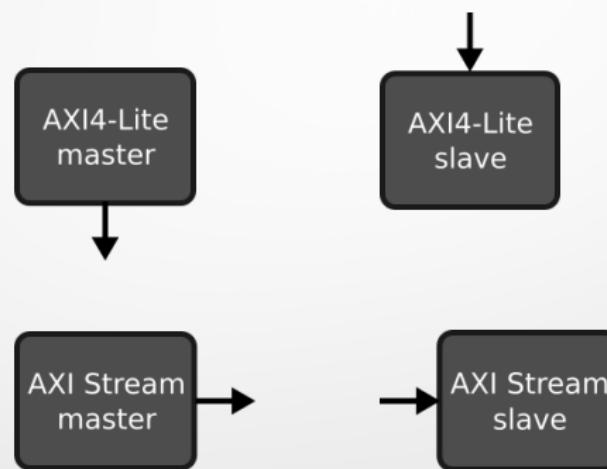
```
427 .../hdl/axi_interconnect/test/axi_interconnect_tb.sv  
53 .../hdl/axi_interconnect/test/sim.do  
480 total
```

```
282 .../hdl/axi_dma/test/axi_dma_tb.sv  
62 .../hdl/axi_dma/test/sim.do  
344 total
```

```
68 .../hdl/axis_prbs/test/sim.do
```

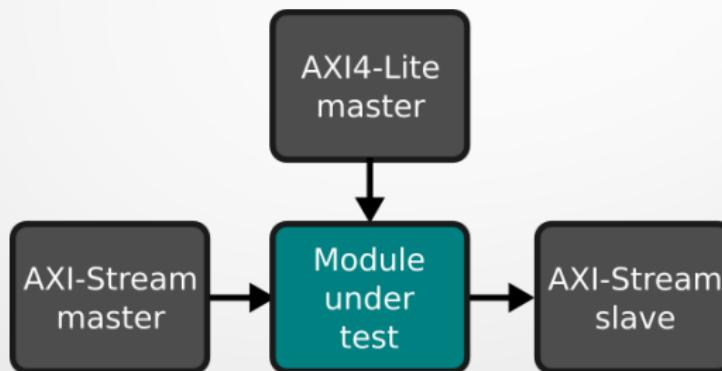
Bus Functional Models

SystemVerilog modules which can be used to verify the functionality of each individual module in the design

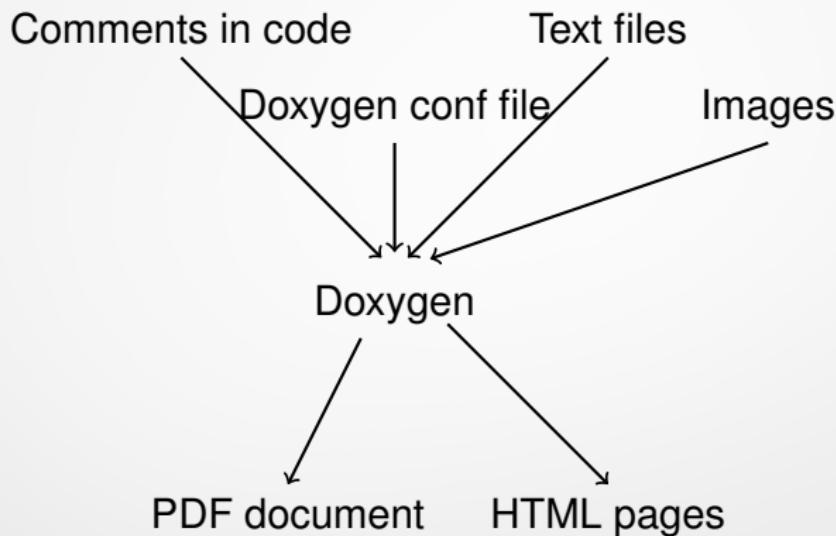


Use of BFM in testbench

Each individual module can be tested using BFM to configure the module, driver the inputs and collect the outputs.



Doxygen

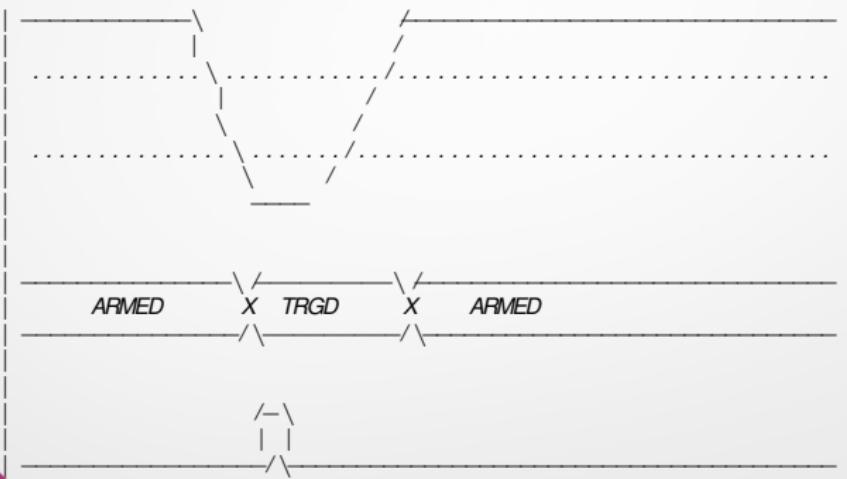


Module description

```
//% @brief Edge detection kernel
//%
//% This module provides an edge detection with hysteresis.
//%
//% 'comp_gt' signal can be used to select between triggering on rising edge or
//% negative edge of the input signal. The value of 'comp_gt' should only be
//% changed when the module is disabled. Otherwise a state machine can remain
//% in undesired state.
//%
//% In both trigger-edge modes 'abs(level_hi)' should always be greater than
//% 'abs(level_lo)'.
//%
//%
//% @project ESS-nBLM-500M
//% @author Jan Marjanovic
//% @company CAEN ELS s.r.l.
//% @date September 2016
```

Module description

```
//% ### Negative pulse detection (comp_gt == 0)
//%
//% @verbatim
//%
//% lvl_lo
//%
//% lvl_hi
//%
//%
//% state
//%
//%
//% trg_o
//%
//% @endverbatim
```



Module description

```
///% @reg_map
///%
///% Registers
///%
///% Register          | Address | Access | Description
///%
///% REG_ID            | 0x00    | R       | Identification register
///% REG_STATUS         | 0x04    | R       | Status register
///% REG_CONTROL        | 0x08    | R/W     | Control register
///% reserved           | 0x0C    | —      | —
///% REG_LVL_LO         | 0x10    | R/W     | Hysteresis low limit register
///% REG_LVL_HI         | 0x14    | R/W     | Hysteresis high limit register
///% REG_EVNT_CNT       | 0x18    | R       | Event counter
///%
///%
///%
///% Control register (REG_CONTROL)
///%
///% Field              | Bits   | Description
///%
///% reserved           | 31:9   | reserved
///% COMP_GT             | 8       | Triggers on pos pulses (see edge-detection-kernel)
///% reserved           | 7:1    | reserved
///% TRG_ENABLE          | 0       | Enables detection of trigger events
```

Doxygen HTML output: Module description

The screenshot shows a web browser window with the title "ESS-nBLM-500M: edge_detection". The address bar shows the URL: file:///home/jan/CAENels/Projects/ESS/ess-nblm-500m/nblm_damc_virtex/docs/output. The browser interface includes standard buttons for back, forward, search, and refresh.

The main content area has a header with tabs: Main Page, Related Pages, Design Unit List (which is selected), Files, Design Unit List, Design Units, Design Unit Hierarchy, and Design Unit Members. A search bar is also present.

Below the tabs, there are links for Inputs, Outputs, Signals, Module Instances, Parameters, and Always Constructs.

edge_detection Module Reference

Edge detection module. [More...](#)

Inheritance diagram for edge_detection:

```
graph TD; edge_detection_tb --> edge_detection; edge_detection --> edge_detection_slave
```

List of all members.

Always Constructs

<code>proc_delay_cntr (clk)</code>
<code>proc_state (clk)</code>
<code>ALWAYS_94 (clk)</code>
<code>proc_M_AXIS_TVALID (clk)</code>
<code>proc_trg_start_pr1 ()</code>
<code>proc_M_AXIS_TUSER (clk)</code>
<code>proc_M_AXIS_TDATA (clk)</code>

[Signals](#)

Doxygen HTML output: Register map

ESS-nBLM-500M: e x

file:///home/jan/CAENels/Projects/ESS/ess-nblm-500m/nblm_damc_virtex/docs/output

Register map:

Registers

Register	Address	Access	Description
REG_ID	0x00	R	Identification register
REG_STATUS	0x04	R	Status register
REG_CONTROL	0x08	R/W	Control register
reserved	0x0C	-	-
REG_LVL_LO	0x10	R/W	Hysteresis low limit register
REG_LVL_HI	0x14	R/W	Hysteresis high limit register
REG_EVT_CNT	0x18	R	Event counter

Status register (REG_ID)

Field	Bits	Description
ID_MAGIC	31:27	0x3D93_D37C (slightly reassembles EDGE DETC)

Status register (REG_STATUS)

Field	Bits	Description
reserved	31:0	reserved

Control register (REG_CONTROL)

Field	Bits	Description
reserved	31:9	reserved
COMP_GT	8	Triggers on pos pulses (see edge_detection_kernel)

Doxygen PDF output: Register map

The screenshot shows a PDF document titled "refman.pdf" with page 55. The left sidebar contains thumbnails for pages 55, 56, and 57. The main content area displays the "Module Instances" section, which lists two instances: "edge_detection_kernel::edge_detection_kernel [generate]" and "edge_detection_slave::slave_i". Below this is the "5.28.1 Detailed Description" section, which describes the Edge detection module. It states that the module performs edge detection on 4 signals in parallel and asserts one (and only one) of the bits in M_AXIS_TUSER signal when the trigger event is detected. The "Register map:" section follows, which is currently empty. At the bottom is the "Registers" section, containing a table with the following data:

Register	Address	Access	Description
REG_ID	0x00	R	Identification register
REG_STATUS	0x04	R	Status register
REG_CONTROL	0x08	R/W	Control register
reserved	0x0C	-	-
REG_LVL_LO	0x10	R/W	Hysteresis low limit register
REG_LVL_HI	0x14	R/W	Hysteresis high limit register
REG_EVT_CNT	0x18	R	Event counter

Conclusion

- Having an organized compilation flow makes jobs easier for everybody who wants to work on the code
- Modular design allows easy porting between different FPGAs and programs
- Testing is important for modular design - each should be tested separately
- Low-level documentation is a useful interface between HW and SW