SpinQuest V1495 Scaler Firmware & Interfacing

Ethan Hazelton
hazeltet@umich.edu

August 16, 2022

1 Design Overview

In order to upgrade the current SpinQuest scaler DAQ, new scaler firmware was designed on a V1495 FPGA board that will take the place of the Scale32 boards, which are 32 input scaler boards, and the Sis3610 interrupt module. The current firmware versions is FF1E.

1.1 V1495 Diagram

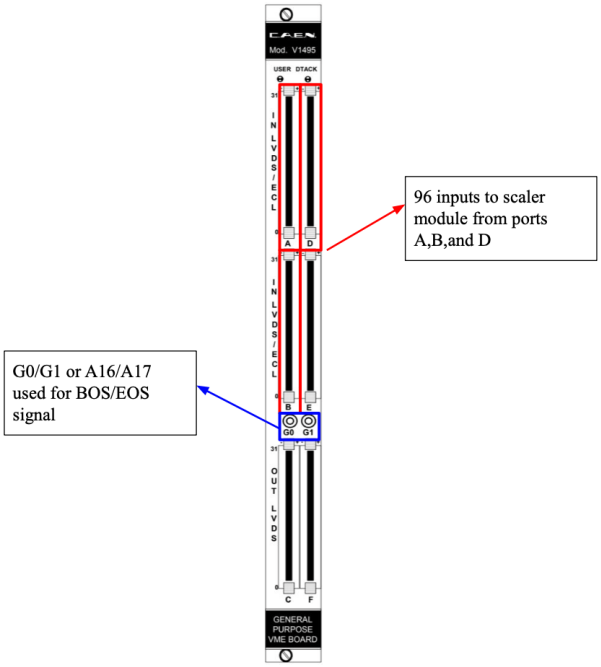

Figure 1: Diagram of V1495 from CAEN V1495 Manual

In Figure 1, the inputs to the scaler module are ports A,B, and D. Each port contains 32 inputs which will be referenced with an index starting at 0 (Ex: A0 is the first channel of A). D requires mezzanine card A395A according to the CAEN V1495 Manual. The BOS signal is inputted through G0/A16 and the EOS signal is inputted through G1/A17. These signals are important to enable/disable and reset the scaler. The EOS signal is also important for the interrupt.

1.2 Firmware Block Diagram

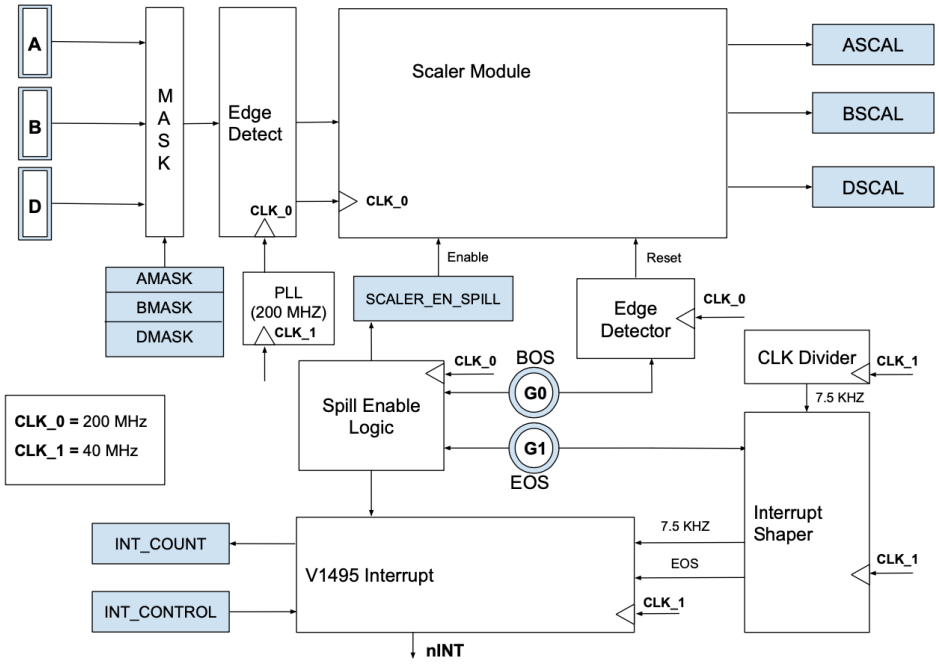

Figure 2: The block diagram of the FPGA design.

The scaler module counts the number of input pulses and stores those values on the scaler registers from their respective ports: ASCAL, BSCAL, and DSCAL. These registers are arrays of dimension 32x32bit to account for each of the 32 inputs from each port. The block diagram is shown in Figure 2. The sampling frequency is 200 MHz, generated by a PLL, indicating that the minimum input pulse width is 5 ns(100 MHz). The design contains a mask, controlled by the registers AMASK, BMASK,and DMASK, determining which channels will recieve input data. An edge detector is used to accurately detect each input pulse and synchronize the input with the 200 MHz clock. The scaler module is enabled between the BOS and EOS signals and reset at the BOS signal, more

of which will be discussed in section 4.3. VME interrupts are sent at a 7.5 KHz rate during the spill and 5 additional interrupts are sent at the rising edge of the EOS signal.

2 Control Registers

The control registers are the registers used to configure the scaler module. These registers have addresses in the range 0x1000-0x101C. The table is shown in Figure 3.

Name	Address	R/W	Description
SCRATCH	0x1000	RW	Test register. Default = "0xBEEF"
REVISION	0x1002	R	Firmware revision register
SCALER_EN_SPILL	0x1004	R	Enables/Disables scaler. Count is enabled at the BOS signal and is disabled at the EOS signal.
D_IDCODE	0x1006	R	ID of Mezzanine card for D
E_IDCODE	0x1008	R	ID of Mezzanine card for E
AMASK	0x100A-0x100C	RW	Controls what channels in A receive data
BMASK	0x100E-0x1010	RW	Controls what channels in B receive data
DMASK	0x1012-0x1014	RW	Controls what channels in D receive data
EMASK	0x1016-1018	RW	Controls what channels in E receive data
A_INT_CONTROL	0x101A	RW	Enables/Disables the interrupts. Interrupts are only sent when this register is '0x0001'.
A_INT_COUNT	0x101C	R	The number of interrupts sent by the V1495. This is useful to see missed interrupts on the software level.

Figure 3: This is a register table of the control registers

3 Scaler Registers

The scaler registers are used to record the count. Each of these registers are an array of 32 channels where each channel contains 32 bits. For example, the count from input A[0] is stored as a 32 bit value on ASCAL[0].These registers have addresses in the range 0x2000-0x217E. The table is shown in Figure 4.

Name	Address	R/W	Description
ASCAL	0x2000-0x207E	R	Holds 32-bit count from each channel of input A [32 channels]
BSCAL	0x2080-0x20FE	R	Holds 32-bit count from each channel of input B [32 channels]
DSCAL	0x2100-0x217E	R	Holds 32-bit count from each channel of input D [32 channels]

Figure 4: This is a register table of the scaler registers

4 Design Components and Test Benches

The firmware design is comprised of five major components: the clock divider, the edge detector, the spill enable logic, the v1495 interrupt and interrupt shaper, and the scaler component.

4.1 Edge Detector

The edge detector generates a one clock period wide pulse at the rising edge of the input. This pulse is generated at the next clock edge after the rising edge of the input pulse. The edge detector is used for the reset and the input channels to the scaler module. This is to assure that the reset pulse is a set width and the edge of the BOS signal was detected. The edge detector is used in the spill state logic to detect the edge of BOS and EOS as well. Lastly, the edge detector is used for the interrupt shaper as well.

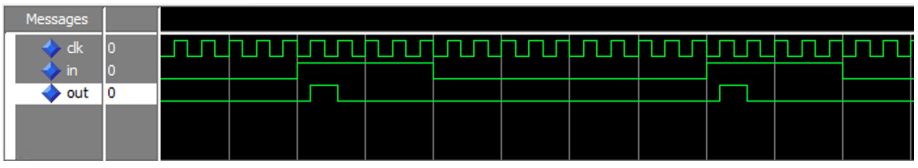

Figure 5: This is the waveform generated from "edge_detector_tb.vhd"

The waveform of edge detector testbench is shown in Figure 5.

4.2 Clock Divider

The clock divider is used to generate clocks with a frequency less than the V1495 local clock, which has a frequency of 40 MHz. This component is used to generate the 7.5 KHz pulses needed in the interrupt. Additionally, it is used to generate additional clocks used concurrently with the edge detector to shape the interrupt pulses.

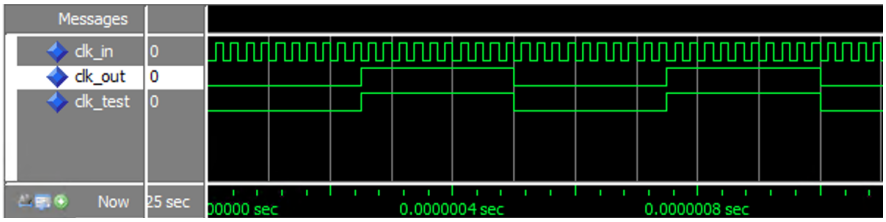

Figure 6: This is the waveform generated from "clk_divider_tb.vhd"

The waveform of clock divider testbench is shown in Figure 6. The signal "clk_out" is the clock generated by the clock divider. The signal "clk_test" is the expected generated clock from the clock divider. As shown, "clk_out" and "clk_test" are the same, meaning the clock generated from the clock divider is the correct frequency.

4.3 BOS/EOS Scaler Control

The BOS signal is connected through G0 or A16. The EOS signal is connected through G1 or A17. At the rising edge of the BOS signal, SCALER_EN_SPILL is set to '1', enabling the scaler module. The scaler module is then disabled at the rising edge of EOS. The BOS signal activates the reset in this case through its input into the pulse stretcher. This module is clocked with the 200MHz clock generated by the PLL.

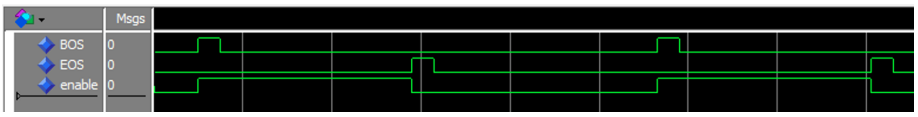

Figure 7: This is the waveform generated from "spill_state_tb.vhd"

The BOS/EOS scaler control module has a file name of "spill_state.vhd". The testbench waveform is shown in Figure 7, proving that the enable is indeed activated at BOS and deactivated at EOS.

4.4 Scaler Component

The scaler component counts the number of pulses received through each input and stores that value. Two separate modules were used to implement a single channel scaler component for many different channels. The input is sampled at a 200 MHz clock, meaning the maximum input frequency is 100 MHz, or a minimum 5 ns pulse width. The edge detector is used to a generate pulse after detecting the rising edge of the input. The scaler counts the number of edge pulses.

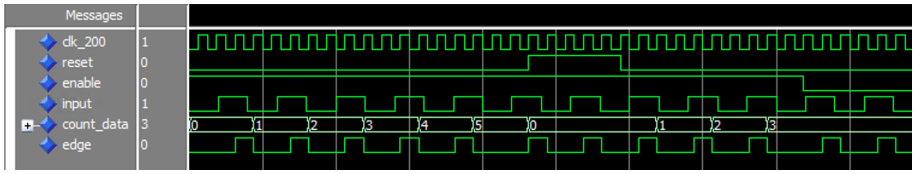

Figure 8: This is the waveform generated from "scal_single_ch_tb.vhd"

The single channel scaler component has a file name of "scal_single_ch.vhd". This module only implements a counter for a single channel. The count is stored in a 32-bit variable. The waveform of the testbench is shown in Figure 8.

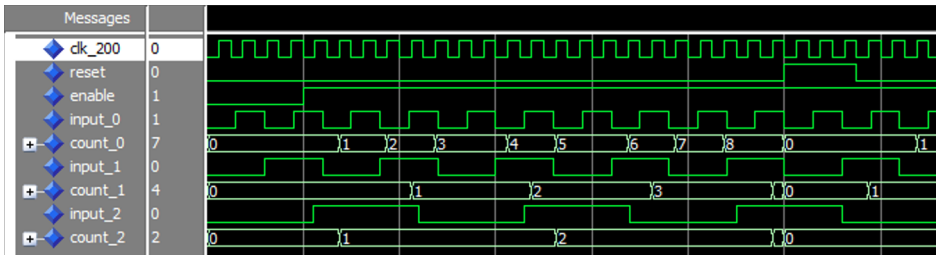

Figure 9: This is the waveform generated from "scal32_mult_ch_tb.vhd"

In the multi-channel scaler, the single channel component is used repeatedly based on the number of inputs.The multi-channel scaler component has a file name of "scal32_mult_ch.vhd" The waveform of the testbench is shown in Figure 9. Three inputs were used to demonstrate that the multi-channel scaler implements multiple channels correctly. This module can be scaled for any number of channels. In the case of this firmware, the scal32_mult_ch module was implemented once for each port, with each multi-scaler module containing 32 inputs.

4.5 V1495 Interrupt & Interrupt Shaper

Interrupt functionality is necessary for the V1495 firmware to replace the Sis3610 interrupt module as well. The V1495 contains a single pin that raises the interrupt on the VMEBus back plane. User interrupts can be set by driving this active low pin with user logic. In this case, the interrupt must be raised at the rising edge of a 7.5 KHz clock during the spill, when enable is high, and at the EOS signal. In order to achieve this, a clock divider is used to generate the 7.5 KHz clock. Another clock divider and edge detector are used in conjunction to generate 500ns wide pulses at a 7.5KHz rate to be outputted to the V1495 interrupt pin, "nINT", during the spill. At the rising edge of the EOS signal, five 100 μs wide pulses are generated. These pulse widths are much larger in order for the VME controller to have time to read registers to determine the interrupt event type. Five pulses are generated to prevent an EOS interrupt being missed.

The interrupt is enabled when "int_control" is high. The number of interrupts sent during the spill is stored in the "int_count" register. This is used to analyze whether the VME controller misses any interrupts.

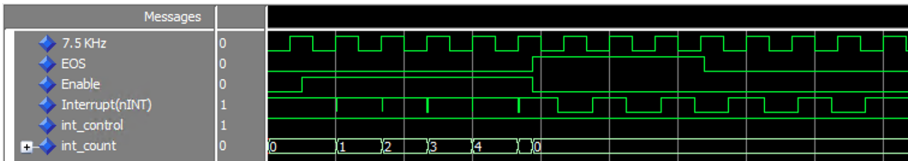

Figure 10: This is the waveform generated from "v1495_int_tb.vhd"

The testbench of the V1495 interrupt and interrupt shaper is shown in Figure 10. The 7.5 KHz clock in this testbench is generated by the clock divider. As shown, the interrupt is driven by the 7.5 KHz clock during the spill and by the delayed EOS signal after the spill.

5 Additional Important Elements

5.1 V1495usr_scaler and V1495usr_hal

V1495usr_hal is a hardware abstraction layer that is necessary for the correct hardware fitting. V1495usr_scaler is the top level entity.

5.2 V1495scaler_pkg

This is a VHDL package that contains the addresses of each register,and declares important functions and variables including the firmware revision number.

5.3 V1495_reference

V1495_reference connects each of the modules described in Section 4 to the required control and scaler registers. Masking, signal redirection, and register read/write functionality is implemented in this module.

6 Software Interfacing

CODA and the specific V1495 drivers are used to recieve the interrupts from the V1495 and read out the correct registers. This is managed using the CRL code. Additionally, CODA requires additional files in order to manage each interrupt correctly.

A backup was made for any file that was changed. The previous files all have the tag ".scale32" at the end of the file name. For example, "gen_int_list.crl.scale32" is the backup for "gen_int_list.crl".

6.1 Drivers

Two drivers are used by the V1495 with files: v1495.c/v1495.h and v1495scaler.c/v1495scaler.h. These can be found on the e1039sc4 computer with path "/home/e1039daq/V1495scaler_daq/scaler_driver". Both compiled driver object files are referenced in the VME controller boot script in order to be used.

6.1.1 v1495.o

This driver contains functions needed to upload new firmware to the V1495.

6.1.2 v1495scaler.o

This drive contains specific functions needed for readout using this specific firmware. "v1495Init()" initializes the v1495 using the correct address of the board as an argument. Two functions, "v1495ScalerStatus()" and "v1495ScalerData()", are used to readout registers from the board directly from the VME controller screen. These functions are not to be used in the CRL code. "v1495ScalerStatus()" prints out the control registers. "v1495ScalerData()" prints the scaler registers using an integer as an argument that correlates to a specifc port (1 = A, B = 2, C= 3). The function, "v1495ScalerCodaRead()", takes an argument for the specific port(1 = A, B = 2, C= 3) and an argument for which 16 bit scaler register. The scaler registers in the drivers are arrays of unsigned shorts with size 64. For example, ascal[0] is the high 16 bits of the scaler register for A0 and ascal[1] is the low 16 bits of the scaler register for A0. "v1495ScalerCodaRead()" is needed in the CRL code to output 32-bits from each scaler register into the CODA buffer stream. For example, "v1495ScalerCodaRead(1,2)" will output the entire 32-bit scaler register for the channel A1.

6.2 CRL

The CRL code provides information to the VME controller on what registers to read from the V1495 based on different interrupt types(1 or 2). The CRL code used for the Scaler DAQ is named "gen_int_list.crl". This can be found on the e1039sc4 computer with path "/usr/local/coda/2.6.1/extensions/e906".

This new CRL code mimics the previous CRL code but uses separate event types to distinguish what registers must be printed. The registers are readout using the "v1495ScalerCodaRead()" function. Previously, the EOS and BOS events were found from using two channels on the Scale32 boards and the scalers were enabled at the CRL level. Now, the scaler is enabled due to EOS and BOS signals inputted at the firmware level. The CRL code detects the EOS from type 2 physics events. The 7.5 KHz interrupt is found from type 1 events, which only come during the spill. Different registers must be readout based on the EOS and 7.5KHz interrupts.

Additionally, there is a section of code that prints an error message if a 7.5 KHz interrupt is missed.

6.3 CODA Interrupt files

CODA requires a filed named "GEN_source.h" for the VME controller to acknowledge interrupts and determine the event type. This file is also found on the e1039sc4 computer with path "/usr/local/coda/2.6.1/extensions/e906". This code has been edited to acknowledge the V1495 interrupts instead of the Sis3610 interrupts.

To determine the event types, the SCALER_ENABLE_SPILL register is readout. If this register is '0', the event is a type 2 event (EOS), otherwise the event is a type 1 event (during the spill).

6.4 RunFFT Coda Decoder

RunFFT is the CODA decoder for the Scaler DAQ. The code for this can be found on the e1039sc4 computer with path "/data2/e1039/daq/seaquest-daq/E906ScalerDAQ/RealtimeMonitor". The file "E906ScalerDAQ.C", a file that defines a class used for the coda decoder, was changed in order for the coda decoder to detect different event types and guarantee that only a single EOS interrupt is readout. This file needed to be edited in order for the Scaler DAQ to use this V1495 firmware. Two specific functions in this class were changed: "E906DAQ_goodevent_check()" and "E906DAQ_endofspill_check()".

6.4.1 E906DAQ_endofspill_check()

This function detects the EOS event using the event type and the previous event type. The EOS event type is type 2 while the events that happen during the spill are type 1. The previous event type is stored in a variable named "tp_prev". Specifically, a boolean variable named "eosflag" is used to flag an EOS event. This function assures that "eosflag" is only true for a single EOS event. This

is important because "eosflag" is used as an argument to determine whether histograms will be updated with the latest EOS data.

6.4.2 E906DAQ_goodevent_check()

This function is the first check that determines what interrupt events will be readout. Using the previous event type stored in "tp_prev" and the current event type, only type 1 events and a single type 2 EOS event is allowed to be read through this check. Multiple EOS interrupts are sent using this firmware and the CODA decoder is used to take a single EOS event and discard the rest.

7 Using this firmware with the Scaler DAQ

All firmwares needed for the Scaler DAQ are located on the e1039sc4 computer with path "/home/e1039daq/V1495scaler_daq/".

7.1 To use/test with the emulator:

Upload the firmware "v1495acc_em_100_FF05.rbf" to the V1495 that will be used as an emulator. This firmware sends 100 MHz pulses out of the C port during the spill. The BOS signals is outputted through G0 and the EOS signal is outputted through G1 in TTL standard. Upload the scaler firmware "v1495usr_scaler_FF1E_w_emulator.rbf" to the Scaler DAQ V1495 board. This firmware sets G0/G1 to TTL standard. Connect the G0 to G0 of the other board and do the same for G1.

7.2 To use for Scaler DAQ:

Upload the scaler firmware "v1495usr_scaler_FF1E_daq.rbf" to the Scaler DAQ V1495 and connect the SC0 cables to port A, SC1 to port B, and SC2 to port D. The BOS and EOS signals should connect to A16 and A17 through the SC0 cable.

8 Source Code

The source code can be found on the e1039sc4 computer with path "/home/e1039daq/V1495scaler_daq/V1495_Scaler_Firmware_FF1E_3port_final". Contact hazeltet@umich.edu for access to github with older versions of the firmware at https://github.com/hazeltet845/V1495scaler_daq_upgrade_spinquest