CBETA FAST SHUTDOWN SYSTEM

Tommy Tang, Rob Michnoff, Rob Hulsart, Julio Renta

APRIL 8, 2020

BROOKHAVEN NATIONAL LABORATORY
Building 924

System Description 3
Hardware System 3
Schematic 6
Function Diagram 9
FPGA Firmware 10
Setup 10
Clock Management 10
I/O Planning 10
Output Control Signals (or_gate instance, see Figure-7 for the schematic) 10
User Enable Control for Input Signals 11
Front End Control Software Interface (See Figure-9 for the block diagram) 11
Failed Device Alert 12
Control Software 13
Parameters .... **Error! Bookmark not defined.**
Records .... **Error! Bookmark not defined.**
Enable Records 14
Input Status & Latched Status Records 14
Reset Record 14
Subroutine Functions 15
CS-Studio GUI 15
Reset 16
Enable Switches 16
LED Lights 16
Raw Status 16
Latched Status 16
System Mode 16
Reference 19

MicroZed Breakout Carrier Card Zynq System-on-Module Hardware User Guide: 19
MicroZed Schematic Rev G: 19
Xilinx 7 Series FPGA Libraries Guide for Schematic Designs: 19
EPICS Build Procedure 19
EPICS Manual and References: 19
EPICS Manual and References: 19
EPICS Manual and References: 19
EPICS Manual and References: 19
EPICS Manual and References: 19
MicroZed Hardware User Guide: 19
MicroZed Quick Starter Guide: 19
BDF files for downloading: 19

System Description

The system was designed to serve as a complimentary component for fast beam loss monitor. There were three identical fast shutdown chassis in the system. As shown in Figure-1, the digital output of two chassis were connected to the inputs of the third chassis while both were connected to the fast beam loss monitor through their input connectors. The purpose of the system was to combine sixteen single-ended input signals to inhibit the beam when one or more input digital signals indicated a “Fail” state. The output signals were expected to indicate a “Fail” state as soon as any one of the connected devices reached a logical zero. The delay was expected to be only the propagation delay from the cable length and the electronics. The digital output signal can be used to either connect to the machine protection system to inhibit the beam or to cascade to another fast shutdown system chassis. In the initial design scope, the system was a complimentary component of CBETA fast beam loss monitor which features a Red Pitaya system in the architecture. Later the design scope was expanded. The new design implementation created some leeway for other MPS devices in the future though necessary modifications might be needed if connected MPS devices have different signal behaviors.

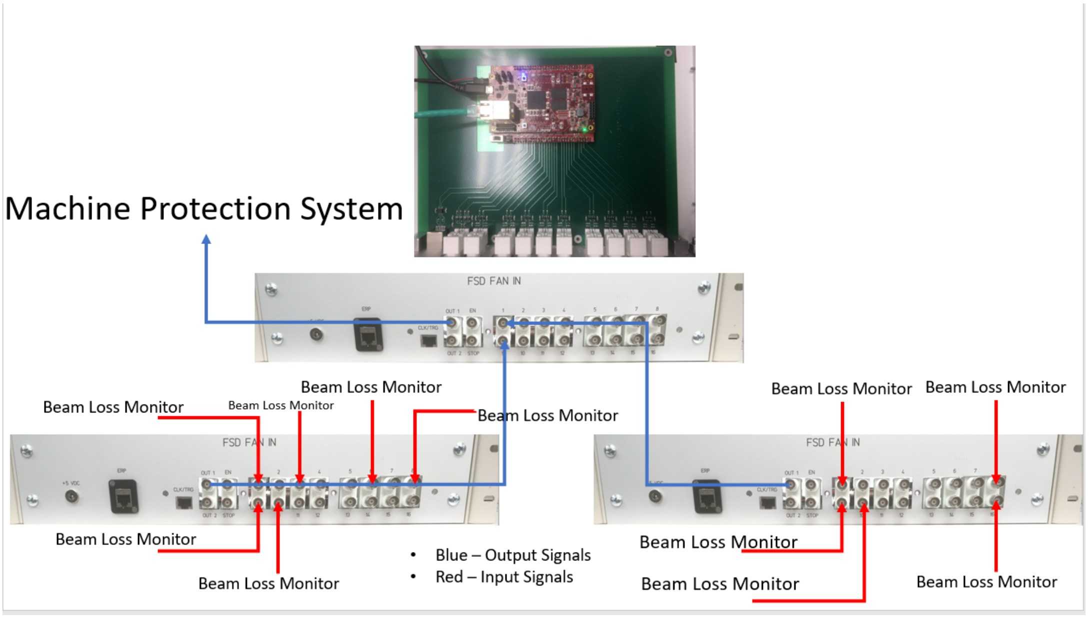


Figure – 1

Hardware System

The electronics are housed inside a customized chassis. The chassis has sixteen BNC connectors for input devices, two BNC connectors for BPM signals, two BNC connectors for output signals, one ethernet connector and one 3.3V power jack. The MicroZed FPGA board is chosen as the main FPGA/processor for the hardware system. The MicroZed FPGA used for the system is MicroZed 7010 board (See Figure-2). For more details about MicroZed board. See Reference[10] & Reference[11].

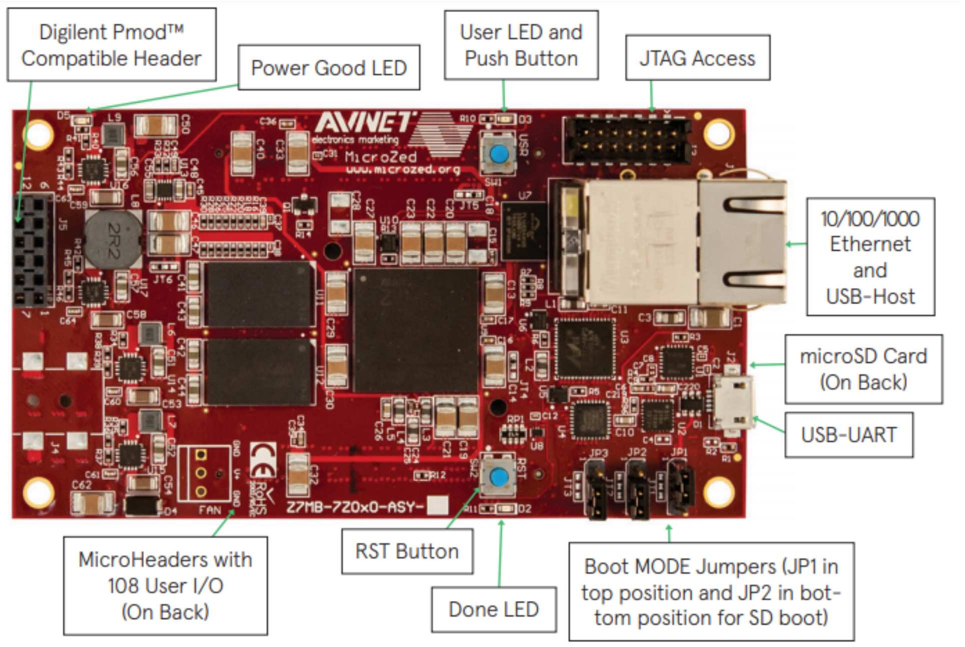

Digilent Pmod™ Compatible Header
Power Good LED
User LED and Push Button
JTAG Access
10/100/1000 Ethernet and USB-Host
microSD Card (On Back)
USB-UART
MicroHeaders with 108 User I/O (On Back)
RST Button
Done LED
Boot MODE Jumpers (JP1 in top position and JP2 in bottom position for SD boot)

Figure-2

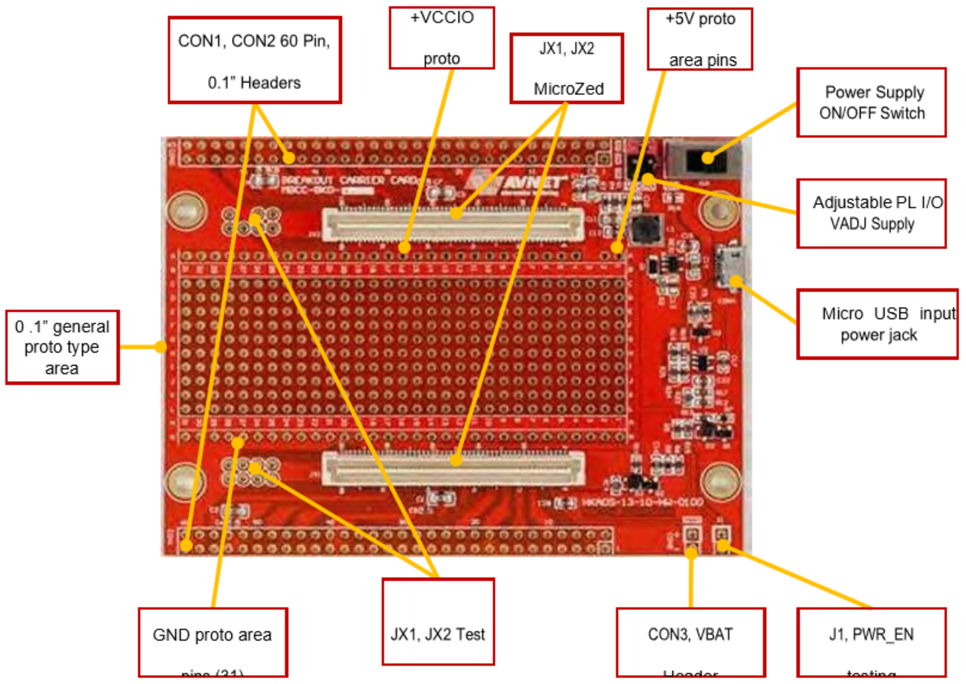

CON1, CON2 60 Pin, 0.1" Headers
+VCCIO proto
JX1, JX2 MicroZed
+5V proto area pins
Power Supply ON/OFF Switch
Adjustable PL I/O VADJ Supply
Micro USB input power jack
0 .1" general proto type area
GND proto area
JX1, JX2 Test
CON3, VBAT
J1, PWR_EN

Figure 2 – BKO-CC Board Functions

Figure-3

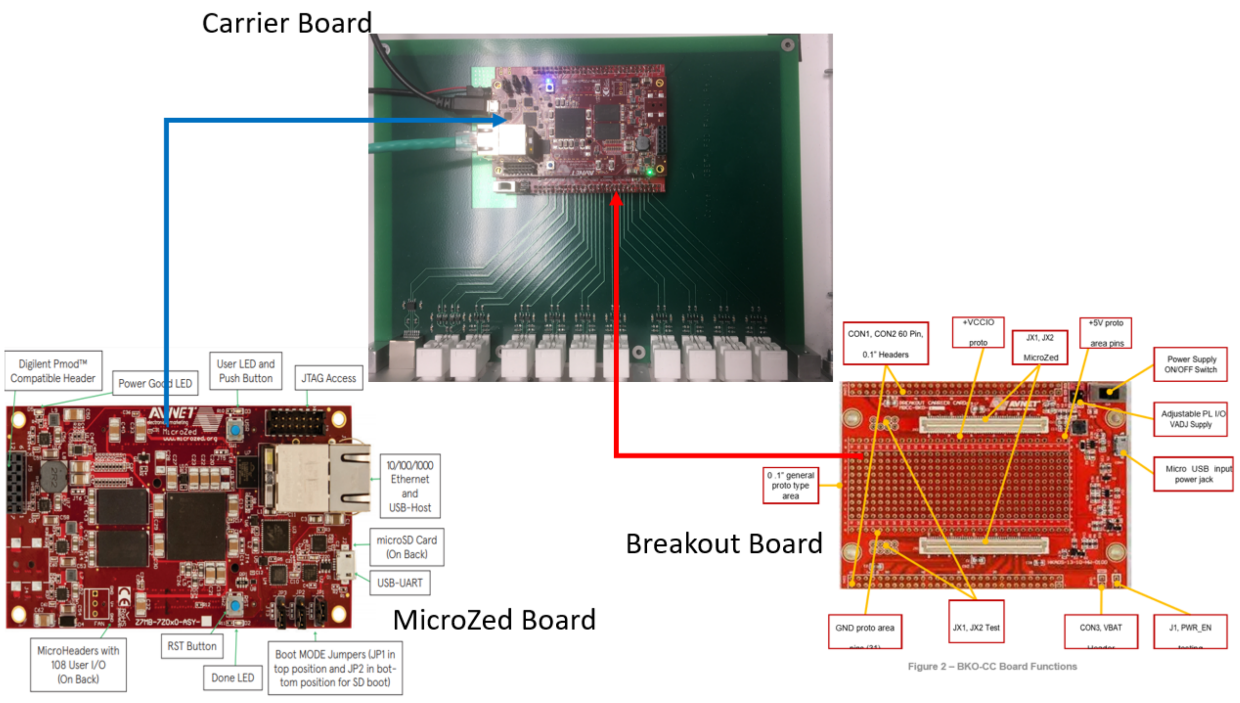

Carrier Board
Digilent Pmod™ Compatible Header
Power Good LED
User LED and Push Button
JTAG Access
10/100/1000 Ethernet and USB-Host
microSD Card (On Back)
USB-UART
MicroHeaders with 108 User I/O (On Back)
RST Button
Done LED
Boot MODE Jumpers (JP1 in top position and JP2 in bottom position for SD boot)
MicroZed Board
Breakout Board
CON1, CON2 60 Pin, 0.1" Headers
+VCCIO proto
JX1, JX2 MicroZed
+5V proto area pins
Power Supply ON/OFF Switch
Adjustable PL I/O VADJ Supply
Micro USB input power jack
0.1" general proto type area
GND proto area
JX1, JX2 Test
CON3, VBAT
J1, PWR_EN
Figure 2 – BKO-CC Board Functions

Figure-4

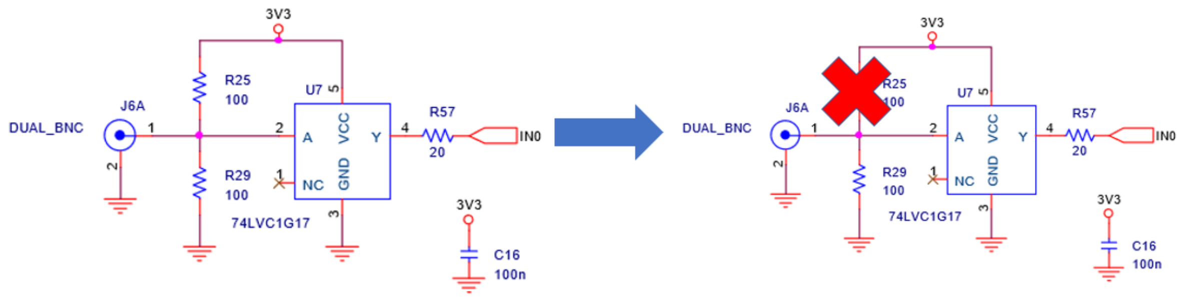


Figure -5

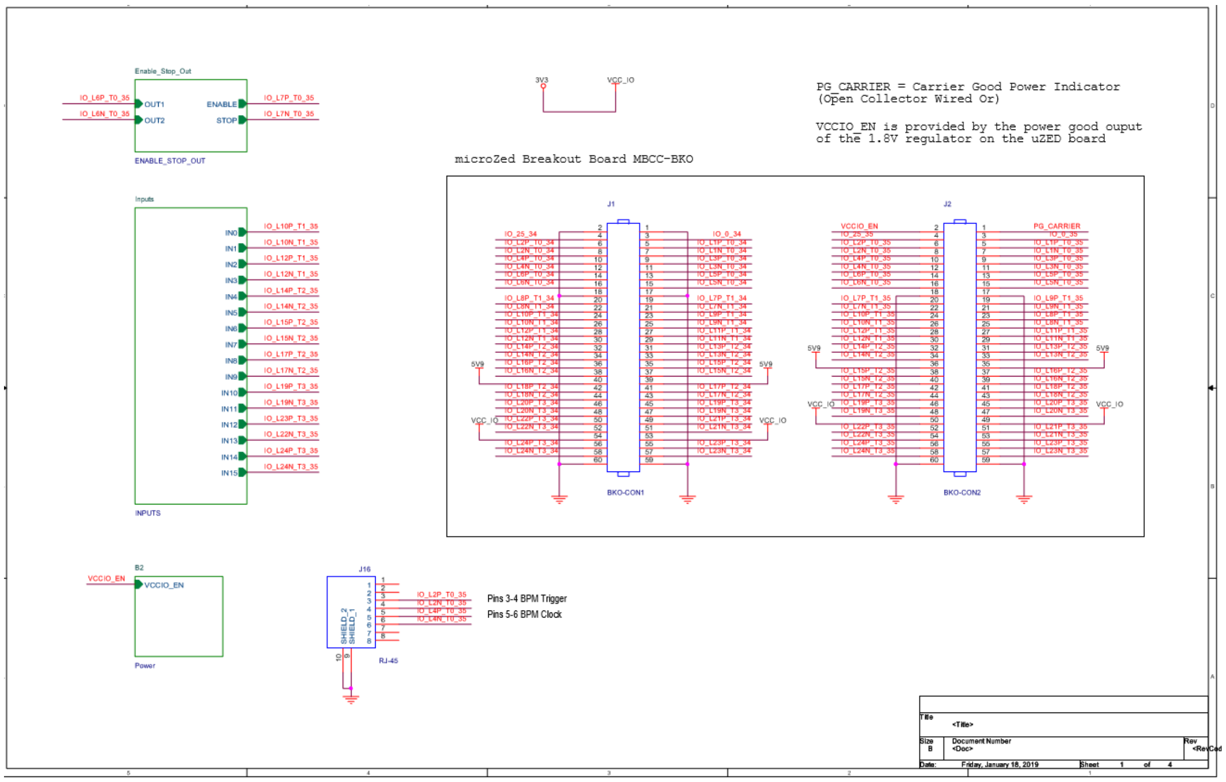
It is placed inside the chassis and connected to MicroZed Breakout Board (Figure-3) and PCB carrier board. See Figure-4. The Breakout board provides a simple solution to access I/O pin on the MicroZed board externally. It is connected to MicroZed through JX1 and JX2 connectors. There are two 60-pin headers CON1 and CON2. The pins on the on the pin headers are routed to designated I/O ports through circuit board. The PCB carrier board is designed by John Dobbins, an electrical engineer from Cornell University. The PCB board provides necessary circuitry for the I/O interface between the external signals and the boards. The input I/O circuit consists of pull-down, pull-up resistors and Schmitt triggers. It can take up to 3.3V LVDS signal and Schmitt triggers prevents possible spikes in the input signals from disturbing the system behavior. However, during the development, a parallel of pull-down and pull-up resistors forced the input voltage level to be at mid-level between logic 0 and logic 1 when it was disconnected or not connected, thereby creating a latch. In case when it was disconnected when it was previously in a high state, the high state would be latched and failed to detect potential failure. To be a safe operation system, any open input should be detected as a "Fail" state. It was speculated that one of the reasons that there was a pair of 100-ohm resistors in parallel was to create a 50-ohm termination, but it was not entirely essential to have a 50-ohm termination. As a result, as Figure-5 shown, the pull-up resistor to 3.3V was removed. With only pull-down resistor being present, the voltage level at the inputs when there was no signal connected would be pulled down to low level. It was also discovered that the resistors value for two of the sixteen input I/O circuitry were wrong and were replaced. Also, the label on the front panel was not consistent with the schematics and PCB routing to the FPGA pins. Table-1 is a summary of connections among PCB ports, Zynq AP Soc Pin & Connection. See Reference[1] and Reference[2] for more additional information.

Inputs/Outputs	Zynq AP SoC Pin Name	Zynq AP SoC Pin Connection
IN0	IO_L10P_T1_35	Bank 35, K19
IN1	IO_L10N_T1_35	Bank 35, J19
IN2	IO_L12P_T1_35	Bank 35, K17
IN3	IO_L12N_T1_35	Bank 35, K18
IN4	IO_L14P_T2_35	Bank 35, J18
IN5	IO_L14N_T2_35	Bank 35, H18
IN6	IO_L15P_T2_35	Bank 35, F19
IN7	IO_L15N_T2_35	Bank 35, F20
IN8	IO_L17P_T2_35	Bank 35, J20
IN9	IO_L17N_T2_35	Bank 35, H20
IN10	IO_L19P_T3_35	Bank 35, H15
IN11	IO_L19N_T3_35	Bank 35, G15
IN12	IO_L23P_T3_35	Bank 35, M14
IN13	IO_L22N_T3_35	Bank 35, L15
IN14	IO_L24P_T3_35	Bank 35, K16
IN15	IO_L24N_T3_35	Bank 35, J16
OUT1	IO_L6P_T0_35	Bank 35, F16
OUT2	IO_L6N_T0_35	Bank 35, F17

Table-1

Schematic

Figure-6 and Figure-9 provide the schematic for the hardware system.



PG_CARRIER = Carrier Good Power Indicator (Open Collector Wired Or)
VCCIO_EN is provided by the power good ouput of the 1.8V regulator on the uZED board
microZed Breakout Board MBCC-BKO
Enable_Stop_Out
ENABLE_STOP_OUT
Inputs
INPUTS
J1
J2
BKO-CON1
BKO-CON2
J16
RJ-45
Pins 3-4 BPM Trigger
Pins 5-6 BPM Clock
Power
VCCIO_EN

Figure-6

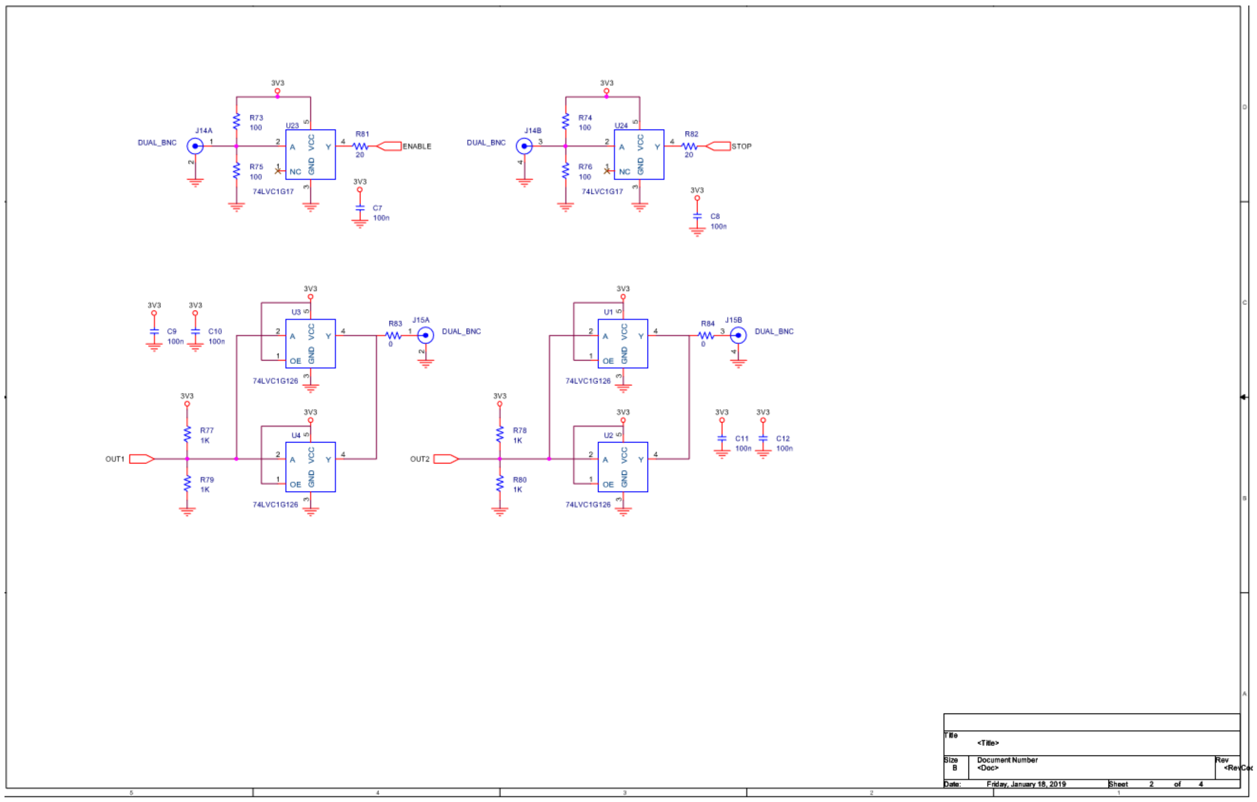


Figure-7

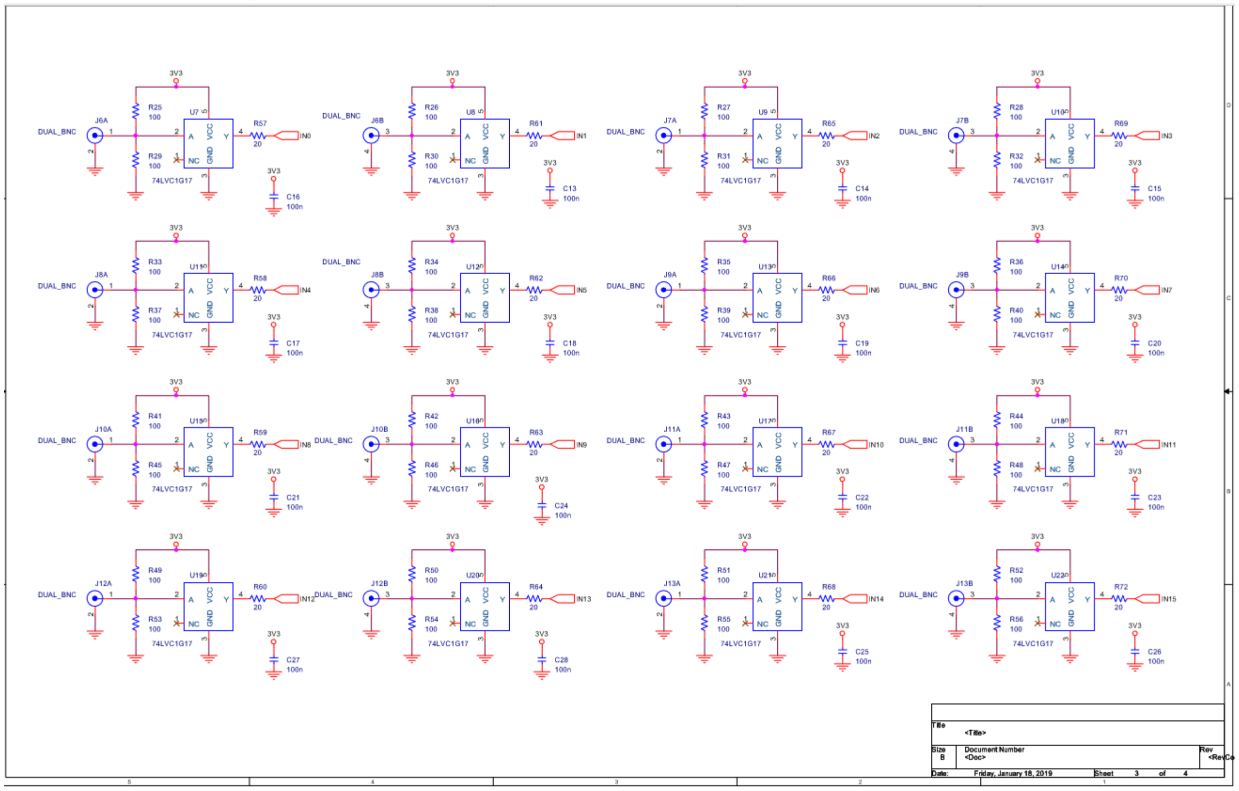


Figure-8

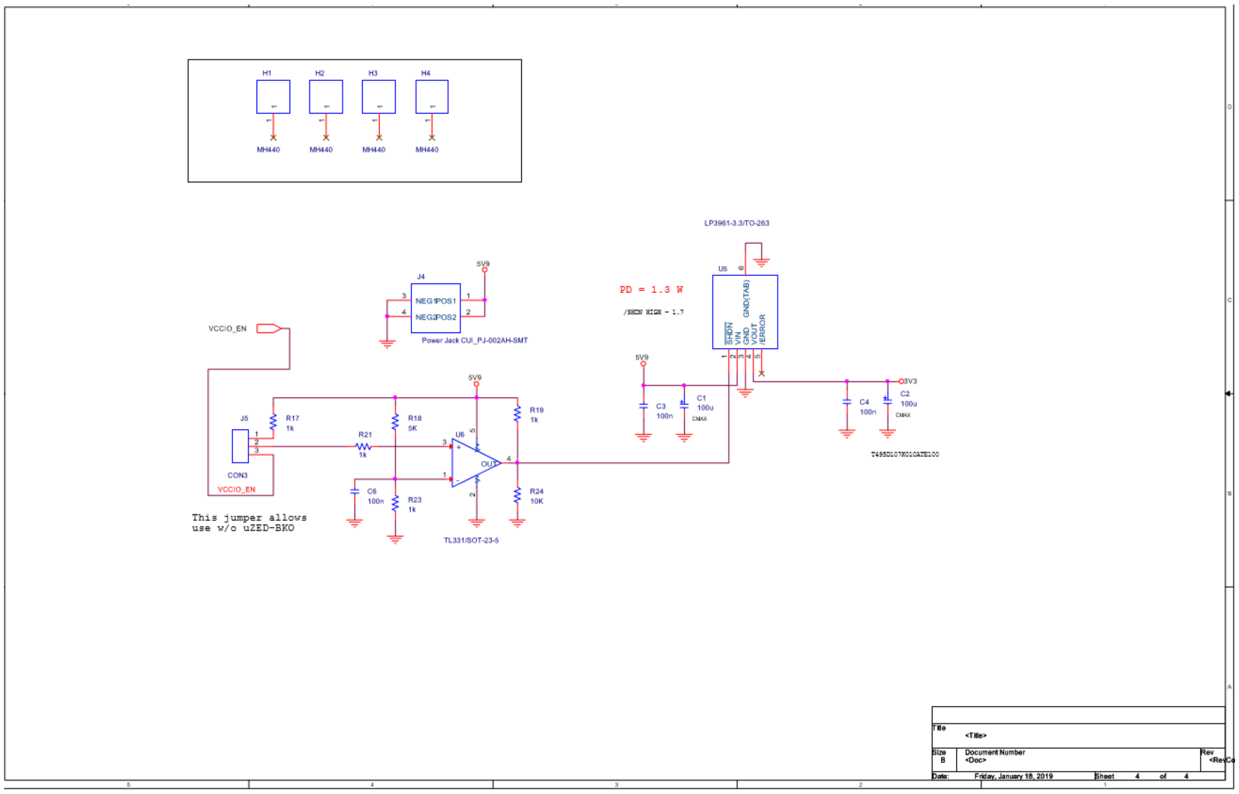


Figure-9

Function Diagram

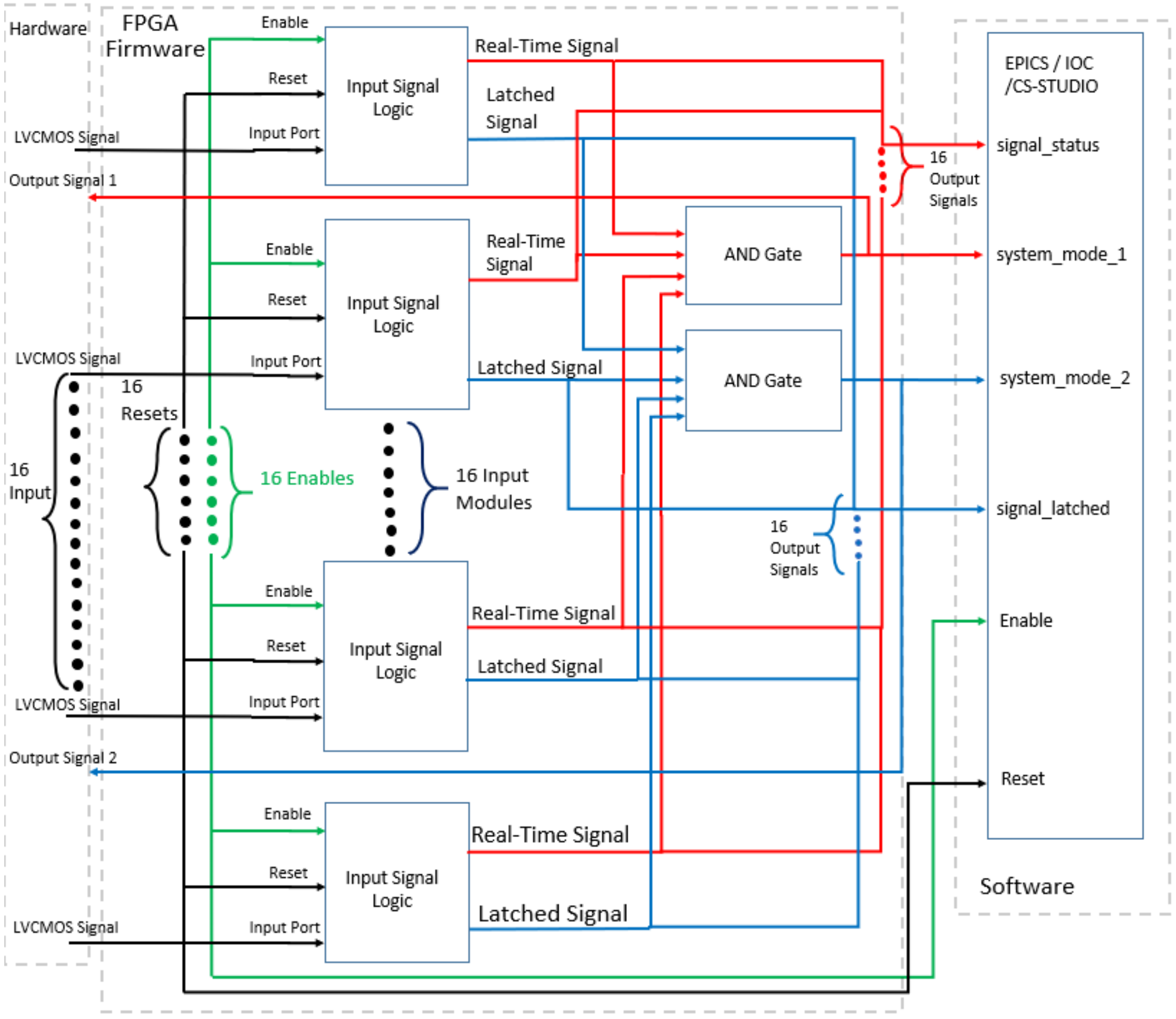


Figure-10

The red lines are all the signals associated with real-time output.

The green lines are all the signals associated with enable.

The blue lines are all the signals associated with latched output.

The black lines are all the signals associated with reset and input port.

The dashed region shows the territory of hardware, firmware and software system.

Only four instances of sixteen instances are show in the function diagram and the rest of them are omitted and replaced by dots and curly brackets.

FPGA Firmware

The firmware was implemented with an AND gate to combine all 16 one-bit signals logically into a one-bit output signal so that if one of the signals goes low (logical zero), the output of the AND gate will go to a logical zero. With current firmware configuration, since there were two output ports, one of the outputs was an “AND” gate of 16 raw signals from the I/O ports, and the other one was an “AND” gate of latched signals from the I/O ports. Strictly speaking, under CBETA’s fast beam loss monitor system, the Red Pitaya issued a logic level change in the signals. Using the raw port signals was enough to achieve the goal. However, with a boarder scope of application discussed later, using the latched signals might be better for different devices in different applications. The current firmware setup was the best for not only the demonstration purpose but also for the testing purpose as it had a combination of both types of the signals. Depending on what application and devices would be used with the system, this part of the configuration was subject to change.

Setup

Since the MicroZed board is distributed through the third-party vendor Avent, Vivado software doesn’t have the board description file in the folder. The board description file can be downloaded from the Github link and on a Windows PC system, moved the folder under <install_location>\Vivado\2019.2\data\boards\board_files.

Clock Management

The system clock is a 100 MHz PL-fabric clock derived from the processor block.

I/O Planning

The I/O ports must be configured in the FPGA to give firmware their accessibility. There is a total of sixteen inputs and two outputs. All the I/O ports are located at Bank 35 of the I/O bank on the MicroZed. Based on Table-1 above, the digital signal is assigned to the correct SoC pin name and configured to be a single-ended 3.3V LVCMOS signal. There are no pull-down or pull-down resistors added from the FPGA side because the system relies on the ones on the PCB board.

Output Control Signals (or_gate instance, see Figure-11 for the schematic)

The firmware uses IBUF instances to take sixteen external digital signals from the I/O connections on the hardware. Each signal is assigned as an input pin going into the or_gate instance. The block logically combines all sixteen digital signals by using an “AND” gate and generates two digital output signals. With the current implementation, the output signals will immediately respond to digital level changes in any one of the sixteen digital signals. These two digital output signals are passed to the output pins on the hardware by using OBUF instances. Along with the I/O ports, the or_gate block also requires other control pins and a clock pin. It also generates registers going into the software.

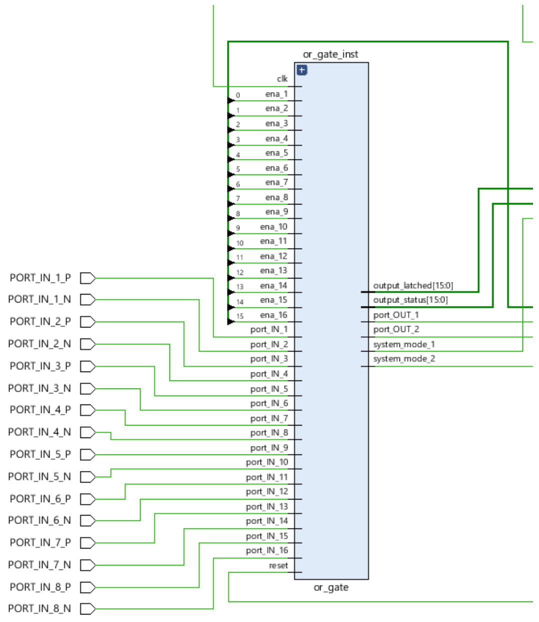


Figure-11

User Enable Control for Input Signals

There are 16 signals used for the enable signals to control the port signals. When the enable signal is a logical zero, the port signal connects to the I/O port. When the enable signal is a logical one, the port signal is tied to a logical level high. A logical zero indicates an enabled input from the user input whereas a logical one indicates a disabled input from the user input. The setup guarantees that when the board is first initialized, all the input signals will be enabled. Users have a choice to enable or disable any input channel if they are so to choose not to connect or use them.

Front End Control Software Interface (See Figure-12 for the block diagram)

The firmware also consists of a customized IP block which has the hardware registers to be accessed from the software side. There are four signals connected to this IP block. The enable signal is a 16-bit signal that each enable pin for the port signal is concatenated to form a 16-bit signal. Likewise, ouput_status and out_signal are 16-bit signals that each port signal and each latched signal is concatenated into two 16-bit signals. This IP block serves as an interface between firmware and software control. With the AXI interface, the IP is assigned with a base address at 0x43C00000 by using the address editor. The software can retrieve the value of a register by accessing its hardware address, and depending on the type of the register, the software can either write to or read from the hardware address. See more details in the next chapter, Control Software.

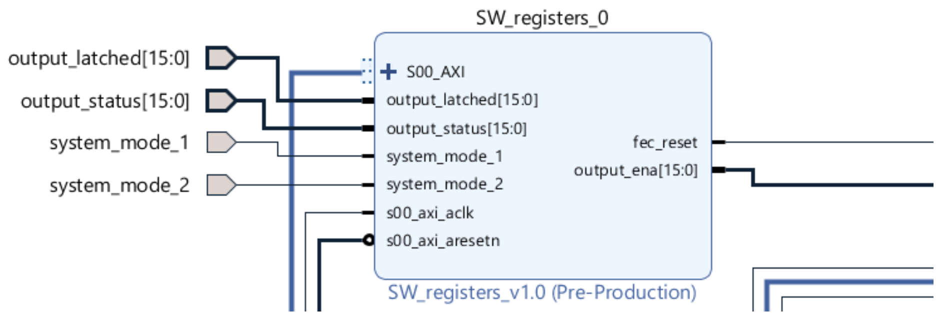


Figure-12

Failed Device Alert

The system latched the signal from the I/O ports whenever the device is in a failed state. A low-level (logical zero) signal indicates a failed state. The failed state will be latched until the user resets it. This feature is implemented by inferring a D-flip flop with negative-edge clock, clock enable and asynchronous preset for each latched signal. The asynchronous preset (PRE) input is tied to the user reset to release the latch. The asynchronous user reset is asserted by the user issuing a software command, latched high for three clock cycles and de-asserted by itself. The clock enable (CE) is always tied to high. The clock is connected to each port signal. By treating the port signal as a clock signal, the flip flop checks the falling edge of the port signal when it indicates a failed state from the device, and the falling edge stores/latches the value of the data which is zero because when there is a falling edge the output should be latched to zero until the user releases it. The data pin, as a result, is always tied to a logical zero. Figure 13 shows a layout of this type of D flip flop and Table-2 presents its logic table.

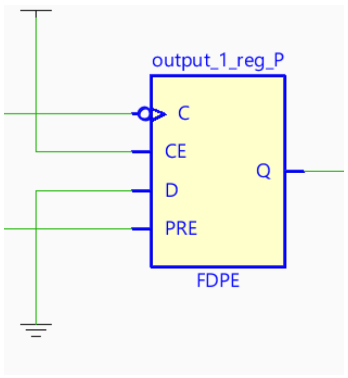


Figure-13

Inputs				Outputs
PRE	CE	D	C	Q
1	X	X	X	1
0	0	X	X	No Change
0	1	D	↓	D

Table-2

One of the issues occurred for this implementation is that since it only looks at the falling edge of the port signal, after the latch is released by the user and if the port signal still stays low, without a falling edge, the output will not be latched to zero again. To fix this problem, the latched output will only be released to one if the input port is a logical one. In practice it creates a latch in addition to the previous D flip flop.

Control Software

The control software is built on the EPICS IOC platform. A version of Petalinux is loaded onto the SD card. The Petalinux is customized and rebuilt to have necessary network features, such as MAC address, DHCP, hostname and SSH. The IOC was built in /ride/EPICS/base-3.14.12.5/MZED. Several files were changed to make the necessary changes. They were the following:

/ride/EPICS/base-3.14.12.5/MZED /mzedApp/src/ReadData.c

/ride/EPICS/base-3.14.12.5/MZED/mzedApp/src/bpm301.hxx

/ride/EPICS/base-3.14.12.5/MZED/mzedApp/Db/Data.db

/ride/EPICS/base-3.14.12.5/MZED/dbd/mzed_subroutines.dbd

/ride/EPICS/base-3.14.12.5/MZED/iocBoot/iocMZED/autosave/asList.req

ReadData.c and bpm301.h were changed to add C code for the new subroutine functions and new memory addresses. Data.db was changed to add more records and subroutine records. mzed.dbd was changed to incorporate the new subroutine functions that were called in the new records. asList.req was changed to add new record into the cache/initialization file. The source code then was built in the same directory. The mzed binary file, Data.db file, mzed.dbd file and asList.req file were copied into MicroZed linux system. The IOC system was started by running a script called init.epics. The CS-Studio application is located at /ride/EPICS/base-3.14.12.5/cs-studio. In order to connect CS-Studio with the MicroZed, the IP address of MicroZed was configured into CS-Studio application.

Records

There are 51 records in the software control system. Table-3 shows a brief summary followed by some detailed implementation in the next sections.

Number of Records	Name of a Record	States of a Record	Function of a Record
16	inputEna1S, inputEna2S, ….., inputEna16S	Normal, Tripped	Enable/disable the input signals
16	inputStatus1S, inputStatus2S, …, inputStatus16S	Normal, Tripped	Indicate the real-time signal status
16	inputLatch1M, inputLatch2M, …, inputLatch16M	Normal, Tripped	Indicate the latched signal status
2	systemMode1M, systemMode2M	Normal, Tripped	Indicate the system output status
1	ChannelResetS	Reset	Reset the latched signal

Table-3

Each record has a definition of FLNK which defines a forward link. According to the EPICS manual, reference [5], A forward link allows the PV to scan passively the record in the forward link when it is processed. In each record, the forward link is linked to its subroutine. As a result, every time the record is processed its forwarded subroutine will be processed. The record type for the records is mbbo which stands for multi-bit binary output. According to the EPICS manual, reference[7], the mbbo record takes either a zero or one from the hardware address and converts it into a statement module. In this case, it can be “Normal” or “Tripped” and “Enable” or “Disable”. In addition, the mbbo record has a scan definition which processes the record every 0.01 second and passively processes its forward link subroutine every time when it is processed.

Enable Records

There are sixteen enable records. Each record has a scan definition of 0.01 seconds. There are two states in the record. When the state is one, it is displayed as “Enable”. When the state is zero, it is displayed as “Disable”. A subroutine record which provides the necessary functionalities to the mbbo record is linked to the mbbo record through a forward link. The subroutine record can not only read from the hardware address when the value is changed but can also write the value into the record. In order to achieve this, the subroutine record defines a field called SNAME which links the record to the subroutine function written in C code. In addition, the OUT field is defined to capture the scalar or array values pushed out to the output links when the subroutine record writes the value to the mbbo record. According to the EPICS manual, reference[9], the INPA field is defined to fetch the values from the record when a record tries to read from the mbbo record. Furthermore, based on the EPICS manual, reference[9], INAM field defines an initialization routine which is called once at iocInit function. Since the subroutine function accesses the hardware address through mmap operation, the ReadDataInit function in the INAM field is invoked as an initialization routine. This function is only called in one of the records.

Input Status & Latched Status Records

There is a total of 32 records for the input status and the latched status. All of them are identical except that their names and notations are different. Like Enable records, each record is a mbbo type record. That’s because the value only reads the logical level from the hardware address and it is converted into text messages that the user can understand. They are scanned every 0.01 seconds and have a forward link connected to its subroutine record. There are two states in the mbbo record. When the state is one, it is displayed as “Normal”. When the state is zero, it is displayed as “Tripped”. Importantly noted, with the current software implementation, if a channel is enabled, its status will be displayed as “Normal”. There was another version of design that changes the display to “Disable” when the channels are disabled but under current special circumstance, that version though was better, currently compromised the presentation of other features in the software. Therefore, to illustrate all the features in the system for now, that version of design was not incorporated. In each subroutine record, it has a definition of SNAME which links the subroutine record to the subroutine function written in C code. In addition, the subroutine record is required to define an output link to hold the scalars or arrays that are pushed from the subroutine function.

Reset Record

There is an additional record to reset the latched channel. It is defined as a mbbo record. The record only has one state which is “Reset” state. It has a forward link that is connected to its subroutine record. In the subroutine record, according to the EPICS manual, reference[8], SNAME is defined to connect the subroutine record to its subroutine function.

Subroutine Functions

The subroutine functions interact with their subroutine records and records with the definition of SNAME. The first initialization function uses mmap function to directly make the hardware addresses accessible from the user side. There are four hardware registers in the firmware. The register addresses are defined in the header file. They are assigned with a base address at 0x43C00000. Each register has an offset of four bits from one and another. See Table-4 for the memory map.

Register Name	Memory Address
Enable	0x43C00000
Reset	0X43C00004
Input Status	0X43C00008
Input Latched Status	0X43C0000C

Table-4

For enable register, as it is a 16-bit register, only one of the sixteen bits is assigned to an input device (there are a maximum of sixteen devices connected to the system) and each parameter only needs to read or set one of the sixteen bits. To read or set a correctly assigned bit from the parameter while reading a 16-bit register altogether from the firmware, proper bit-wise operations are required for the function. When enabled, the set operation is achieved by shifting the ‘1’ bit to the bit position that it is going to be set and then performing an “OR” operation with the original value at the address. When disabled, the clear operation is achieved by shifting the ‘1’ bit to the bit position that it is going to be set and then inverting all the bits and lastly performing an “AND” operation with the original value at the address.

Similarly, for input status and latched input status, each parameter checks its assigned bit among the 16-bit register read from the firmware. The check operation is achieved by first masking all other bits except the targeted bit and then shifting the bit to the least significant bit. If the least significant bit is one that means the device is in a normal state. If the least significant bit is zero that means the device is in a failed state.

For the reset parameter, the subroutine function first writes a ‘1’ to the assigned hardware address to assert the reset. After that it sleeps for 1000 milliseconds. Then it writes a ‘0’ to the assigned hardware address to de-assert the reset. It sleeps for another 500 milliseconds for the next reset.

CS-Studio GUI

The graphic user interface was designed to provide a better presentation for the system control. It can also help the user manage and control the system easier with the text displays and animation. The GUI page was specifically designed in a specific CS-Studio file type called OPI, where selected widgets can be dropped on the panel and connected to their PVs in the EPICS IOC software system. There are three types of widgets used in the GUI system. The text update is used to display the status of each channel. The switch buttons are used to set the parameters. The LED lights are used to alert the status of each channel visually. The detailed functionalities are demonstrated in the following. In addition, the graphic user interface is illustrated in the Figure 14-Figure 17. These figures show the possible scenarios under different user inputs.

Reset

The reset button is an action button on the top left corner of the page. The tripped input can be released by clicking the action button. The reset button was set to connect to MZED-ChannelResetA record. It is enabled, and if clicked, it will invoke the subroutine record to run the function in the EPICS IOC system.

Enable Switches

The enable switches are illustrated as flip switches. The flip switches enable the channels when they are turned on. Likewise, the flip switches disable the channels when they are turned off. All switches are initialized in “ON” mode and enabling all the input signals when the system starts up for the first time. The flip switches are enabled and connected to MZED-inputEnaS.

LED Lights

The LED lights are a visual presentation of the status of each channel. If the channel is in a “Normal” state, the LED lights are green. If the channel is in a “Tripped” state, the LED lights are red. The LED lights are enabled and connected to MZED-inputLatchM.

Raw Status

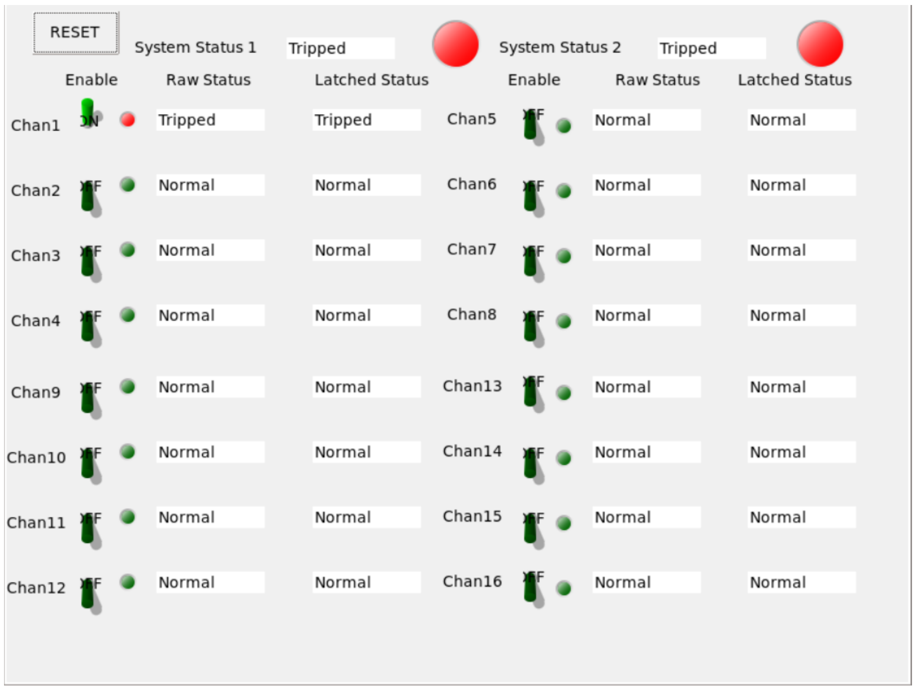
The raw status fields are text updates that report the status of each channel from the hardware address. They display either “Normal” to indicate that the channels are in a “Normal” state or “Tripped” to indicate that the channels are in a “Tripped” state. The raw status fields are enabled and connected to MZED-inputStatusM.

Latched Status

The latched status are text updates that report the latched status of each channel from the hardware address. Similar to raw status, they also display either “Normal” to indicate that the channels are in a “Normal” state or “Tripped” to indicate that the channels are in a “Tripped” state. The latched status fields are enabled and connected to MZED-inputLatchedM.

System Mode

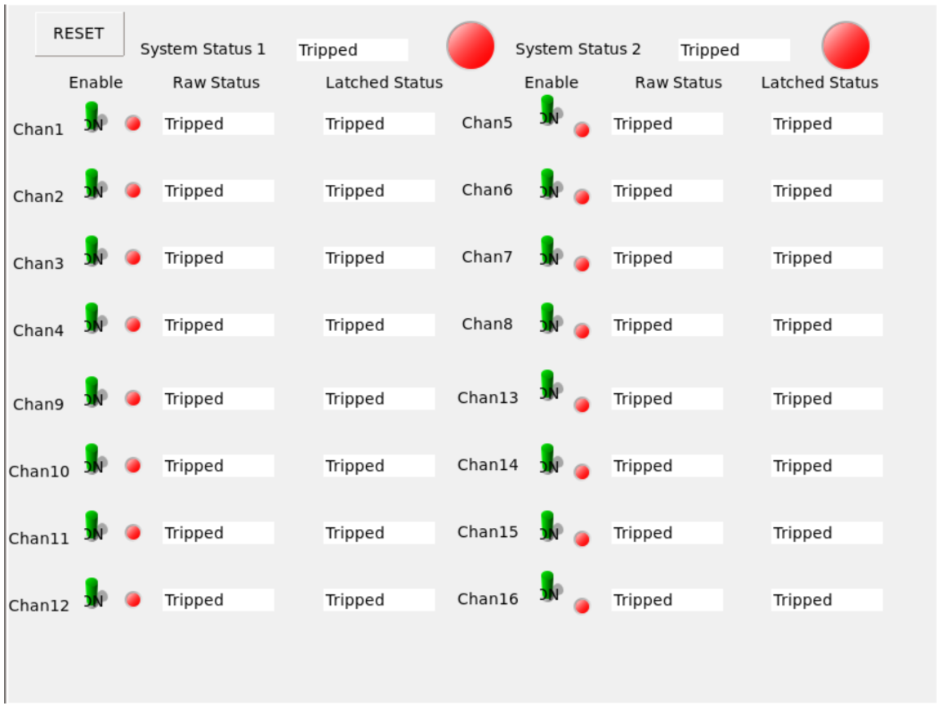
The system mode is text update that report the status of the entire system. If any of the channels is tripped, it will display “Tripped” and if everything is normal it will display “Normal”. It is enabled and connected to MZED-systemMode1M and MZED-systemMode2M. It is served as a validation check that the output status of the system is consistent with the status of individual channels.



RESET System Status 1 Tripped System Status 2 Tripped

	Enable		Raw Status	Latched Status		Enable		Raw Status	Latched Status
Chan1	ON	🔴	Tripped	Tripped	Chan5	OFF	🟢	Normal	Normal
Chan2	OFF	🟢	Normal	Normal	Chan6	OFF	🟢	Normal	Normal
Chan3	OFF	🟢	Normal	Normal	Chan7	OFF	🟢	Normal	Normal
Chan4	OFF	🟢	Normal	Normal	Chan8	OFF	🟢	Normal	Normal
Chan9	OFF	🟢	Normal	Normal	Chan13	OFF	🟢	Normal	Normal
Chan10	OFF	🟢	Normal	Normal	Chan14	OFF	🟢	Normal	Normal
Chan11	OFF	🟢	Normal	Normal	Chan15	OFF	🟢	Normal	Normal
Chan12	OFF	🟢	Normal	Normal	Chan16	OFF	🟢	Normal	Normal

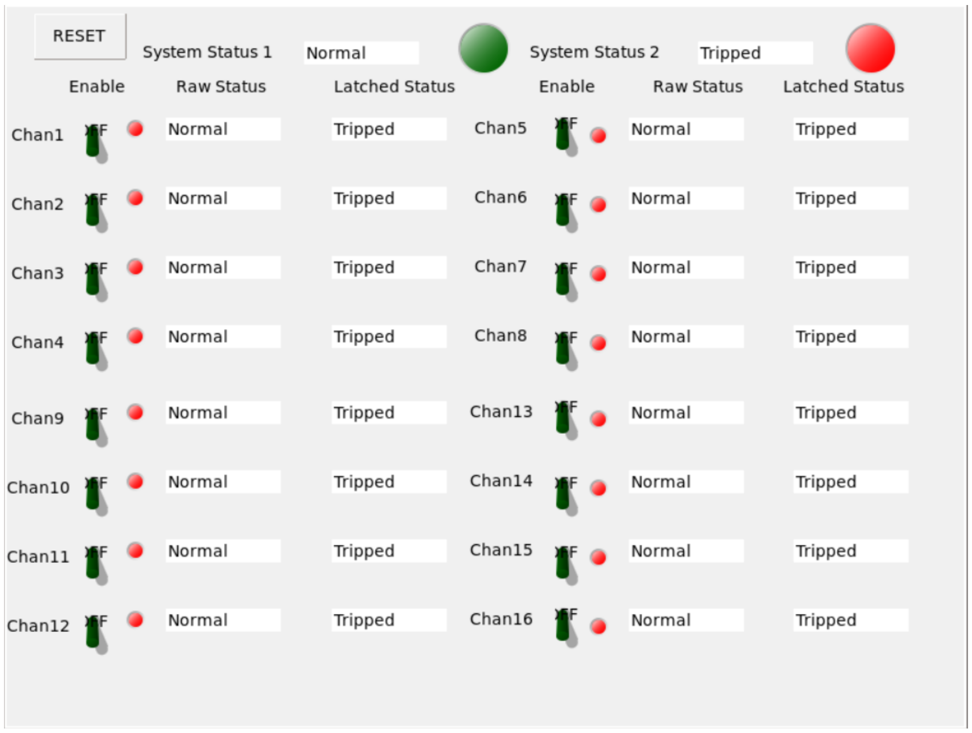
Figure-14 The figure shows when one of the devices was tripped, both outputs were tripped. When the input was tripped, the latched output was tripped immediately. Both System status 1 and 2 are changed to "Tripped"



RESET System Status 1 Tripped System Status 2 Tripped

	Enable		Raw Status	Latched Status		Enable		Raw Status	Latched Status
Chan1	ON	🔴	Tripped	Tripped	Chan5	ON	🔴	Tripped	Tripped
Chan2	ON	🔴	Tripped	Tripped	Chan6	ON	🔴	Tripped	Tripped
Chan3	ON	🔴	Tripped	Tripped	Chan7	ON	🔴	Tripped	Tripped
Chan4	ON	🔴	Tripped	Tripped	Chan8	ON	🔴	Tripped	Tripped
Chan9	ON	🔴	Tripped	Tripped	Chan13	ON	🔴	Tripped	Tripped
Chan10	ON	🔴	Tripped	Tripped	Chan14	ON	🔴	Tripped	Tripped
Chan11	ON	🔴	Tripped	Tripped	Chan15	ON	🔴	Tripped	Tripped
Chan12	ON	🔴	Tripped	Tripped	Chan16	ON	🔴	Tripped	Tripped

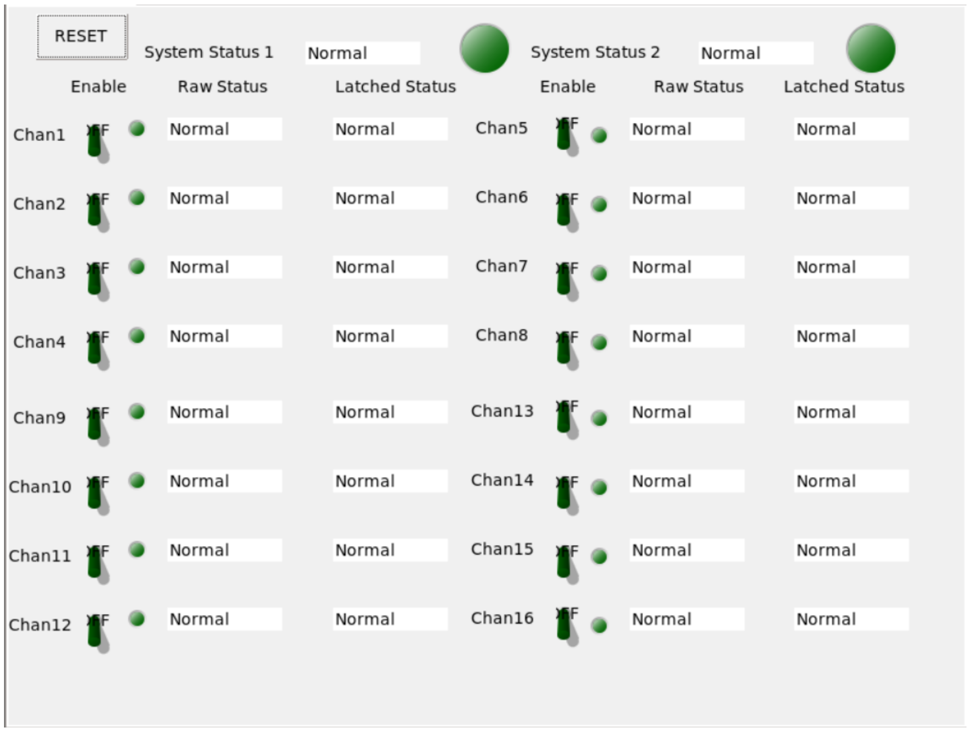
Figure-15 The figure verifies all the channels were working as expected.



RESET | System Status 1 | Normal | 🟢 | System Status 2 | Tripped | 🔴

	Enable		Raw Status	Latched Status		Enable		Raw Status	Latched Status
Chan1	F	🔴	Normal	Tripped	Chan5	F	🔴	Normal	Tripped
Chan2	F	🔴	Normal	Tripped	Chan6	F	🔴	Normal	Tripped
Chan3	F	🔴	Normal	Tripped	Chan7	F	🔴	Normal	Tripped
Chan4	F	🔴	Normal	Tripped	Chan8	F	🔴	Normal	Tripped
Chan9	F	🔴	Normal	Tripped	Chan13	F	🔴	Normal	Tripped
Chan10	F	🔴	Normal	Tripped	Chan14	F	🔴	Normal	Tripped
Chan11	F	🔴	Normal	Tripped	Chan15	F	🔴	Normal	Tripped
Chan12	F	🔴	Normal	Tripped	Chan16	F	🔴	Normal	Tripped

Figure-16 The figure shows when the raw status became normal, the latched status was tripped until the reset was pressed. The system status 2 which is based on the latched status was still tripped. All the channels were working as expected



RESET | System Status 1 | Normal | 🟢 | System Status 2 | Normal | 🟢

	Enable		Raw Status	Latched Status		Enable		Raw Status	Latched Status
Chan1	F	🟢	Normal	Normal	Chan5	F	🟢	Normal	Normal
Chan2	F	🟢	Normal	Normal	Chan6	F	🟢	Normal	Normal
Chan3	F	🟢	Normal	Normal	Chan7	F	🟢	Normal	Normal
Chan4	F	🟢	Normal	Normal	Chan8	F	🟢	Normal	Normal
Chan9	F	🟢	Normal	Normal	Chan13	F	🟢	Normal	Normal
Chan10	F	🟢	Normal	Normal	Chan14	F	🟢	Normal	Normal
Chan11	F	🟢	Normal	Normal	Chan15	F	🟢	Normal	Normal
Chan12	F	🟢	Normal	Normal	Chan16	F	🟢	Normal	Normal

Figure-17 When the reset was asserted, the latched status was released and followed the raw status to show the “Normal” state. Both system status 1 and 2 were changed to “Normal”

Reference

1. MicroZed Breakout Carrier Card Zynq System-on-Module Hardware User Guide:
http://zedboard.org/sites/default/files/documentations/5271-UG-MBCC-BKO-V1.2.pdf

2. MicroZed Schematic Rev G:
http://zedboard.org/sites/default/files/documentations/Schematic_G-04-02.zip

3. Xilinx 7 Series FPGA Libraries Guide for Schematic Designs:
https://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/7series_scm.pdf

4. EPICS Build Procedure
https://www.cadops.bnl.gov/Instrumentation/InstWiki/index.php/EPICS-V301

5. EPICS Manual and References:
https://epics.anl.gov/EpicsDocumentation/AppDevManuals/RecordRef/Recordref-6.html

6. EPICS Manual and References:
https://epics.anl.gov/EpicsDocumentation/AppDevManuals/RecordRef/Recordref-23.html

7. EPICS Manual and References:
https://epics.anl.gov/EpicsDocumentation/AppDevManuals/RecordRef/Recordref-25.html

8. EPICS Manual and References:
https://wiki-ext.aps.anl.gov/epics/index.php/RRM_3-14_Array_Subroutine

9. EPICS Manual and References:
https://epics.anl.gov/EpicsDocumentation/AppDevManuals/RecordRef/Recordref-6.html

10. MicroZed Hardware User Guide:
http://zedboard.org/sites/default/files/documentations/5276-MicroZed-HW-UG-v1-7-V1.pdf

11. MicroZed Quick Starter Guide:
http://zedboard.org/sites/default/files/documentations/QSC-Z7MB-7Z010-G-V1.pdf

12. BDF files for downloading:
https://github.com/Avnet/bdf