Paolo Baesso

AIDA Trigger logic unit (TLU)

Sunday 22nd October, 2017

Board FMC_TLU_v1E .

Paolo Baesso - October, 2017
paolo.baesso@bristol.ac.uk

Contents

Contents ii

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 FPGA . . . 2
- 1.3 Power . . . 2

2 TLU Hardware 3
- 2.1 Inputs and interfaces . . . 3
- 2.2 Clock LEMO . . . 5
- 2.3 Trigger inputs . . . 5
- 2.4 I^2C slaves . . . 6

3 Clock 9
- 3.1 Input selection . . . 9
- 3.2 Logic clocks registers . . . 10

4 DUT signals 11

5 Trigger inputs 13
- 5.1 Trigger logic . . . 13
- 5.2 Stretch and delay . . . 18
- 5.3 Event buffer . . . 20

6 Functions 23
- 6.1 Functions . . . 24

7 IPBus Registers 27

8 EUDAQ Parameters 35
- 8.1 INI file . . . 35
- 8.2 CONF file . . . 37

9 Appendix 41

Chapter 1

Introduction

This manual describes the AIDA Trigger Logic Unit (TLU) designed for the AIDA-2020 project by David Cussans[1] and Paolo Baesso[2].
The unit is designed to be used in High Energy Physics beam-tests and provides a simple and flexible interface for fast timing and triggering signals at the AIDA pixel sensor beam-telescope.
The current version of the hardware is an evolution of the EUDET-TLU and the miniTLU and is shipped in a metallic case that includes an Field Programmable Gate Array (FPGA) board and the TLU Printed Circuit Board (PCB): the FPGA is responsible for all the logic functions of the unit, while the PCB contains the clock chip, discriminator and interface blocks needed to communicate with other devices.
The current version of the PCB is FMC_TLU_v1E and is designed to plug onto a carrier FPGA board like any other FPGA Mezzanine Card (FMC) mezzanine board, although its form factor does not comply with the ANSI-VITA-57-1 standard.

1.1 Overview

The AIDA TLU provides timing and synchronization signals to test-beam readout hardware.
The hardware can provide an internally generated low-jitter 40 MHz clock or can accept and external clock reference.
It accepts the asynchronous trigger signals from up to six external sources, such as beam-scintillators, and generate synchronous signals (including global trigger or control signals) to send to up to four devices under tests. The logic function used to generate the trigger can be defined by the user among all the possible logic combinations of the inputs.
Depending on the chosen mode of operation, the TLU can accept busy signals

[1]University of Bristol, Particle Physics group
[2]University of Bristol, Particle Physics group

or other veto signals from Device Under Test (DUT)s and react accordingly, for instance avoinding any further trigger until all the busy signals have been de-asserted.
Whenever a global trigger is generated by the unit, a 48-bit time-stamp is attached to it. This time stamp is based on the 40 MHz clock. Additionally ???
The configuration parameters and data are sent and received via the IPbus. IPbus is a simple way to control and communicate TCA-based hardware via the UDP/IP protocol.

1.2 FPGA

The TLU is shipped with an FPGA board already programmed with the latest version of the firmware needed to operate the unit.
The firmware developed at University of Bristol is targeted to work with the Enclustra AX3 board, which must be plugged onto a PM3 base, also produced by Enclustra. The firmware is written on the FPGA using a Joint Test Action Group (JTAG) interface. Typically a breakout board will be required to connect the Xilinx programming cable to the Enclustra PM3. All these components are included in the TLU enclosure so the user can upload a new version of the firmware by simply connecting a Universal Serial Bus (USB)-B cable in the back panel of the unit.
At the time of writing this work[3] the AX3 is the only FPGA for which a firmware has been developed. However, we plan to ship future versions of the TLU with a custom made FPGA designed by Samer Kilani.

> **Note**
> If the FPGA detects a programming cable connected it will not load the firmware from its memory after a power cycle.
> It is recommended to leave the USB cable disconnected from the back panel unless there is the intention to re-program the firmware.

1.3 Power

The TLU requires 12 V to operate. Power can be provided using the circular jack on the back panel of the unit.
During normal operation the current drawn by the unit is about 0.5 A.

[3] Oct 2017

Chapter 2

TLU Hardware

Board FMC_TLU_v1E is an evolution of the miniTLU designed at the University of Bristol (UoB). The board shares a few features with the miniTLU but also introduces several improvements. This chapter illustrates the main features of the board to provide a general view of its capabilities and an understanding of how to operate it in order to communicate with the DUTs.

2.1 Inputs and interfaces

FMC

The board must be plugged onto a FMC carrier board with an FPGA in order to function correctly. The connection is achieved using a low pin count FMC connector. The list of the pins used and the corresponding signal within the FPGA are provided in appendix at page 41.

Device under test

The DUTs are connected to the TLU using standard size High-Definition Multimedia Interface (HDMI) connectors[1].
In the current version of the hardware, up to four DUTs can be connected to the board. In this document the connectors will be referred to as `HDMI1`, `HDMI2`, `HDMI3` and `HDMI4`.
The connectors expect 3.3 V Low Voltage Differential Signaling (LVDS) signals and are bi-directional, i.e. any differential pair can be configured to be an output (signal from the TLU to the DUT) or an input (signals from the DUT to the TLU) by using half-duplex line transceivers. Figure 2.1 illustrates how the differential pairs are connected to the transceivers.

[1] In the miniTLU hardware there were miniHDMI connectors.

Table 2.1: HDMI pin connections.

HDMI PIN	HDMI Signal Name	Enable Signal Name
1	HDMI_CLK	ENABLE_CLK_TO_DUT or ENABLE_DUT_CLK_FROM_FPGA
2	GND	–
3	HDMI_CLK *	ENABLE_CLK_TO_DUT or ENABLE_DUT_CLK_FROM_FPGA
4	CONT	ENABLE_CONT_FROM_FPGA
5	GND	–
6	CONT*	ENABLE_CONT_FROM_FPGA
7	BUSY	ENABLE_BUSY_FROM_FPGA
8	GND	–
9	BUSY*	ENABLE_BUSY_FROM_FPGA
10	SPARE	ENABLE_SPARE_FROM_FPGA
11	GND	–
12	SPARE*	ENABLE_SPARE_FROM_FPGA
13	n.c.	
14	HDMI_POWER	–
15	TRIG	ENABLE_TRIG_FROM_FPGA
16	TRIG*	ENABLE_TRIG_FROM_FPGA
17	GND	
18	n.c.	
19	n.c.	

Note

The input part of the transceiver is configured to be always on. This means that signals going *into* the TLU are always routed to the logic (FPGA). By contrast, the output transceivers have to be enabled and are off by default: signal sent from the logic to the DUTs cannot reach the devices unless the corresponding enable signal is active.

Table 2.1 shows the pin naming and the corresponding output enable signal. The clock pairs have two different enable signals to select the clock source (see section 3 for more details). In general only one of the clock sources should be active at any time.

The enable signals can be configured by programming two General Purpose Input/Output (GPIO) bus expanders via Inter-Integrated Circuit (I^2C) interface as described in section 2.4.

In terms for functionalities, the four HDMI connectors are identical with

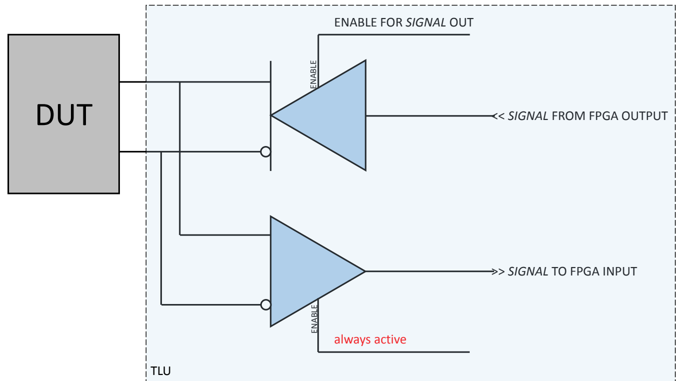


Figure 2.1: Internal configuration of the HDMI pins for the DUTs. The path from the DUT to the FPGA is always active. The path from the FPGA to the DUT can be enabled or disabled by the user.

one exception: the clock signal from `HDMI4` can be used as reference for the clock generator chip mounted on the hardware. For more details on this functionality refer to section 3.

SFP cage

FMC_TLU_v1E hosts a Small Form-factor Pluggable (SFP) cafe and a Clock and Data Recovery (CDR) chip that can be used to decode a data stream over optical/copper interface. The data from the stream is routed to the FPGA while the clock can be fed to the Si5345 to provide a clock reference.

2.2 Clock LEMO

The board hosts a two-pin LEMO connector that can be used to provide a reference clock to the clock generator (see section 3) or to output the clock from the TLU to the external world, for instance to use it as a reference for another TLU. The signal level is 3.3 V LVDS.
As for the differential pairs of the DUTs, the pins of this connector are wired to a transceiver configured to always accept the incoming signals. The outgoing direction must be enabled by using the `ENABLE_CLK_TO_LEMO` signal, which can be configured using the bus expander described in in section 2.4.

2.3 Trigger inputs

Board FMC_TLU_v1E can accept up to six trigger inputs over the LEMO connectors labelled `IN_1`, `IN_2`, `IN_3`, `IN_4`, `IN_5` and `IN_6`. The FMC_TLU_v1E

Table 2.2: DAC outputs and corresponding threshold inputs.

	Output	
	DAC2(Ic2)	**DAC1 (Ic1)**
Threshold 0	1	
Threshold 1	0	
Threshold 2		3
Threshold 3		2
Threshold 4		1
Threshold 5		0

uses internal high-speed[2] discriminators to detect a valid trigger signal. The voltage thresholds can be adjusted independently for each input in a range from -1.3 V to +1.3 V with 40 μV resolution.
The adjustment is performed by writing to two 16-bit Digital to Analog Converter (DAC)s via I^{2}C interface as described in section 2.4.
The DACs can either use an internal reference voltage of 2.5 V or an external one of 1.3 V provided by the TLU: it is recommended to choose the external one by configuring the appropriate register in the devices.
The correspondence between DAC slave and thresholds is shown in table 2.2.

2.4 I^{2}C slaves

The I^{2}C interface on the FMC_TLU_v1E can be used to configured several features of the board.
Table 2.3 lists all the valid addresses and the corresponding slave on the board. The Enclustra lines refer to slaves located on the PM3 board; these slaves can be ignored with the exception of the bus expander. The Enclustra expander is used to enable/disable the I^{2}C lines going to the FMC connector.

Note

After a power cycle the Enclustra expander is configured to disable the I^{2}C interface pins. This means that it is impossible to communicate to any I^{2}C slave on the TLU until the expander has been enabled.
The interface is enable by setting bit 7 to 0 on register 0x01 of the Enclustra expander.

Once the interface is enabled it is possible to read and write to the devices listed in the top part of table 2.3.
The user should reference the manual of each individual component to determine the register that must be addressed. The rest of this section is meant to provide an overview of the slave functionalities.

[2] 500$\pm$30 ps propagation delay.

Table 2.3: I^2C addresses of the TLU.

CHIP	ID	FUNCTION	ADDRESS
IC1	AD5665RBRUZ	DAC1	0x1F
IC2	AD5665RBRUZ	DAC2	0x13
IC5	24AA025E48T	EEPROM	0x50
IC6	PCA9539PW	I2C Expander1	0x74
IC7	PCA9539PW	I2C Expander2	0x75
IC8	ADN2814ACPZ	CDR	0x60
IC8_9	Si5345A	Clock Generator	0x68
Enclustra slaves			
		Enclustra Bus Expander	0x21
		Enclustra System Monitor	0x21
		Enclustra EEPROM	0x54
		Enclustra slave	0x64

DAC

Each DAC has four outputs that can be configured independently. `DAC1` is used to configure the thresholds of the first four trigger inputs; `DAC2` configures the remaining two thresholds.
The DACs should be configured to use the TLU voltage reference of 1.3 V. In these conditions, writing a value of 0x00000 to a DAC output will set the corresponding threshold to -1.3 V while a value of 0xFFFF will set it to +1.3 V.

EEPROM

The Electrically Erasable Programmable Read-Only Memory (EEPROM) located on the board contains a factory-set unique number, used to identify each FMC_TLU_v1E unequivocally. The number is comprised of six bytes written in as many memory locations.
The identifier is always in the form: `0xD8 80 39 XX XX XX` with the top three bytes indicating the manufacturer and the bottom three unique to each device.

Bus expander

The expanders are used as electronic switched to enable and disable individual lines. Each expander has two 8-bit banks; the values of the bits, as well as their direction (input/output) can be configured via the I^2C interface. For the purpose of the TLU, all the expander pins should be configured as outputs since they must drive the enable signals on the DUT transceivers.

Clock and data recovery chip

The CDR is used in conjunction with the SFP cage to recover data and clock from the incoming bit stream. The functionality has not yet been implemented in the firmware so the I^{2}C slave can be ignored for now.

Clock generator

The clock for FMC_TLU_v1E can be generated using various external or internal references (see section 3 for further details). In order to reduce any jitter from the clock source and to provide a stable clock, the board hosts a Si5345 clock generator that needs to be configured via I^{2}C interface.
The configuration involves writing ∼380 register values. A configuration file, containing all the register addresses and the corresponding values, can be generated using the ClockBuilder tool available from Silicon Labs.
The registers addresses between 0x026B and 0x0272 contain user-defined values that can be used to identify the configuration version: it is advisable to check those registers and check that they contain the correct code to ensure that the chip is configured according to the TLU specifications. As an indication, files generated for the current version of the TLU should have a configuration identifier in the form `TLU1E_XX`, where `XX` is a sequential number.

> **TLU Producer**
>
> When using the TLU producer to configure hardware, the location of the configuration file can be specified by setting the `CLOCK_CFG_FILE` value in the *conf* file for the producer.
> If no value is specified, the software will look for the configuration file `../conf/confClk.txt` i.e. if the `euRun` binary file is located in `./eudaq/bin`, then the default configuration file should reside in `./eudaq/conf`. The configuration will produce an error if the file is not found.

Chapter 3

Clock

The TLU can use various sources to produce a stable 40 MHz clock[1]. A Low-voltage Positive Emitter-Coupled Logic (LVPECL) crystal provides the reference 50 MHz clock for a Si5345A jitter attenuator. The Si5345A can accept up to four clock sources and use them to generate the required output clocks.
In FMC_TLU_v1E the possible sources are: differential LEMO connector LM1_9, one of the four HDMI connectors (`HDMI4`), a CDR chip connected to the SFP cage. The fourht input is used to provide a zero-delay feedback loop.
The low-jitter clock generated by the Si5345A can be distributed to up to ten recipients. In the TLU these are: the four DUTs via HDMI connectors, the differential LEMO cable, the FPGA, connector J1 as a differential pair (pins 4 and 6) and as a single ended signal (pin 8). The final output is connected to the zero-delay feedback loop.
The DUTs can receive the clock either from the Si5435A or directly from the FPGA: when provided by the clock generator, the signal name is `CLK\_TO\_DUT` and is enabled by signal `ENABLE_CLK_TO_DUT`; when the signal is provided directly from the FPGA the line used is `DUT_CLK_FROM_FPGA` and is enabled by `ENABLE_DUT_CLK_FROM_FPGA`.
The firmware uses the clock generated by the Si5345A except for the block `enclustra_ax3_pm3_infra` which relies on a crystal mounted on the Enclustra board to provide the IPBus functionalities (in this way, at power up the board can communicate via IPBus even if the Si5345A is not configured).

3.1 Input selection

The Si5345 has four inputs that can be selected to provide the clock alignment; the selection can be automatic or user-defined. For further details on this aspect the user should consult the chip documentation.

[1] For some applications a 50 MHz clock will be required instead

Table 3.1: Si5345 Input Selection Configuration.

Register Name	Hex Address [Bit Field]	Function
CLK_SWITCH_MODE	0x0536[1:0]	Selects manual or automatic switching modes. Automatic mode can be revertive or non-revertive. Selections are the following: 00 Manual 01 Automatic non-revertive 02 Automatic revertive 03 Reserved
IN_SEL_REGCTRL	0x052A [0]	0 for pin controlled clock selection 1 for register controlled clock selection
IN_SEL	0x052A [2:1]	0 for IN0 1 for IN1 2 for IN2 3 for IN3 (or FB_IN)

3.2 Logic clocks registers

LogicClocksCSR: in the new TLU the selection of the clock source is done by programming the Si5345. As a consequence, there is no reason to write to this register. Reading it back returns the status of the PLL on bit 0, so this should read 0x1.

Chapter 4

DUT signals

In the old versions of the TLU the direction of the signals on the HDMI* connectors were pre-defined. The new hardware has separate lines for signals going into the TLU and signals out of the TLU. See section 2.1 for further details.

Chapter 5

Trigger inputs

The six inputs on the TLU can be used to generate a global trigger that is then issued to all the DUTs.
Each input has a programmable voltage discriminator that can be configured in the range [-1.3 : 1.3] V.
All the inputs are protected by clamping diodes that limit the input voltage in the range [-5 : +5] V.

5.1 Trigger logic

The TLU has six trigger inputs than can be used to generate a valid trigger event. The number of possible different trigger combinations is $2^6 = 64$ so a 64-bit word can be used to decide the valid combinations. In the hardware the 64-bit word is split into two 32-bit words (indicated as Most Significant Bit (MSB) and Least Significant Bit (LSB) word) and the rules to generate the trigger can be specified by the user by writing in the two 32-bit registers `TriggerPattern_highW` and `TriggerPattern_lowW`: the first stores the 32 most significative bits of the trigger word, the latter stores the least significative bits.
The user can select any combination of the trigger inputs and declare it a valid trigger pattern by setting a 1 in the corresponding trigger configuration word. Tables 5.1 and 5.2 show an example of how to determine the trigger configuration words: whenever a valid trigger combination is encountered, the user should put a 1 in the corresponding row under the PATTERN column. The pattern thus obtained is the required word to write in the configuration register.
It is important to note that this solution allows the user to set veto pattern as well: for instance if only word 31 from table 5.1 were picked, then the TLU would only register a trigger when the combination $\overline{I_5} * I_4 * I_3 * I_2 * I_1 * I_0$ was presented at its inputs. In other words, in this specific case I_5 would act as a veto signal and the TLU would **not** produce a global trigger if I_5=1.
The default configuration in the firmware is Hi= 0xFFFFFFFF, Low= 0xFF-

Table 5.1: Example of configuration word for the least significative bits of the trigger registers: the only valid configuration is represented by $\overline{I_5} + I_4 + I_3 + I_2 + I_1 + I_0$, i.e. a trigger is accepted if all the inputs, except I_5, present a logic 1 at the same time. The user would then write the resulting word 0x80000000 in the TriggerPattern_lowW register.

DEC	I5	I4	I3	I2	I1	I0	PATTERN	CONFIG. WORD		2^n
0	0	0	0	0	0	0	0	0	LOWEST 32-bits	1
1	0	0	0	0	0	1	0			2
2	0	0	0	0	1	0	0			4
3	0	0	0	0	1	1	0			8
4	0	0	0	1	0	0	0	0		16
5	0	0	0	1	0	1	0			32
6	0	0	0	1	1	0	0			64
7	0	0	0	1	1	1	0			128
8	0	0	1	0	0	0	0	0		256
9	0	0	1	0	0	1	0			512
10	0	0	1	0	1	0	0			1024
11	0	0	1	0	1	1	0			2048
12	0	0	1	1	0	0	0	0		4096
13	0	0	1	1	0	1	0			8192
14	0	0	1	1	1	0	0			16384
15	0	0	1	1	1	1	0			32768
16	0	1	0	0	0	0	0	0		65536
17	0	1	0	0	0	1	0			131072
18	0	1	0	0	1	0	0			262144
19	0	1	0	0	1	1	0			524288
20	0	1	0	1	0	0	0	0		1048576
21	0	1	0	1	0	1	0			2097152
22	0	1	0	1	1	0	0			4194304
23	0	1	0	1	1	1	0			8388608
24	0	1	1	0	0	0	0	0		16777216
25	0	1	1	0	0	1	0			33554432
26	0	1	1	0	1	0	0			67108864
27	0	1	1	0	1	1	0			134217728
28	0	1	1	1	0	0	0	8		268435456
29	0	1	1	1	0	1	0			536870912
30	0	1	1	1	1	0	0			1073741824
31	0	1	1	1	1	1	1			2147483648

Table 5.2: Example of the most significative word of the register: a valid trigger is obtained when the inputs show the same configuration as row DEC 36, 37, 38, 39, 41, 43 and 63. These configuration are in logic OR with that presented in table 5.1. The resulting configuration word is 0x80000AF0.

DEC	I5	I4	I3	I2	I1	I0	PATTERN	CONFIG. WORD		2^n
32	1	0	0	0	0	0	0	0	HIGHEST 32-bits	1
33	1	0	0	0	0	1	0			2
34	1	0	0	0	1	0	0			4
35	1	0	0	0	1	1	0			8
36	1	0	0	1	0	0	1	F		16
37	1	0	0	1	0	1	1			32
38	1	0	0	1	1	0	1			64
39	1	0	0	1	1	1	1			128
40	1	0	1	0	0	0	0	A		256
41	1	0	1	0	0	1	1			512
42	1	0	1	0	1	0	0			1024
43	1	0	1	0	1	1	1			2048
44	1	0	1	1	0	0	0	0		4096
45	1	0	1	1	0	1	0			8192
46	1	0	1	1	1	0	0			16384
47	1	0	1	1	1	1	0			32768
48	1	1	0	0	0	0	0	0		65536
49	1	1	0	0	0	1	0			131072
50	1	1	0	0	1	0	0			262144
51	1	1	0	0	1	1	0			524288
52	1	1	0	1	0	0	0	0		1048576
53	1	1	0	1	0	1	0			2097152
54	1	1	0	1	1	0	0			4194304
55	1	1	0	1	1	1	0			8388608
56	1	1	1	0	0	0	0	0		16777216
57	1	1	1	0	0	1	0			33554432
58	1	1	1	0	1	0	0			67108864
59	1	1	1	0	1	1	0			134217728
60	1	1	1	1	0	0	0	8		268435456
61	1	1	1	1	0	1	0			536870912
62	1	1	1	1	1	0	0			1073741824
63	1	1	1	1	1	1	1			2147483648

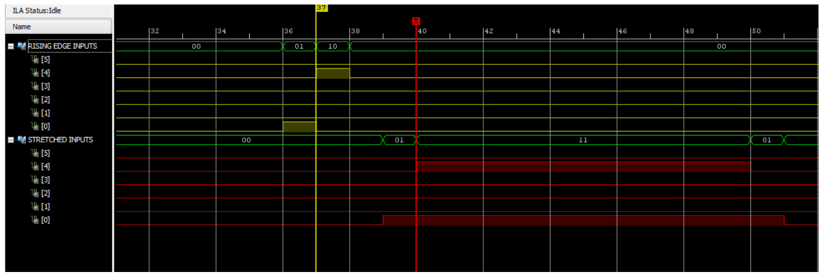


Figure 5.1: Input pulses (yellow) and corresponding stretched signals (red). Input 0 is stretched by 10 cycles, input 4 by 8, hence the difference in pulse widths.

FEFFFE, which means that as long as any trigger input fires, a trigger will be generated. These words are loaded in the FPGA every time the firmware is flushed.

Trigger logic definition

The user should pay attention to what trigger logic they want to define in order to avoid confusion in the data.
A “1” in the logic table means that the corresponding input must be active to produce a valid trigger. Similarly, a “0” indicates that the corresponding input must be inactive (i.e. is a veto, not an ignore). Any change in input configuration will cause the logic to re-assess the trigger status. The following section gives a brief example.

Bit 0 meaning

A 1 in the lowest bit of the LSB word indicates that $\overline{I_5} * \overline{I_4} * \overline{I_3} * \overline{I_2} * \overline{I_1} * \overline{I_0}$ is a valid trigger combination, so the TLU will produce a trigger when all the inputs are unactive (i.e. even if all the inputs are unplugged). Apart from very specific cases, this is generally not a desired behaviour.

Example

In this example we have connected a pulser to two inputs of the TLU, namely `IN_1` and `IN_5`. The inputs fire with a small, random delay with respect to each other.
In order to ensure that the signals overlap adequately, we use the *stretch* register (see chapter 5.1) to increase the length of the pulses: we extend *in0* to 10 clock cycles and *in4* to 8 clock cycles, where the clock has a frequency of 160 MHz. The resulting signals are shown in figure 5.1.
We can now define the trigger logic to be used to assert a valid trigger: we

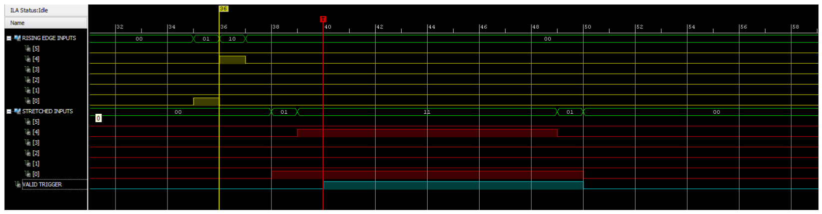


Figure 5.2: Trigger configuration 0x00020000. The valid trigger (blue) is asserted only when both signals are high. This condition occurs at frame 39. The trigger is asserted on the following frame.

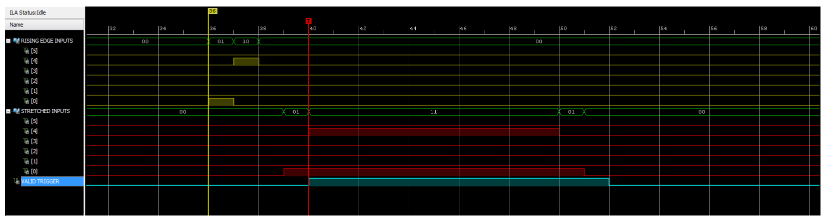


Figure 5.3: Trigger configuration 0x00020002. The valid trigger (blue) is asserted if `IN_1` is high OR when `IN_1` and `IN_5` are both high at the same time.

only consider the lower 32-bits of the trigger word and see how different values can produce very different results.

- Trigger LSB word= 0x00020000. This indicates that the only valid trigger combination occurs when both `IN_1` and `IN_5` are high. The valid trigger goes high 1 clock cycle after this condition is met and remains high up to 1 clock cycle after the condition is no longer valid. This is illustrated in figure 5.2.

- Trigger LSB word= 0x00020002. This indicates that a valid trigger is achieved in two separated configurations (in logic OR): when both inputs are high at the same time (as in the previous case) or if `IN_1` is active on its own. This is illustrated in figure 5.3. It can be seen that the valid trigger is asserted immediately one clock cycle after `IN_1` is high and remains high as long as this condition is met. One might assume that specifying the combination with `IN_5` is redundant, but the following example should show that this is not the case.

- Trigger LSB word= 0x00000002. This indicates that the only valid configuration is the one where only `IN_1` is high. It is important to understand that in this configuration all other inputs act as veto. This might produce unexpected results if the user is not careful[1].

[1] Specifically, pulse stretch, pulse delay and trigger logic must be configured correctly to avoid unwanted results.

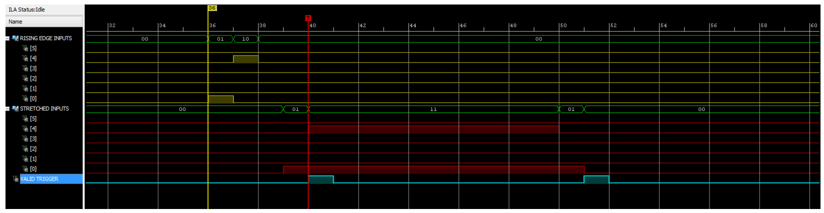


Figure 5.4: Trigger configuration 0x00000002. The valid trigger (blue) is asserted only when `IN_1` is active on its own. As such, two separated trigger pulses are produced because `IN_5` goes high and returns low before `IN_1`.

In figure 5.4 it is possible to see that the logic produces two separated trigger valid pulses, both shorter than the ones in previous examples: the first one is due to `IN_1` going high while `IN_5` is low. As soon as `IN_5` goes high, the trigger condition is no longer met. When `IN_5` returns low, a trigger condition is met again because `IN_1` is still high. In this specific case, the double pulse is caused by the different width of the pulses.

5.2 Stretch and delay

The trigger logic is designed to detect edge transitions[2] at the trigger inputs and produce a pulse for each transition detected. The pulse has an initial duration of one clock cycle (f= 160 MHz, one cycle 6.25 ns) and occurs on the next rising edge of the 160 MHz internal clock.
Each pulse can be stretched and delayed in integer numbers of clock cycles to compensate for differences in cable length. Two separate 5-bit registers are used for the task: the value written in the registers will stretch/delay the pulse by a corresponding number of clock cycles.
Diagram 5.5 shows the effect of the delay and stretch words on the trigger logic.
Further details on how to configure the stretch and delay values are provided in section 8.

[2] Currently only negative edges are registered. A future firmware version will implement user-selectable positive or negative edge detection.

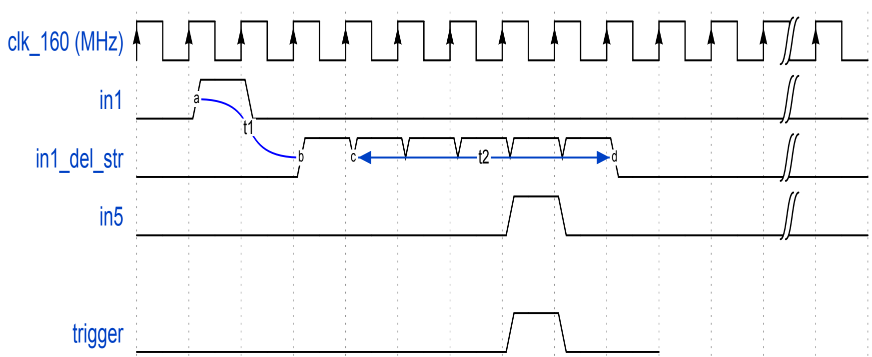


Figure 5.5: Effect of the stretch and delay values. In1 is delayed by 2 clock cycles (t1= 12.5 ns) and stretched by 5 clock cycles (t2= 31.25 ns) to create a coincidence window with in5 and produce the resulting trigger signal.

5.3 Event buffer

The event buffer IPBus slave has four registers. Writing to `EventFifoCSR` will reset the First In First Out (FIFO). Reading from either of the register will put their data on the IPBus data line.
Reading from `EventFifoCSR` returns the following:

- bit 0: FIFO empty flag
- bit 1: FIFO almost empty flag
- bit 2: FIFO almost full flag
- bit 3: FIFO full flag
- bit 4: FIFO programmable full flag
- other bits: 0

The status register (SerdesRst) is as follows:

- bit 0: reset the ISERDES
- bit 1: reset the trigger counters
- bit 2: calibrate IDELAY: This seems to be disconnected at the moment.
- bit 3: fixed to 0
- bit 4, 5: status of `thresholdDeserializer(Input0)`. When the IDELAY modules (prompt, delayed) have reached the correct delay, these two bits should read 00.
- bit 6, 7: status of `thresholdDeserializer(Input1)`
- bit 8, 9: status of `thresholdDeserializer(Input2)`
- bit 10, 11: status of `thresholdDeserializer(Input3)`
- bit 12, 13: status of `thresholdDeserializer(Input4)`
- bit 14, 15: status of `thresholdDeserializer(Input5)`
- bit 16, 19: fixed to 0
- bit 20: `s_deserialized_threshold_data(Input0)(7)`
- bit 21: `s_deserialized_threshold_data(Input1)(7)`
- bit 22: `s_deserialized_threshold_data(Input2)(7)`
- bit 23: `s_deserialized_threshold_data(Input3)(7)`
- bit 24: `s_deserialized_threshold_data(Input4)(7)`

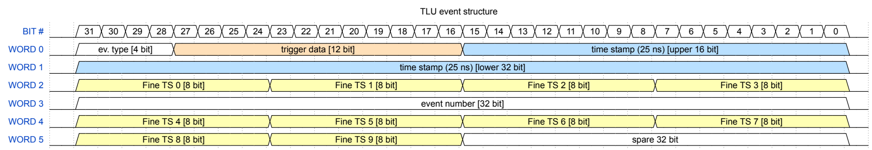


Figure 5.6: Event structure

- bit 25: `s_deserialized_threshold_data(Input5)(7)`

9 bits are used to determine trigger edges. 8 are from the deserializers, 1 is added as the LSB and is the MSB from the previous word.

Chapter 6

Functions

The following is a list of files containing the code for the TLU:

- `./eudaq2/user/eudet/misc/fmctlu_runcontrol.ini`:
  initialization file for the hardware. The location of the file can be passed to the EUDAQ code in the Graphic User Interface (GUI).

- `./eudaq2/user/eudet/misc/fmctlu_runcontrol.conf`:
  configuration file. It contains all the parameters to be loaded in the TLU at the beginning of the run. If this file is not found, EUDAQ will use a list of default settings. The location of the file (and its name) can be passed to the EUDAQ code in the GUI.

- `./eudaq2/user/eudet/misc/fmctlu_connection.xml`:
  define the IP address and address map of the TLU. The one listed is the default location for the file. A different location can be specified with the `ConnectionFile` option in the *conf* file for the TLU.

- `./eudaq2/user/eudet/misc/fmctlu_address.xml`:
  address map for the TLU. The location of the file is specified in the `fmctlu_connection.xml` file.

- `./eudaq2/user/eudet/misc/fmctlu_clock_config.txt`:
  configuration for the Si5345 clock chip. In order for the hardware to work a configuration file must be present. Those listed are the default name and location for the file; a different file can be specified with the `CLOCK_CFG_FILE` option in the *conf* file for the TLU.

- `./eudaq2/user/eudet/module/src/FMCTLU_Producer.cc`:
  eudaq producer for the TLU. Contains the methods to initialize, configure, start, stop the TLU producer.

- `./eudaq2/user/eudet/hardware/src/FmctluController.cc`:
  Contains the definition of the hardware class for the TLU and the methods to set and read from its hardware, such as clock chip, DAC, etc. This

lever is abstract with respect to the actual hardware, so that if a future version of the board uses different components it should be possible to re-use this code.

- `./eudaq2/user/eudet/hardware/include/FmctluController.hh`:
  Headers for the controller.
- `./eudaq2/user/eudet/hardware/src/FmctluController.cxx`:
  Executable for the controller.
- `./eudaq2/user/eudet/hardware/src/FmctluHardware.cc`:
  This is the code that deals with the actual hardware on the TLU, and contains specific instructions for the chips mounted in the current version. It contains several classes for the ADC, the clock chip, the I/O expanders etc.
- `./eudaq2/user/eudet/hardware/include/FmctluHardware.hh`:
  Header for the hardware.
- `./eudaq2/user/eudet/hardware/src/FmctluI2c.cc`:
  core functions used to read and write from I^2C compatible slaves.
- `./eudaq2/user/eudet/hardware/include/FmctluI2c.hh`:
  Headers for the I^2C core.

6.1 Functions

enableClkLEMO Enable or disable the output clock to the differential LEMO connector.

enableHDMI Set the status of the transceivers for a specific HDMI connector. When enable= False the transceivers are disabled and the connector cannot send signals from FPGA to the outside world. When enable= True then signals from the FPGA will be sent out to the HDMI.
In the configuration file use `HDMIx_on = 0` to disable a channel and `HDMI1_on = 1` to enable it (x can be 1, 2, 3, 4).
NOTE: the other direction is always enabled, i.e. signals from the DUTs are always sent to the FPGA.
NOTE: Clock source must be defined separately using SetDutClkSrc (DUTClkSrc in python script).
NOTE: this is called `DUTOutputs` on the python scripts.

GetFW dsds

getSN dsd

I2C_enable dsd

InitializeClkChip

InitializeDAC

InitializeIOexp

InitializeI2C

PopFrontEvent

ReadRRegister

ReceiveEvents

ResetEventsBuffer

SetDutClkSrc Set the clock source for a specific HDMI connector. The source can be set to 0 (no clock), 1 (Si5345) or 2 (FPGA). In the configuration file use `HDMIx_on = N` to select the source (x can be 1, 2, 3, 4, N is the clock source).
NOTE: this is called `DUTClkSrc` on python scripts.

SetPulseStretchPk Takes a vector of six numbers, packs them (5-bits each) and sends them to the PulseStretch register.

SetThresholdValue

setTrgPattern Writes two 32-bit words to define the trigger pattern for the inputs. See section 5 for details.

SetWRegister

SetUhalLogLevel

Chapter 7

IPBus Registers

version Returns the current version of firmware used to program the TLU

DUTINTERFACES

DUTMaskW Writing to this register allows to define which DUTs are active when in AIDA mode. The lower 4 bits of the register can be used to define the status of the DUTs: 1 for active, 0 for masked. `hdmi1` is defined by bit 0, `hdmi2` is defined by bit 1, `hdmi3` is defined by bit 2, `hdmi4` is defined by bit 3.

IgnoreDUTBusyW Writing to this register allows to ignore the busy signal from a particular DUT while in AIDA mode. The lower 4 bits are used to define the status for each device. A 1 indicates that the logic should ignore busy signals from the specific DUT.

IgnoreShutterVetoW The LSB of this register can be written to define whether the DUT should ignore the shutter veto signal. Normally, when the shutter signal is asserted the DUT reports busy. If this bit is flag the DUT will ignore the shutter signal.

DUTInterfaceModeW Write register to define the mode of operation for a DUT. Two bits per device can be used to define the mode; currently only two modes are available (AIDA, EUDET) but the second but is reserved for additional modes introduced in the future.
The bit pairs are packed from the LSB starting with `hdmi1` (bits 0, 1), `hdmi2` (bits 2, 3), `hdmi3` (bits 4, 5), `hdmi4` (bits 6, 7).

- bit pair `X0`: EUDET
- bit pair `X1`: AIDA

DUTInterfaceModeModifierW Write register. This register only affects the EUDET mode of operation. For each DUT two bits can be configured although cyrrently only the lower of the pair is considere. The bit packing

Table 7.1: IPBus register

NODE	SUBNODE	ADDRESS	MASK	PERMISSION
version		0x1		r
DUTInterfaces		**0x1000**		
	DUTMaskW	0x0		w
	IgnoreDUTBusyW	0x1		w
	IgnoreShutterVetoW	0x2		w
	DUTInterfaceModeW	0x3		w
	DUTInterfaceModeModifierW	0x4		w
	DUTInterfaceModeR	0xB		r
	DUTInterfaceModeModifierR	0xC		r
	DUTMaskR	0x8		r
	IgnoreDUTBusyR	0x9		r
	IgnoreShutterVetoR	0xA		r
Shutter		**0x2000**		
	ShutterStateW	0x0		w
	PulseT0	0x1		w
i2c_master		**0x3000**		
	i2c_pre_lo	0x0	0xFF	r/w
	i2c_pre_hi	0x1	0xFF	r/w
	i2c_ctrl	0x2	0xFF	r/w
	i2c_rxtx	0x3	0xFF	r/w
	i2c_cmdstatus	0x4	0xFF	r/w
eventBuffer		**0x4000**		
	EventFifoData	0x0		r
	EventFifoFillLevel	0x1		r
	EventFifoCSR	0x2		r/w
	EventFifoFillLevelFlags	0x3		r
Event_Formatter		**0x5000**		
	Enable_Record_Data	0x0		r/w
	ResetTimestampW	0x1		w
	CurrentTimestampLR	0x2		r
	CurrentTimestampHR	0x3		r
triggerInputs		**0x6000**		
	SerdesRstW	0x0		w
	SerdesRstR	0x8		r
	ThrCount0R	0x9		r
	ThrCount1R	0xA		r
	ThrCount2R	0xB		r
	ThrCount3R	0xC		r
	ThrCount4R	0xD		r
	ThrCount5R	0xE		r
triggerLogic		**0x7000**		
	PostVetoTriggersR	0x10		r
	PreVetoTriggersR	0x11		r
	InternalTriggerIntervalW	0x02		w
	InternalTriggerIntervalR	0x12		r
	TriggerVetoW	0x04		w
	TriggerVetoR	0x14		r
	ExternalTriggerVetoR	0x15		r
	PulseStretchW	0x06		w
	PulseStretchR	0x16		r
	PulseDelayW	0x07		w
	PulseDelayR	0x17		r
	TriggerHoldOffW	0x08		w
	TriggerHoldOffR	0x18		r
	AuxTriggerCountR	0x19		r
	TriggerPattern_lowW	0x0A		w
	TriggerPattern_lowR	0x1A		r
	TriggerPattern_highW	0x0B		w
	TriggerPattern_highR	0x1B		r
logic_clocks		**0x8000**		
	LogicClocksCSR	0x0		r/w
	LogicRst	0x1		w

is done in a manner similar to the DUTInterfaceMode. Set bit high to allow asynchronous veto using `DUT_CLK` when in EUDET mode.

DUTInterfaceModeR Read the content of the DUTInterfaceMode register.

DUTInterfaceModeModifierR Read status of the DUTInterfaceMode register.

DUTMaskR Read the status of the DUTMask register.

IgnoreDUTBusyR Read the status of the IgnoreDUTBusy register.

IgnoreShutterVetoR Read the status of the IgnoreShutterVeto word (only the last bit is meanigful).

SHUTTER

ShutterStateW The LSB of this register is propagated to the DUTs as shutter signal. This is the signal that the DUTs receive on the `cont` line.

PulseT0 Writing to this register will cause the firmware to generate a T0 signal.

I2C_MASTER This section includes registers used to talk to the I^2C bus.

i2c_pre_lo Lower part of the clock pre-scaler value. The pre-scaler is used to reduce the clock frequency of the bus and make it compatible with the I^2C slaves on the board.

i2c_pre_hi Higher part of the clock pre-scaler value.

i2c_ctrl

i2c_rxtx

i2c_cmdstatus

EVENTBUFFER

EventFifoData Returns the content of the FIFO. In the current firmware implementation the memory can hold 8192 words (32-bit).

EventFifoFillLevel Read register. Returns the number of words written in the FIFO. The lowest 14-bits are the actual data.

EventFifoCSR Read or write register. When read it returns the status of the FIFO. Five flags are returned:

- bit 0: empty. Asserted when the FIFO is empty.
- bit 1: almost empty. Asserted when one word remains in the FIFO.
- bit 2: almost full. Asserted when the FIFO can only accept one more word before becoming full.
- bit 3: full. In the current firmware the FIFO can hold 8192 words before filling up.
- bit 4: programmable full. This signal is asserted when the number of words in the FIFO is greater than or equal to the assert threshold (8181). It is de-asserted when the number of words in the FIFO is less than the negate threshold (8180).

When any value is written to this register the FIFO is reset.

EventFifoFillLevelFlags Does not do anything? REMOVE CHECK

EVENT_FORMATTER

Enable_Record_Data Read and write register. When written, CHECK When read returns the content of the enable record word.

ResetTimestampW Write register. Writing any value to this register will cause the firmware to produce a retest timestamp signal (high for one clock cycle of clk_4x_logic). At the moment it does not seems to be connected to anything. CHECK

CurrentTimestampLR CHECK

CurrentTimestampHR CHECK

TRIGGERINPUTS

SerdesRstW Write register for the SerDes control.

- bit 0: set this bit to reset the ISERDES
- bit 1: set this bit to reset the input trigger counters
- bit 2: `s_calibrate_delay`

SerdesRstR Read register for the SerDes control.

ThrCount0R Read register. Returns the number of pulses above threshold for the trigger input.

ThrCount1R Read register. Returns the number of pulses above threshold for the trigger input.

ThrCount2R Read register. Returns the number of pulses above threshold for the trigger input.

ThrCount3R Read register. Returns the number of pulses above threshold for the trigger input.

ThrCount4R Read register. Returns the number of pulses above threshold for the trigger input.

ThrCount5R Read register. Returns the number of pulses above threshold for the trigger input.

TRIGGERLOGIC

PostVetoTriggersR Read register. Returns the number of triggers recorded in the TLU after the veto is applied. These are the triggers actually sent to the DUTs.

PreVetoTriggersR Read register. Returns the number of triggers recorded in the TLU before the veto is applied. This is used for debugging purposes.

InternalTriggerIntervalW Write the number of clock cycles to be used as period for the internal trigger generator. If this number is smaller than 5 then the triggers are disabled. Otherwise the period is number -2.

InternalTriggerIntervalR Read the value written in InternalTriggerIntervalW.

TriggerVetoW Write register. The value written to the LSB of this register is used to generate a veto signal. This can be used to put switch the TLU status: if the bit is asserted the logic will not send new triggers to the DUTs. If the bit is reset the board will process new triggers.

TriggerVetoR Read the content of the TriggerVeto register.

ExternalTriggerVetoR Read register. Bit 0 of this register reports the `veto` status (1 for veto active, 0 for no veto). The veto is active if the TLU buffer is full or if one of the DUTs is sending a veto signal.

PulseStretchW Write the stretch word for the trigger pulses. The original trigger pulses collected at a trigger input can be stretched by N cycles of the 4x clock (160 MHz, 6.25 ns). N is a number between 0 and 31. The stretched pulse is always at least as long as the original input.
The stretch values can be written in the `conf` file using the parameters `inX_STR` (X= [0 ... 5]).
The six words for the inputs are packed in a single 32-bit word written to this register according to the format shown in table 7.2.

PulseStretchR Returns the content of the PulseStretch word.

PulseDelayW Write the delay word for the trigger pulses. The original pulse is delayed by N clycles of the 4x clock (160 MHz, 6.25 ns). N is a number between 0 and 31. The six words for the inputs are packed in a single 32-bit word written to this register according to the format shown in table 7.2.
The delay values can be written in the `conf` file using the parameters `inX_DEL` (X= [0 ... 5]).

PulseDelayR Returns the content of the PulseDelay word.

TriggerHoldOffW Does not do anything? CHECK

TriggerHoldOffR Read the previous register... CHECK

AuxTriggerCountR Auxiliary trigger counter. Used for debug.

TriggerPattern_lowW Write register for the lower 32-bits of the trigger pattern. This pattern is used to select the combinations of trigger signals that produce a valid trigger in the TLU. See section 5.1 for details.

TriggerPattern_lowR Read register for the lower 32-bits of the trigger pattern. This pattern is used to select the combinations of trigger signals that produce a valid trigger in the TLU. See section 5.1 for details.

TriggerPattern_highW Write register for the higher 32-bits of the trigger pattern. This pattern is used to select the combinations of trigger signals that produce a valid trigger in the TLU. See section 5.1 for details.

TriggerPattern_highR Read register for the higher 32-bits of the trigger pattern. This pattern is used to select the combinations of trigger signals that produce a valid trigger in the TLU. See section 5.1 for details.

LOGIC_CLOCKS

LogicClocksCSR This is a read/write register. The write function is now obsolete and should be removed. Reading from this register returns the status of the PLL lock: bit 0 is the locked value of the pll (1= locked).

LogicRst Writing a 1 in the LSB of this register will reset the PLL and the clocks used by the TLU firmware. It needs to be checked for bugs.

Table 7.2: Packing scheme for values in registers used to define the pulse stretch and delay.

Register value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	Input 5					Input 4					Input 3					Input 2					Input 1					Input 0				
x	x	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0

Chapter 8

EUDAQ Parameters

List of parameters that are parsed by the EUDAQ run control GUI to configure the TLU.
The parameters must be included in the INI or CONF file passed to the main window (see fig.8.1).
Not all parameters are needed; if one of the parameters is not present in the files, the code will generally assume a default value, indicated in brackets in the following document `[type, default]`.

8.1 INI file

initid `[string, "0"]` Does not serve any purpose in the code but can be useful to identify configuration settings used in a specific run. EUDAQ will store this information in the run data.

ConnectionFile `[string, "file://./FMCTLU_connections.xml"]` Name of

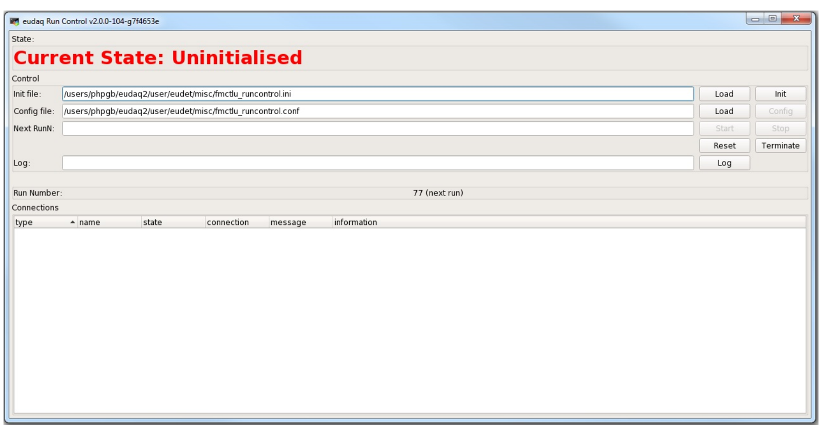


Figure 8.1: Main user iterface of the EUDAQ framework.

the xml file used to store the information required to communicate with the hardware, such as its IP address and the location of the address map. The default location indicates a file that must be located in the `bin` folder.

DeviceName `[string, "fmctlu.udp"]` The name of the type of hardware to be contacted by the IPBus.

TLUmod `[string, "1e"]` Version of the TLU hardware. Reserved for future use.

nDUTs `[positive int, 4]` Number of DUT in the current TLU. This is for future upgrades and should not require editing by the user.

nTrgIn `[positive int, 6]` Number of trigger inputs in the current TLU. This is for future upgrades and should not require editing by the user.

I2C_COREEXP_Addr `[positive int, 0x21]` I^2C address of the core expander mounted on the Enclustra board. This is not required if a different FPGA is used. See section 2.4 for further details.

I2C_CLK_Addr `[positive int, 0x68]` I^2C address of Si5345 clock generator installed on the TLU.

I2C_DAC1_Addr `[positive int, 0x13]` I^2C address of DAC installed on the TLU. The DAC is used to configure the threshold of the trigger inputs.

I2C_DAC2_Addr `[positive int, 0x1F]` I^2C address of DAC installed on the TLU. The DAC is used to configure the threshold of the trigger inputs.

I2C_ID_Addr `[positive int, 0x50]` I^2C address the unique ID EEPROM installed on the TLU. The chip is used to provide a unique identifier to each kit.

I2C_EXP1_Addr `[positive int, 0x74]` I^2C address the bus expander used to select the direction of the HDMI pins on the board.

I2C_EXP2_Addr `[positive int, 0x75]` I^2C address the bus expander used to select the direction of the HDMI pins on the board.

intRefOn `[boolean, false]` If true, the DACs installed on the TLU will use their internal voltage reference rather than the one provide externally.

VRefInt `[float, 2.5]` Value in volts for the internal reference voltage of the DACs. The voltage is chosen by the chip manufacturer. This is only used if `intRefOn= true`.

VRefExt `[float, 1.3]` Value in volts for the external reference voltage of the DACs. The voltage is determined by a circuit on the TLU and the value of this parameter must reflect such voltage. This is only used if `intRefOn= false`.

CONFCLOCK `[boolean, true]` If true, the clock chip Si5345 will be reconfigured when the INIT button is pressed (see figure fig.8.1). The chip is configured via I^2C interface using a specific text file (see next parameter). After a power cycle, the chip is not configured and must be reconfigured to operate the TLU correctly.

CLOCK_CFG_FILE `[string, "./../user/eudet/misc/fmctlu_clock_config.txt"]` Name of the text file used to store the configuration values of the Si5345. The file can be generate using the Clockbuilder Pro software provided by SiLabs.

8.2 CONF file

confid `[string, "0"]` Does not serve any purpose in the code but can be useful to identify configuration settings used in a specific run. EUDAQ will store this information in the run data.

verbose `[int, 0]` Defines the level of output messages from the TLU. 0 indicates minimum output.

HDMI1_set `[unsigned int, 0b0001]` Defines the source of the signal on the pins for the `HDMI1` connector. A 1 indicates that each pin pair is an driven by the TLU, a 0 that they are left floating (with respect to the TLU). This can be used to define the signal direction on each pin pair. The order of the pairs is as follow:
bit 0= CONT, bit 1= SPARE, bit 2= TRIG, trig 3= BUSY. Note that the direction of the DUTClk pair is defined in a separate parameter.
Example to configure the connector to work with an EUDET device:
- in this configuration the BUSY line is driven by the device under test, so it is an input for the TLU and should not be driven by it (bit 3= 0)
- TRIGGER line is an output for the TLU so is driven by it (bit 2= 1)
- SPARE line is used to provide control signals, such as the reset signal to initialize the devices at the start of a run ($\mathtt{T}_0$). It should be configured as driven by the TLU (bit 1= 1)
- CONT is used by the TLU to issue control commands and should be configured as a signal driven by the TLU (bit 0= 1).
Therefore the value of this parameter would be 0x7 (b1110).

HDMI2_set `[unsigned int, 0b0001]` Defines the direction of the pins for the `HDMI2` connector.

HDMI3_set `[unsigned int, 0b0001]` Defines the direction of the pins for the `HDMI3` connector.

HDMI4_set `[unsigned int, 0b0001]` Defines the direction of the pins for the `HDMI4` connector.

HDMI1_clk `[unsigned int, 1]` Defines if the DUTClk pair on the HDMI connector must be driven by the TLU and, if so, what clock source to use. A 0 indicates that the pins are not driven by the TLU. 1 indicates that pins will by driven with the clock produced from the on-board clock chip Si5345. 2 indicates that the driving clock is obtained from the FPGA. Example to configure the connector to work with an EUDET device: in this scenario the clock is driven by the DUT so the parameter should be set to 0. Example to configure the connector to work with an AIDA device: in this scenario the clock is driven by the TLU so the parameter should be set to either 1 or 2 (by default 1).

HDMI2_clk `[unsigned int, 1]` Defines the driving signal on the corresponding HDMI connector.

HDMI3_clk `[unsigned int, 1]` Defines the driving signal on the corresponding HDMI connector.

HDMI4_clk `[unsigned int, 1]` Defines the driving signal on the corresponding HDMI connector.

LEMOclk `[boolean, true]` Defines whether a driving clock is to be provided on the differential LEMO connector of the TLU. By default (value= 1), the clock is driven from the clock chip. If the value is set to 0 no clock will be driven.

in0_STR `[unsigned int, 0]` Defines the number of clock cycles used to stretch a pulse once a trigger is detected by the discriminator on input 0. This feature allows the user to modify the pulses that are then fed into the trigger logic within the TLU. A minimum lenght of 6.25 ns is provided if the value is 0. Any extra clock cycle extend the pulse by 6.25 ns (160 MHz clock). An example of the effect on the stretch setting is shown in figure 5.1.

in0_DEL `[unsigned int, 0]` Defines the delay, in 160 MHz clock cycles, to be assigned to the discriminated pulse from input 0, in order to process the logic for the trigger. This can be used to compensate for differences in cable lengths for the signals used to create a trigger.

in1_STR `[unsigned int, 0]` Same as `in1_STR` but for input 1.

in1_DEL `[unsigned int, 0]` Same as `in1_DEL` but for input 1.

in2_STR `[unsigned int, 0]` Same as `in1_STR` but for input 2.

in2_DEL `[unsigned int, 0]` Same as `in1_DEL` but for input 1.

in3_STR `[unsigned int, 0]` Same as `in1_STR` but for input 3.

in3_DEL `[unsigned int, 0]` Same as `in1_DEL` but for input 1.

trigMaskHi `[unsigned int32, 0]` This word represents the most significative bits of the 64-bits used to determine the trigger mask.
A detailed explanation of how to determine the correct word is provided in section 5.1.

trigMaskLo `[unsigned int32, 0]` This word represents the least significative bits of the 64-bits used to determine the trigger mask.
A detailed explanation of how to determine the correct word is provided in section 5.1.

DUTMask `[unsigned int, 0x1]` This mask indicates which HDMI inputs have an AIDA device connected. Each of the lowest four bits correspond to a connector (bit 0= DUT1, bit 1= DUT2, bit 2= DUT3, bit 3= DUT4). If the bit is set to 1 the TLU expects a device connected and exchanging signals according to the mode selected (see DUTMaskMode).

DUTMaskMode `[unsigned int, 0xFF]` Defines the mode of operation of the device connected to a specific HDMI port.
Two bits are needed for each device, so bits 0,1 refer to HDMI1, bits 2, 3 refer to HDMI2, etc. Currently only the lower bit of each pair is needed to specify if the device is in AIDA mode (`bX1`) or EUDET mode (`bx0`).
Example: to configure device 1 and 2 as EUDET and the rest as AIDA, the parameters should be set to 11-11-x0-x0, i.e. 0xF0 (but 0xFA, 0xF2 and 0xF8 would also work the same).
See also section 7.

DUTMaskModeModifier `[unsigned int, 0xF]` This mask only affects EUDET mode. Each of the lower 4 bits correspond to a device. If the device is in EUDET mode, it can assert DUTClk to produce a global veto in the triggers. This behaviour occurs if the corresponding bit is set to 1. If the bit is set to 0, asserting the DUTClk from the device will not produce a global veto.

DUTIgnoreBusy `[unsigned int, 0xF]` This mask tells the TLU to ignore the BUSY signal from a specific device, either in AIDA or EUDET mode. If the device is in AIDA mode, this means that further triggers will be issued while the device is busy. If the device is in EUDET mode, this means that the TLU will not pause while they are in the handshake phase. In turn, this means that the device will likely receive events where the trigger number does not increase sequentially by one.

DUTIgnoreShutterVeto `[unsigned int, 0x1]` Set bit to 1 to tell the DUT to ignore the shutter signal.

EnableRecordData `[boolean, true]` if set to 1, enable the data recording in the TLU.

InternalTriggerInterval `[unsigned int, 0]` Defines the rate of the trigger generated internally by the TLU: if 0, the internal triggers are disabled. Any other value produces internal triggers with a frequency of 160 MHz/value.

Chapter 9

Appendix

2.4 Top view

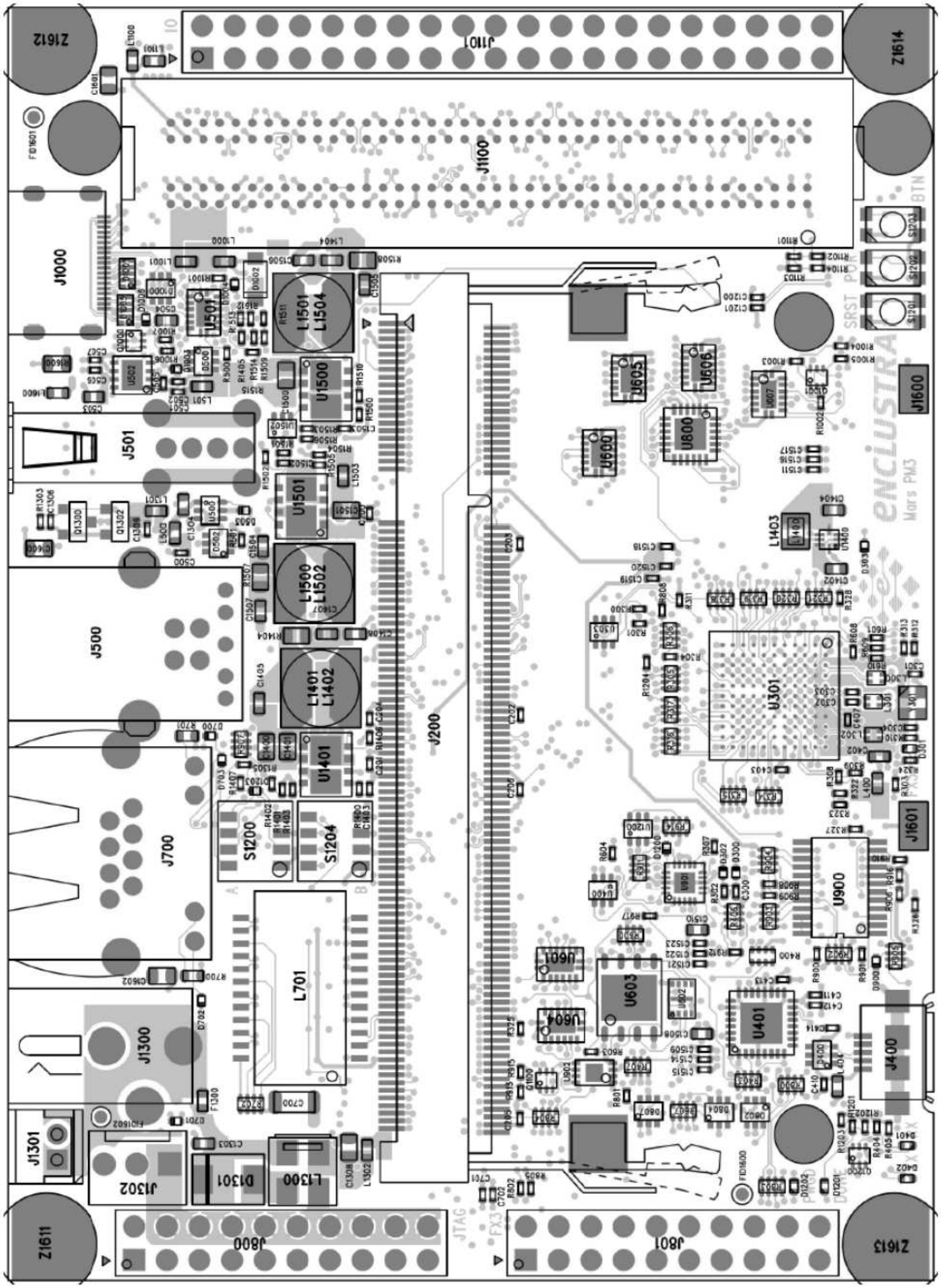
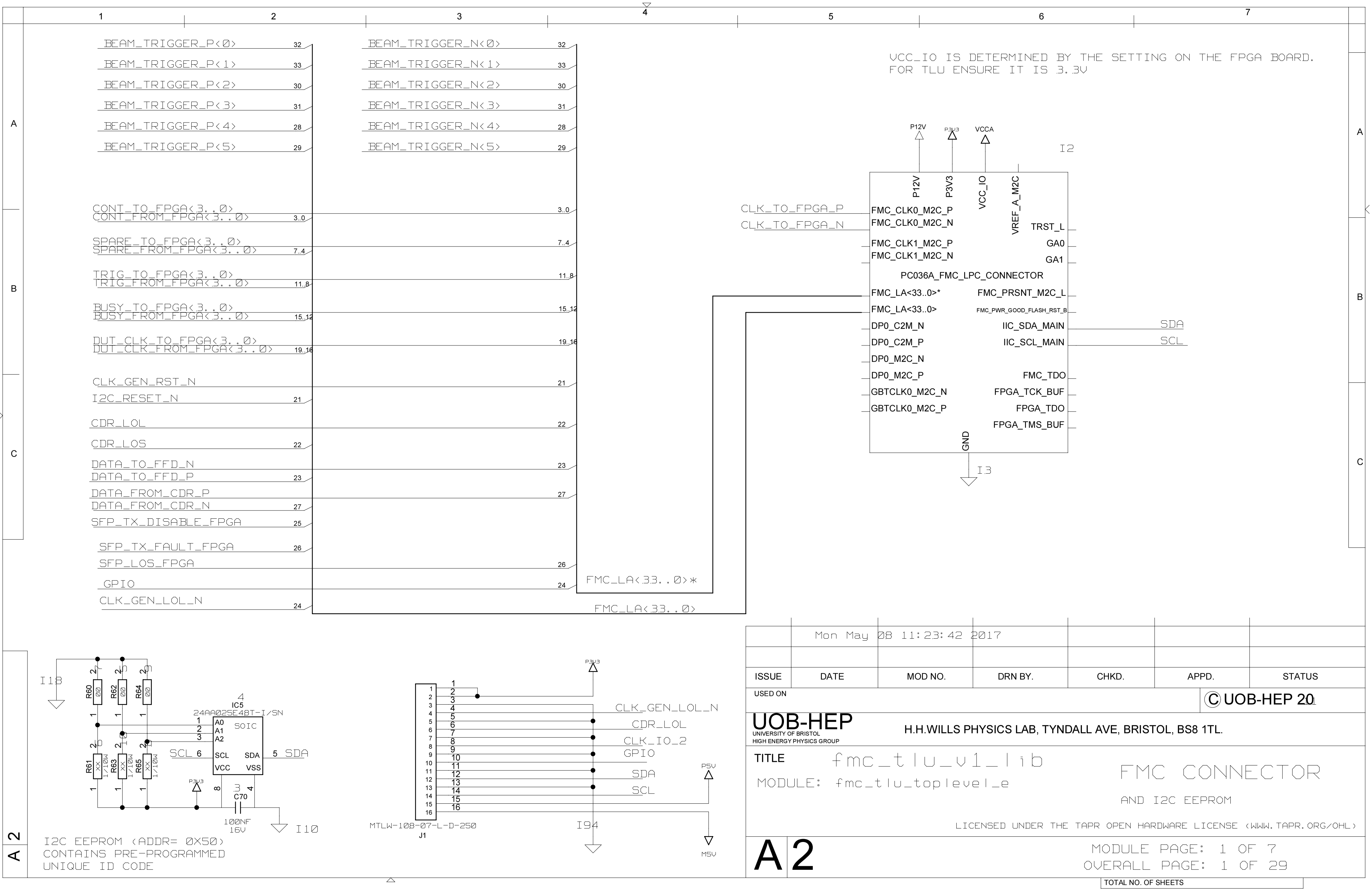


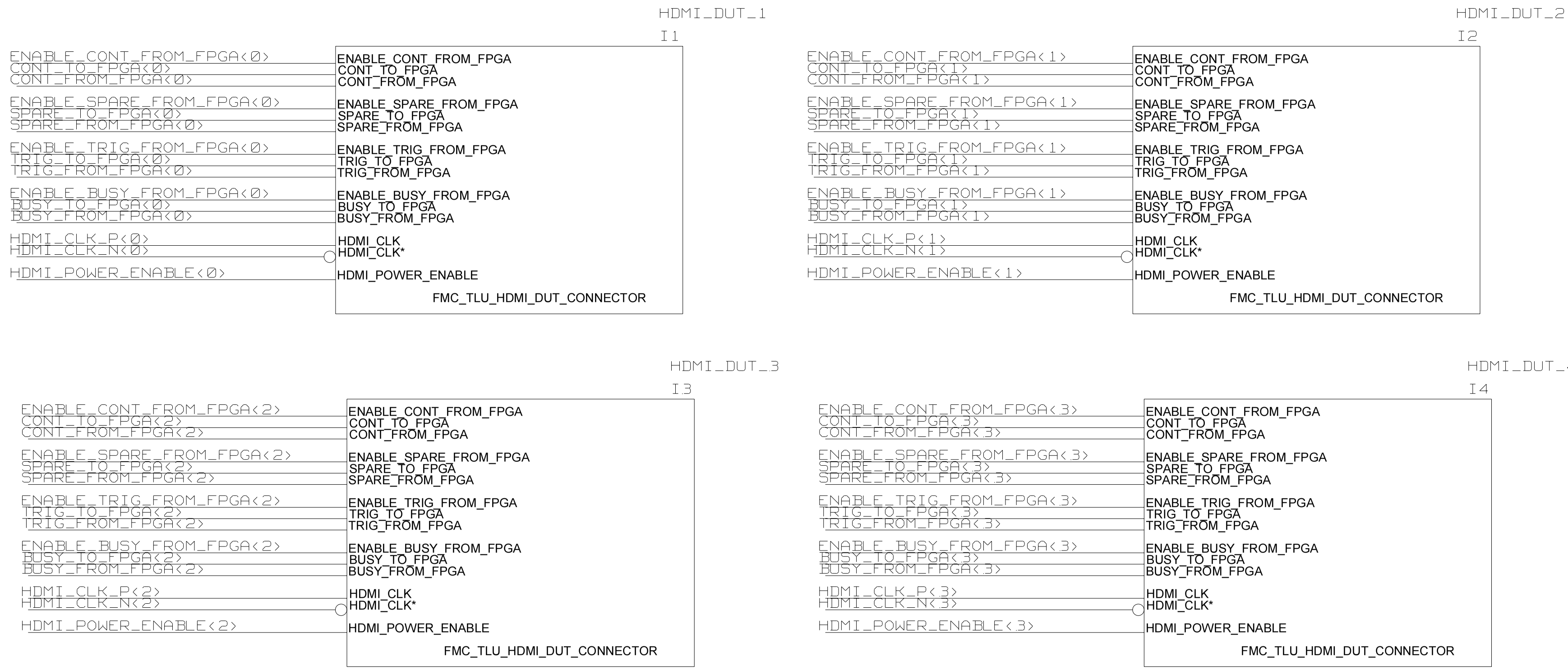
Figure 3: Top view

Schematic side			FPGA Side				
NET NAME	**FMC_LA**	**J4**	**FMC NAME**	**PACKAGE_PIN**	**VDHL NAME**	**FPGA IN/OUT**	**CONSTRAINT INSTRUCTION**
BEAM_TRIGGER_P<0>	FMC_LA<32>	H37	LA32_P	B1	threshold_discr_p_i[0]	In	set_property PACKAGE_PIN B1 [get_ports {threshold_discr_p_i[0]}]
BEAM_TRIGGER_P<1>	FMC_LA<33>	G36	LA33_P	C4	threshold_discr_p_i[1]	In	set_property PACKAGE_PIN C4 [get_ports {threshold_discr_p_i[1]}]
BEAM_TRIGGER_P<2>	FMC_LA<30>	H34	LA30_P	K2	threshold_discr_p_i[2]	In	set_property PACKAGE_PIN K2 [get_ports {threshold_discr_p_i[2]}]
BEAM_TRIGGER_P<3>	FMC_LA<31>	G33	LA31_P	C6	threshold_discr_p_i[3]	In	set_property PACKAGE_PIN C6 [get_ports {threshold_discr_p_i[3]}]
BEAM_TRIGGER_P<4>	FMC_LA<28>	H31	LA28_P	J4	threshold_discr_p_i[4]	In	set_property PACKAGE_PIN J4 [get_ports {threshold_discr_p_i[4]}]
BEAM_TRIGGER_P<5>	FMC_LA<29>	G30	LA29_P	H1	threshold_discr_p_i[5]	In	set_property PACKAGE_PIN H1 [get_ports {threshold_discr_p_i[5]}]
BEAM_TRIGGER_N<0>	FMC_LA*<32>	H38	LA32_N	A1	threshold_discr_n_i[0]	In	set_property PACKAGE_PIN A1 [get_ports {threshold_discr_n_i[0]}]
BEAM_TRIGGER_N<1>	FMC_LA*<33>	G37	LA33_N	B4	threshold_discr_n_i[1]	In	set_property PACKAGE_PIN B4 [get_ports {threshold_discr_n_i[1]}]
BEAM_TRIGGER_N<2>	FMC_LA*<30>	H35	LA30_N	K1	threshold_discr_n_i[2]	In	set_property PACKAGE_PIN K1 [get_ports {threshold_discr_n_i[2]}]
BEAM_TRIGGER_N<3>	FMC_LA*<31>	G34	LA31_N	C5	threshold_discr_n_i[3]	In	set_property PACKAGE_PIN C5 [get_ports {threshold_discr_n_i[3]}]
BEAM_TRIGGER_N<4>	FMC_LA*<28>	H32	LA28_N	H4	threshold_discr_n_i[4]	In	set_property PACKAGE_PIN H4 [get_ports {threshold_discr_n_i[4]}]
BEAM_TRIGGER_N<5>	FMC_LA*<29>	G31	LA29_N	G1	threshold_discr_n_i[5]	In	set_property PACKAGE_PIN G1 [get_ports {threshold_discr_n_i[5]}]
				P17	enclustra_clk	In	
CLK_TO_FPGA_P	FMC_CLK0_M2C_P	H4	CLK0_M2C_P	T5	sysclk_40_i_p	In	set_property PACKAGE_PIN T5 [get_ports {sysclk_40_i_p}]
CLK_TO_FPGA_N	FMC_CLK0_M2C_N	H5	CLK0_M2C_N	T4	sysclk_40_i_n	In	set_property PACKAGE_PIN T4 [get_ports {sysclk_40_i_n}]
CLK_FROM_FPGA_P	FMC_CLK1_M2C_P	G2	CLK1_M2C_P	E3	sysclk_50_o_p	Out	set_property PACKAGE_PIN E3 [get_ports {sysclk_50_o_p}]
CLK_FROM_FPGA_N	FMC_CLK1_M2C_N	G3	CLK1_M2C_N	D3	sysclk_50_o_n	Out	set_property PACKAGE_PIN D3 [get_ports {sysclk_50_o_n}]
I2C_RESET_N	FMC_LA<21>	H25	LA21_P	C2	i2c_reset	Out	set_property PACKAGE_PIN C2 [get_ports {i2c_reset}]
GPIO	FMC_LA*<24>	H29	LA24_N	F6	gpio	In/Out	set_property PACKAGE_PIN F6 [get_ports {gpio}]
CLK_GEN_RST_N	FMC_LA*<21>	H26	LA21_N	C1	clk_gen_rst	Out	set_property PACKAGE_PIN C1 [get_ports {clk_gen_rst}]
CLK_GEN_LOL_N	FMC_LA<0>	G6	LA00_P_CC			In	
CONT_TO_FPGA<0>	FMC_LA*<0>	G7	LA00_N_CC	P5	cont_i[0]	In	set_property PACKAGE_PIN P5 [get_ports {cont_i[0]}]
CONT_TO_FPGA<1>	FMC_LA*<1>	D9	LA01_N_CC	P3	cont_i[1]	In	set_property PACKAGE_PIN P3 [get_ports {cont_i[1]}]
CONT_TO_FPGA<2>	FMC_LA*<2>	H8	LA02_N	N6	cont_i[2]	In	set_property PACKAGE_PIN N6 [get_ports {cont_i[2]}]
CONT_TO_FPGA<3>	FMC_LA*<3>	G10	LA03_N	L5	cont_i[3]	In	set_property PACKAGE_PIN L5 [get_ports {cont_i[3]}]
SPARE_TO_FPGA<0>	FMC_LA*<4>	H11	LA04_N	M1	spare_i[0]	In	set_property PACKAGE_PIN M1 [get_ports {spare_i[0]}]
SPARE_TO_FPGA<1>	FMC_LA*<5>	D12	LA05_N	N4	spare_i[1]	In	set_property PACKAGE_PIN N4 [get_ports {spare_i[1]}]
SPARE_TO_FPGA<2>	FMC_LA*<6>	C11	LA06_N	N1	spare_i[2]	In	set_property PACKAGE_PIN N1 [get_ports {spare_i[2]}]
SPARE_TO_FPGA<3>	FMC_LA*<7>	H14	LA07_N	M2	spare_i[3]	In	set_property PACKAGE_PIN M2 [get_ports {spare_i[3]}]
TRIG_TO_FPGA<0>	FMC_LA*<8>	G13	LA08_N	R5	triggers_i[0]	In	set_property PACKAGE_PIN R5 [get_ports {triggers_i[0]}]
TRIG_TO_FPGA<1>	FMC_LA*<9>	D15	LA09_N	R2	triggers_i[1]	In	set_property PACKAGE_PIN R2 [get_ports {triggers_i[1]}]
TRIG_TO_FPGA<2>	FMC_LA*<10>	C15	LA10_N	T1	triggers_i[2]	In	set_property PACKAGE_PIN T1 [get_ports {triggers_i[2]}]
TRIG_TO_FPGA<3>	FMC_LA*<11>	H17	LA11_N	V1	triggers_i[3]	In	set_property PACKAGE_PIN V1 [get_ports {triggers_i[3]}]
BUSY_TO_FPGA<0>	FMC_LA*<12>	G16	LA12_N	T6	busy_i[0]	In	set_property PACKAGE_PIN T6 [get_ports {busy_i[0]}]
BUSY_TO_FPGA<1>	FMC_LA*<13>	D18	LA13_N	U3	busy_i[1]	In	set_property PACKAGE_PIN U3 [get_ports {busy_i[1]}]
BUSY_TO_FPGA<3>	FMC_LA*<14>	C19	LA14_N	T8	busy_i[2]	In	set_property PACKAGE_PIN T8 [get_ports {busy_i[2]}]

BUSY_TO_FPGA<2>	FMC_LA*<15>	H20	LA15_N	L4	busy_i[3]	In	set_property PACKAGE_PIN L4 [get_ports {busy_i[3]}]
DUT_CLK_TO_FPGA<0>	FMC_LA*<16>	G19	LA16_N	L3	dut_clk_i[0]	In	set_property PACKAGE_PIN L3 [get_ports {dut_clk_i[0]}]
DUT_CLK_TO_FPGA<1>	FMC_LA*<17>	D21	LA17_N_CC	F3	dut_clk_i[1]	In	set_property PACKAGE_PIN F3 [get_ports {dut_clk_i[1]}]
DUT_CLK_TO_FPGA<2>	FMC_LA*<18>	C23	LA18_N_CC	D2	dut_clk_i[2]	In	set_property PACKAGE_PIN D2 [get_ports {dut_clk_i[2]}]
DUT_CLK_TO_FPGA<3>	FMC_LA*<19>	H23	LA19_N	G3	dut_clk_i[3]	In	set_property PACKAGE_PIN G3 [get_ports {dut_clk_i[3]}]
CONT_FROM_FPGA<0>	FMC_LA<0>	G6	LA00_P_CC	N5	cont_o[0]	Out	set_property PACKAGE_PIN N5 [get_ports {cont_o[0]}]
CONT_FROM_FPGA<1>	FMC_LA<1>	D8	LA01_P_CC	P4	cont_o[1]	Out	set_property PACKAGE_PIN P4 [get_ports {cont_o[1]}]
CONT_FROM_FPGA<2>	FMC_LA<2>	H7	LA02_P	M6	cont_o[2]	Out	set_property PACKAGE_PIN M6 [get_ports {cont_o[2]}]
CONT_FROM_FPGA<3>	FMC_LA<3>	G9	LA03_P	L6	cont_o[3]	Out	set_property PACKAGE_PIN L6 [get_ports {cont_o[3]}]
SPARE_FROM_FPGA<0>	FMC_LA<4>	H10	LA04_P	L1	spare_o[0]	Out	set_property PACKAGE_PIN L1 [get_ports {spare_o[0]}]
SPARE_FROM_FPGA<1>	FMC_LA<5>	D11	LA05_P	M4	spare_o[1]	Out	set_property PACKAGE_PIN M4 [get_ports {spare_o[1]}]
SPARE_FROM_FPGA<2>	FMC_LA<6>	C10	LA06_P	N2	spare_o[2]	Out	set_property PACKAGE_PIN N2 [get_ports {spare_o[2]}]
SPARE_FROM_FPGA<3>	FMC_LA<7>	H13	LA07_P	M3	spare_o[3]	Out	set_property PACKAGE_PIN M3 [get_ports {spare_o[3]}]
TRIG_FROM_FPGA<0>	FMC_LA<8>	G12	LA08_P	R6	triggers_o[0]	Out	set_property PACKAGE_PIN R6 [get_ports {triggers_o[0]}]
TRIG_FROM_FPGA<1>	FMC_LA<9>	D14	LA09_P	P2	triggers_o[1]	Out	set_property PACKAGE_PIN P2 [get_ports {triggers_o[1]}]
TRIG_FROM_FPGA<2>	FMC_LA<10>	C14	LA10_P	R1	triggers_o[2]	Out	set_property PACKAGE_PIN R1 [get_ports {triggers_o[2]}]
TRIG_FROM_FPGA<3>	FMC_LA<11>	H16	LA11_P	U1	triggers_o[3]	Out	set_property PACKAGE_PIN U1 [get_ports {triggers_o[3]}]
BUSY_FROM_FPGA<0>	FMC_LA<12>	G15	LA12_P	R7	busy_o[0]	Out	set_property PACKAGE_PIN R7 [get_ports {busy_o[0]}]
BUSY_FROM_FPGA<1>	FMC_LA<13>	D17	LA13_P	U4	busy_o[1]	Out	set_property PACKAGE_PIN U4 [get_ports {busy_o[1]}]
BUSY_FROM_FPGA<2>	FMC_LA<14>	C18	LA14_P	R8	busy_o[2]	Out	set_property PACKAGE_PIN R8 [get_ports {busy_o[2]}]
BUSY_FROM_FPGA<3>	FMC_LA<15>	H19	LA15_P	K5	busy_o[3]	Out	set_property PACKAGE_PIN K5 [get_ports {busy_o[3]}]
DUT_CLK_FROM_FPGA<0>	FMC_LA<16>	G18	LA16_P	K3	dut_clk_o[0]	Out	set_property PACKAGE_PIN K3 [get_ports {dut_clk_o[0]}]
DUT_CLK_FROM_FPGA<1>	FMC_LA<17>	D20	LA17_P_CC	F4	dut_clk_o[1]	Out	set_property PACKAGE_PIN F4 [get_ports {dut_clk_o[1]}]
DUT_CLK_FROM_FPGA<2>	FMC_LA<18>	C22	LA18_P_CC	E2	dut_clk_o[2]	Out	set_property PACKAGE_PIN E2 [get_ports {dut_clk_o[2]}]
DUT_CLK_FROM_FPGA<3>	FMC_LA<19>	H22	LA19_P	G4	dut_clk_o[3]	Out	set_property PACKAGE_PIN G4 [get_ports {dut_clk_o[3]}]



VCC_IO IS DETERMINED BY THE SETTING ON THE FPGA BOARD.
FOR TLU ENSURE IT IS 3.3V
BEAM_TRIGGER_P<0>
BEAM_TRIGGER_P<1>
BEAM_TRIGGER_P<2>
BEAM_TRIGGER_P<3>
BEAM_TRIGGER_P<4>
BEAM_TRIGGER_P<5>
BEAM_TRIGGER_N<0>
BEAM_TRIGGER_N<1>
BEAM_TRIGGER_N<2>
BEAM_TRIGGER_N<3>
BEAM_TRIGGER_N<4>
BEAM_TRIGGER_N<5>
CONT_TO_FPGA<3..0>
CONT_FROM_FPGA<3..0>
SPARE_TO_FPGA<3..0>
SPARE_FROM_FPGA<3..0>
TRIG_TO_FPGA<3..0>
TRIG_FROM_FPGA<3..0>
BUSY_TO_FPGA<3..0>
BUSY_FROM_FPGA<3..0>
DUT_CLK_TO_FPGA<3..0>
DUT_CLK_FROM_FPGA<3..0>
CLK_GEN_RST_N
I2C_RESET_N
CDR_LOL
CDR_LOS
DATA_TO_FFD_N
DATA_TO_FFD_P
DATA_FROM_CDR_P
DATA_FROM_CDR_N
SFP_TX_DISABLE_FPGA
SFP_TX_FAULT_FPGA
SFP_LOS_FPGA
GPIO
CLK_GEN_LOL_N
FMC_LA<33..0>*
FMC_LA<33..0>
P12V
P3V3
VCCA
I2
CLK_TO_FPGA_P
CLK_TO_FPGA_N
P12V
P3V3
VCC_IO
VREF_A_M2C
FMC_CLK0_M2C_P
FMC_CLK0_M2C_N
TRST_L
FMC_CLK1_M2C_P
GA0
FMC_CLK1_M2C_N
GA1
PC036A_FMC_LPC_CONNECTOR
FMC_LA<33..0>*
FMC_PRSNT_M2C_L
FMC_LA<33..0>
FMC_PWR_GOOD_FLASH_RST_B
DP0_C2M_N
IIC_SDA_MAIN
SDA
DP0_C2M_P
IIC_SCL_MAIN
SCL
DP0_M2C_N
DP0_M2C_P
FMC_TDO
GBTCLK0_M2C_N
FPGA_TCK_BUF
GBTCLK0_M2C_P
FPGA_TDO
FPGA_TMS_BUF
GND
I3
I18
R60 00
R62 00
R64 00
R61 XX 1/10W
R63 XX 1/10W
R65 XX 1/10W
IC5
24AA025E48T-I/SN
SOIC
A0
A1
A2
SCL
SDA
VCC
VSS
SCL
SDA
P3V3
C70
100NF
16V
I10
I2C EEPROM (ADDR= 0X50)
CONTAINS PRE-PROGRAMMED
UNIQUE ID CODE
MTLW-108-07-L-D-250
J1
P3V3
CLK_GEN_LOL_N
CDR_LOL
CLK_IO_2
GPIO
SDA
SCL
P5V
M5V
I94
Mon May 08 11:23:42 2017
ISSUE
DATE
MOD NO.
DRN BY.
CHKD.
APPD.
STATUS
USED ON
© UOB-HEP 20
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
TITLE
fmc_tlu_v1_lib
MODULE: fmc_tlu_toplevel_e
FMC CONNECTOR
AND I2C EEPROM
LICENSED UNDER THE TAPR OPEN HARDWARE LICENSE (WWW.TAPR.ORG/OHL)
A2
MODULE PAGE: 1 OF 7
OVERALL PAGE: 1 OF 29
TOTAL NO. OF SHEETS



Thu May 04 13:37:48 2017

ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

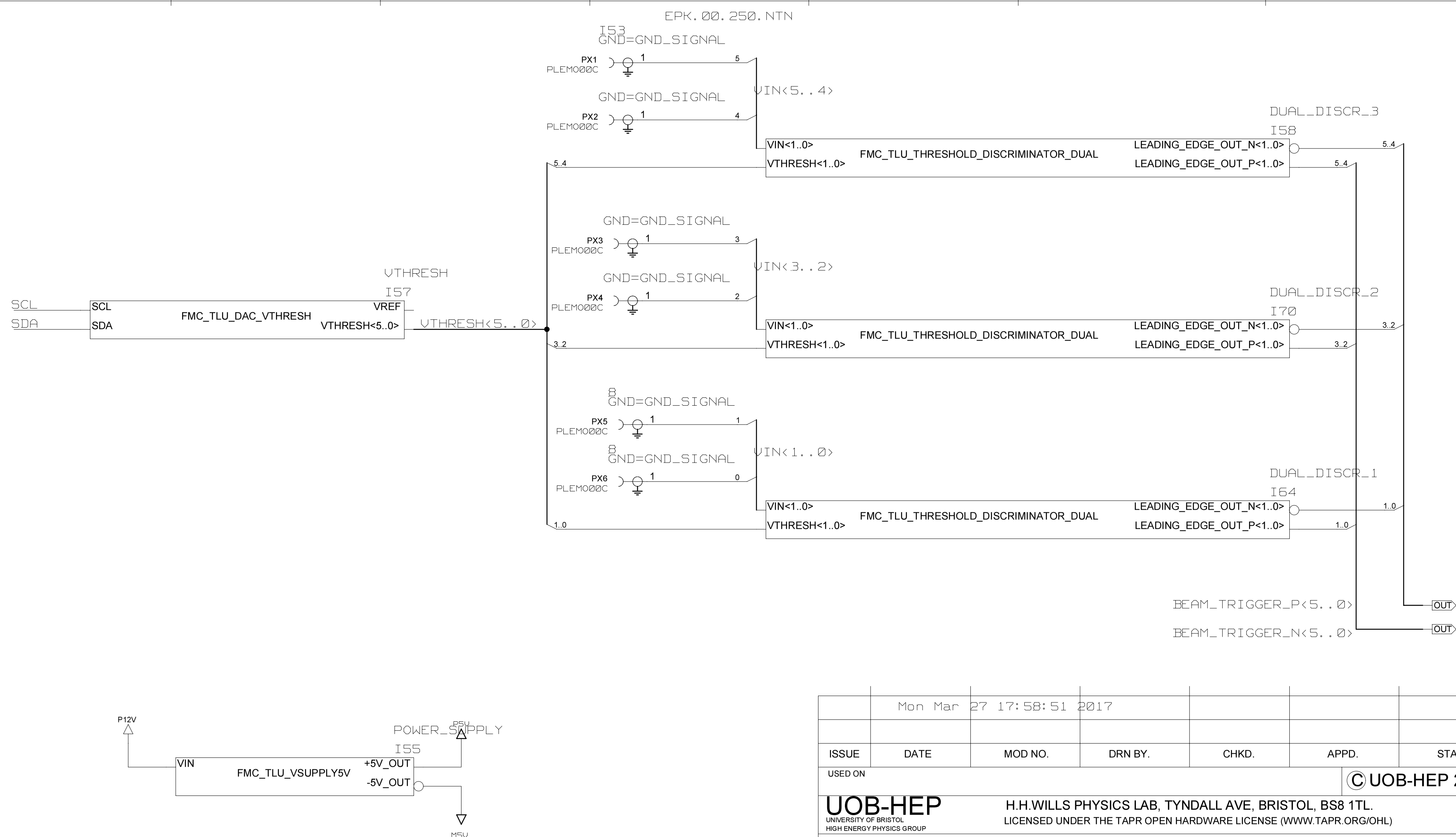
TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_toplevel_e

FRONT PANEL DUT CONNECTORS
3 X HDMI (DUT ONLY)
1 X HDMI (DUT OR UPLINK)

A2

A2



	Mon Mar 27 17:58:51 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

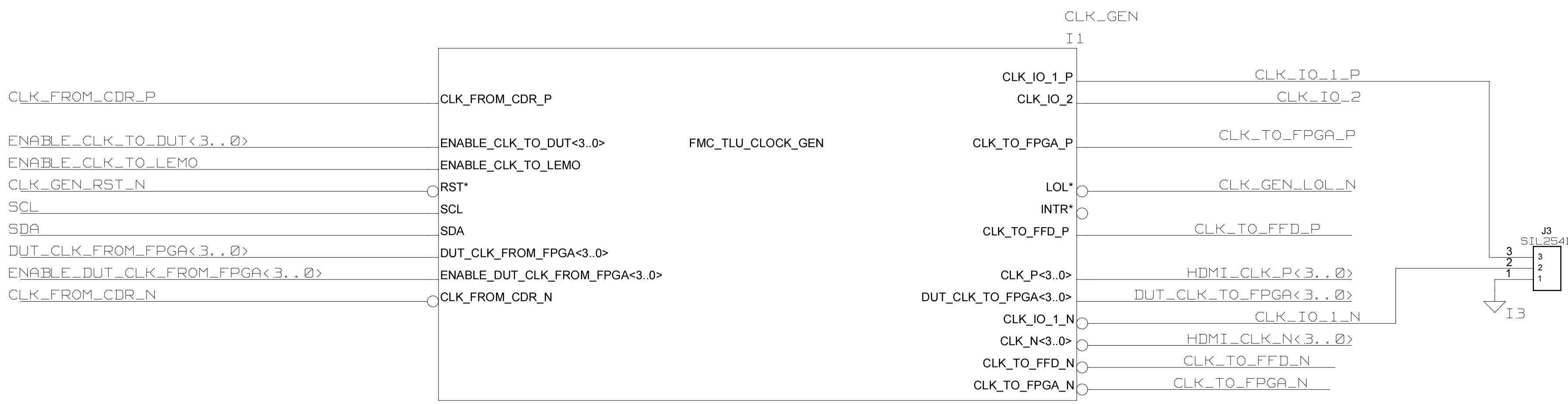
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_toplevel_e

DISCRIMINATORS

A2



Thu Mar 30 17:10:57 2017						
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

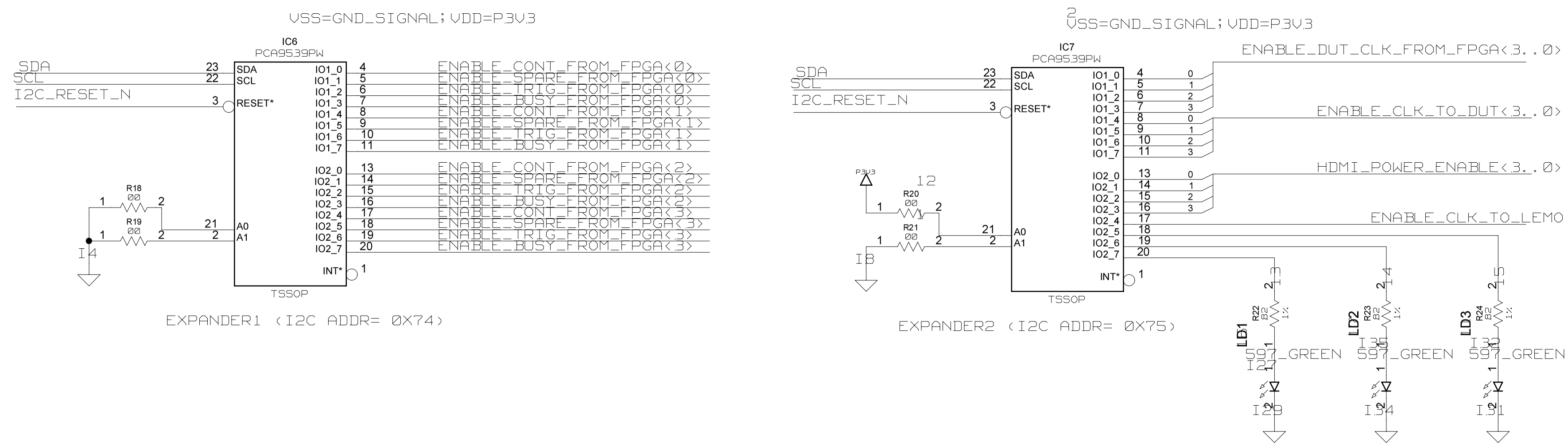
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib CLOCK GENERATION

MODULE: fmc_tlu_toplevel_e

A2



	Mon Mar 27 17:58:54 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20 11

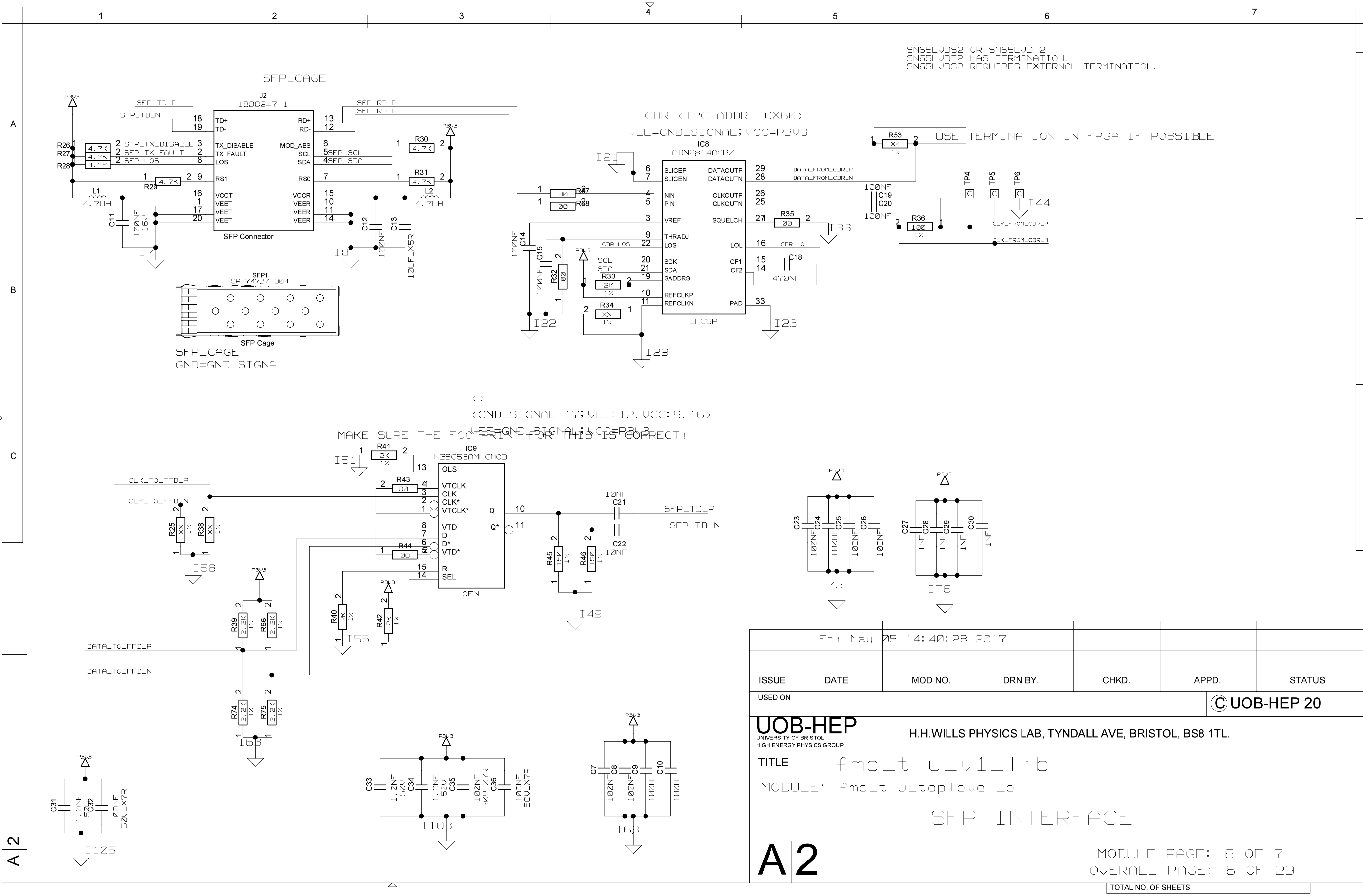
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

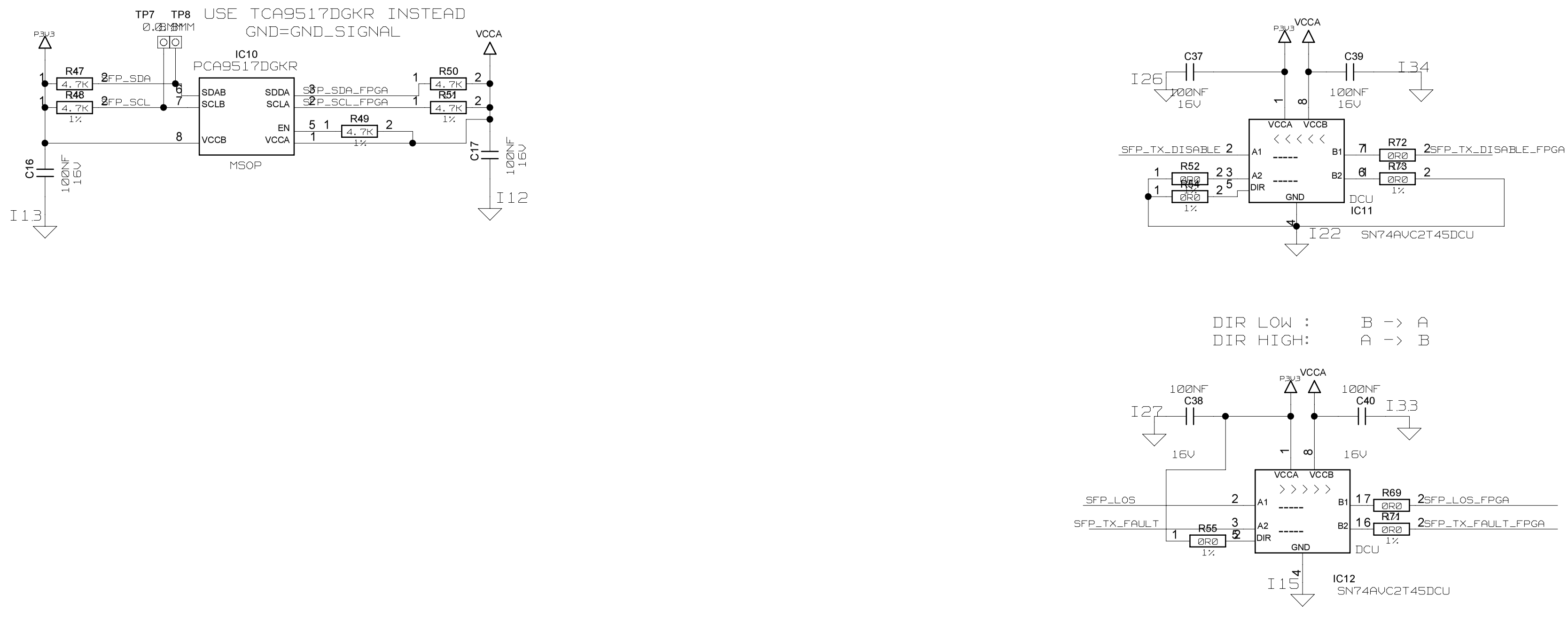
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib I2C I/O EXTENDERS

MODULE: fmc_tlu_toplevel_e

A2


SFP_CAGE
J2
1888247-1
SFP Connector
SFP1
SP-74737-004
SFP Cage
SFP_CAGE
GND=GND_SIGNAL
CDR (I2C ADDR= 0X60)
VEE=GND_SIGNAL;VCC=P3V3
IC8
ADN2814ACPZ
LFCSP
SN65LVDS2 OR SN65LVDT2
SN65LVDT2 HAS TERMINATION.
SN65LVDS2 REQUIRES EXTERNAL TERMINATION.
USE TERMINATION IN FPGA IF POSSIBLE
DATA_FROM_CDR_P
DATA_FROM_CDR_N
CLK_FROM_CDR_P
CLK_FROM_CDR_N
CDR_LOS
CDR_LOL
SFP_TD_P
SFP_TD_N
SFP_RD_P
SFP_RD_N
SFP_TX_DISABLE
SFP_TX_FAULT
SFP_LOS
SFP_SCL
SFP_SDA
()
(GND_SIGNAL:17;VEE:12;VCC:9,16)
VEE=GND_SIGNAL;VCC=P3V3
MAKE SURE THE FOOTPRINT FOR THIS IS CORRECT!
IC9
NBSG53AMNGMOD
QFN
CLK_TO_FFD_P
CLK_TO_FFD_N
DATA_TO_FFD_P
DATA_TO_FFD_N
Fri May 05 14:40:28 2017
ISSUE
DATE
MOD NO.
DRN BY.
CHKD.
APPD.
STATUS
USED ON
© UOB-HEP 20
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
TITLE
fmc_tlu_v1_lib
MODULE: fmc_tlu_toplevel_e
SFP INTERFACE
A2
MODULE PAGE: 6 OF 7
OVERALL PAGE: 6 OF 29
TOTAL NO. OF SHEETS



	Thu Mar 30 17:11:07 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

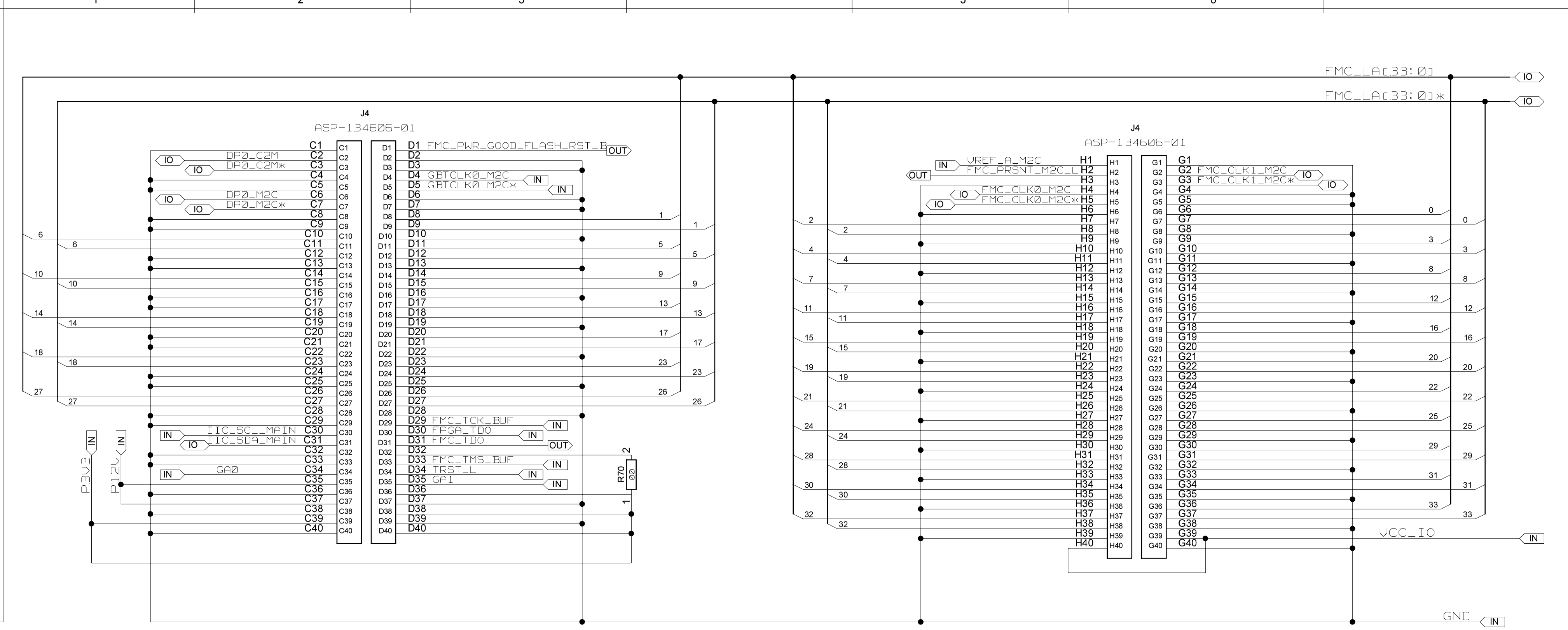
MODULE: fmc_tlu_toplevel_e

LEVEL TRANSLATORS

A2

MODULE PAGE: 7 OF 7
OVERALL PAGE: 7 OF 29

TOTAL NO. OF SHEETS



	Thu Mar 30 11:19:40 2017					
			D. CUSSANS	C. JESKE	D. CUSSANS	
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 2010

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
LICENSED UNDER THE TAPR OPEN HARDWARE LICENSE (WWW.TAPR.ORG/OHL)

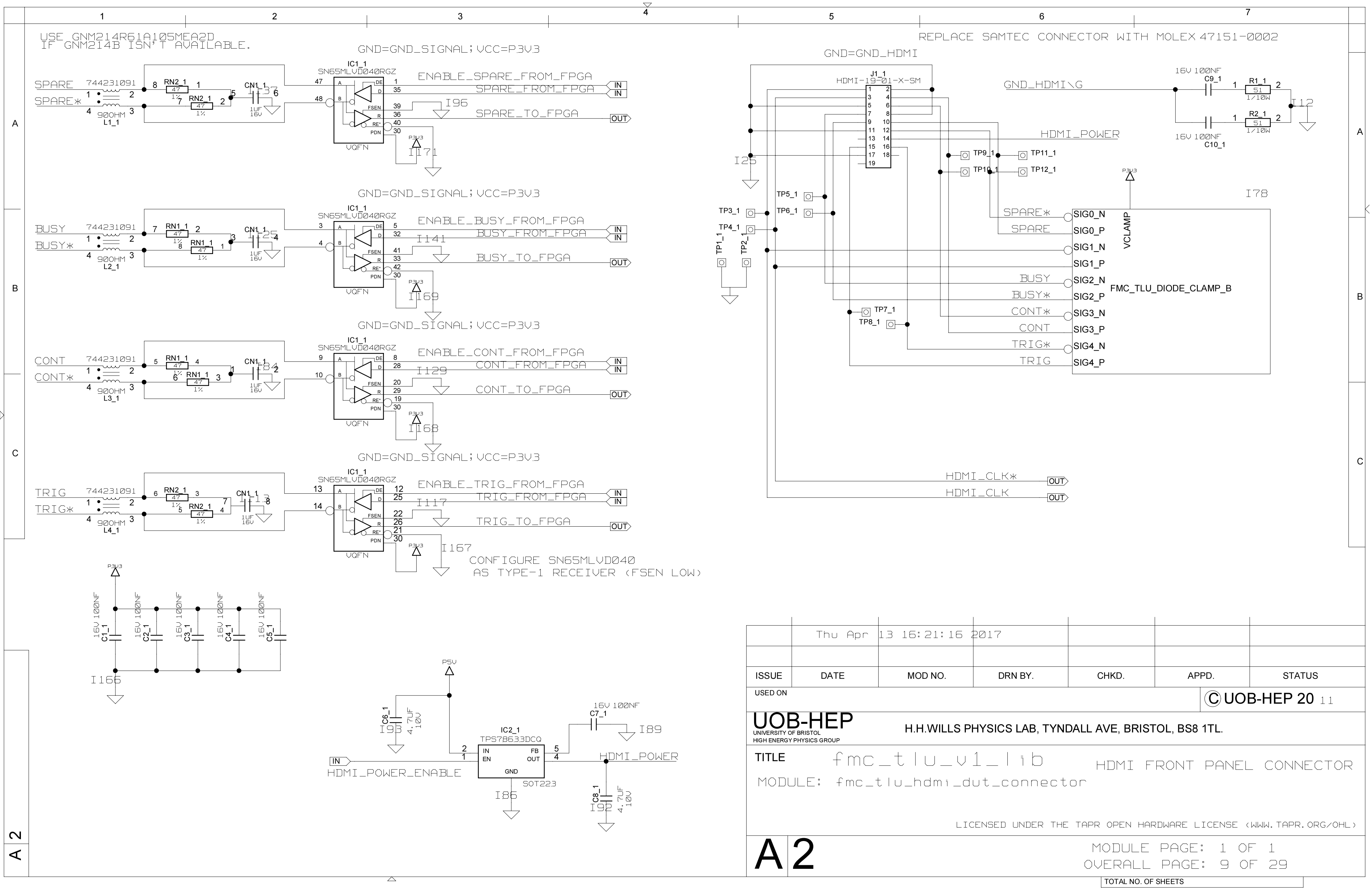
TITLE fmc_tlu_v1_lib

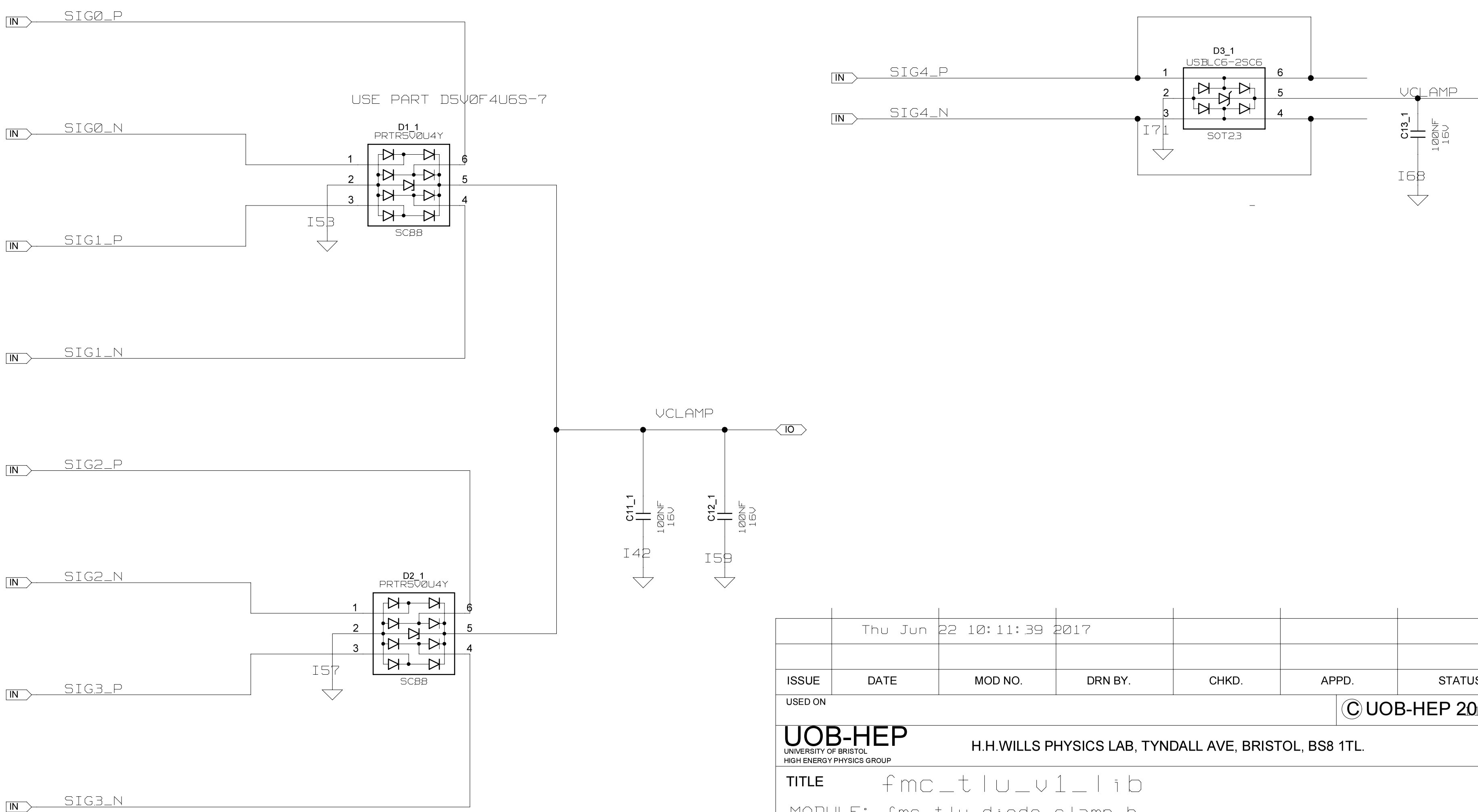
MODULE: pc036a_fmc_lpc_connector

FMC LPC

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 8 OF 29
TOTAL NO. OF SHEETS


USE GNM214R61A105MEA2D
IF GNM214B ISN'T AVAILABLE.
GND=GND_SIGNAL;VCC=P3V3
SPARE
SPARE*
BUSY
BUSY*
CONT
CONT*
TRIG
TRIG*
744231091
90OHM
SN65MLVD040RGZ
VQFN
ENABLE_SPARE_FROM_FPGA
SPARE_FROM_FPGA
SPARE_TO_FPGA
ENABLE_BUSY_FROM_FPGA
BUSY_FROM_FPGA
BUSY_TO_FPGA
ENABLE_CONT_FROM_FPGA
CONT_FROM_FPGA
CONT_TO_FPGA
ENABLE_TRIG_FROM_FPGA
TRIG_FROM_FPGA
TRIG_TO_FPGA
CONFIGURE SN65MLVD040
AS TYPE-1 RECEIVER (FSEN LOW)
REPLACE SAMTEC CONNECTOR WITH MOLEX 47151-0002
GND=GND_HDMI
HDMI-19-01-X-SM
GND_HDMI\G
HDMI_POWER
FMC_TLU_DIODE_CLAMP_B
VCLAMP
HDMI_CLK*
HDMI_CLK
HDMI_POWER_ENABLE
TPS78633DCQ
SOT223
Thu Apr 13 16:21:16 2017
ISSUE
DATE
MOD NO.
DRN BY.
CHKD.
APPD.
STATUS
USED ON
© UOB-HEP 20 11
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
TITLE
fmc_tlu_v1_lib
HDMI FRONT PANEL CONNECTOR
MODULE: fmc_tlu_hdmi_dut_connector
LICENSED UNDER THE TAPR OPEN HARDWARE LICENSE (WWW.TAPR.ORG/OHL)
A2
MODULE PAGE: 1 OF 1
OVERALL PAGE: 9 OF 29
TOTAL NO. OF SHEETS



	Thu Jun 22 10:11:39 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

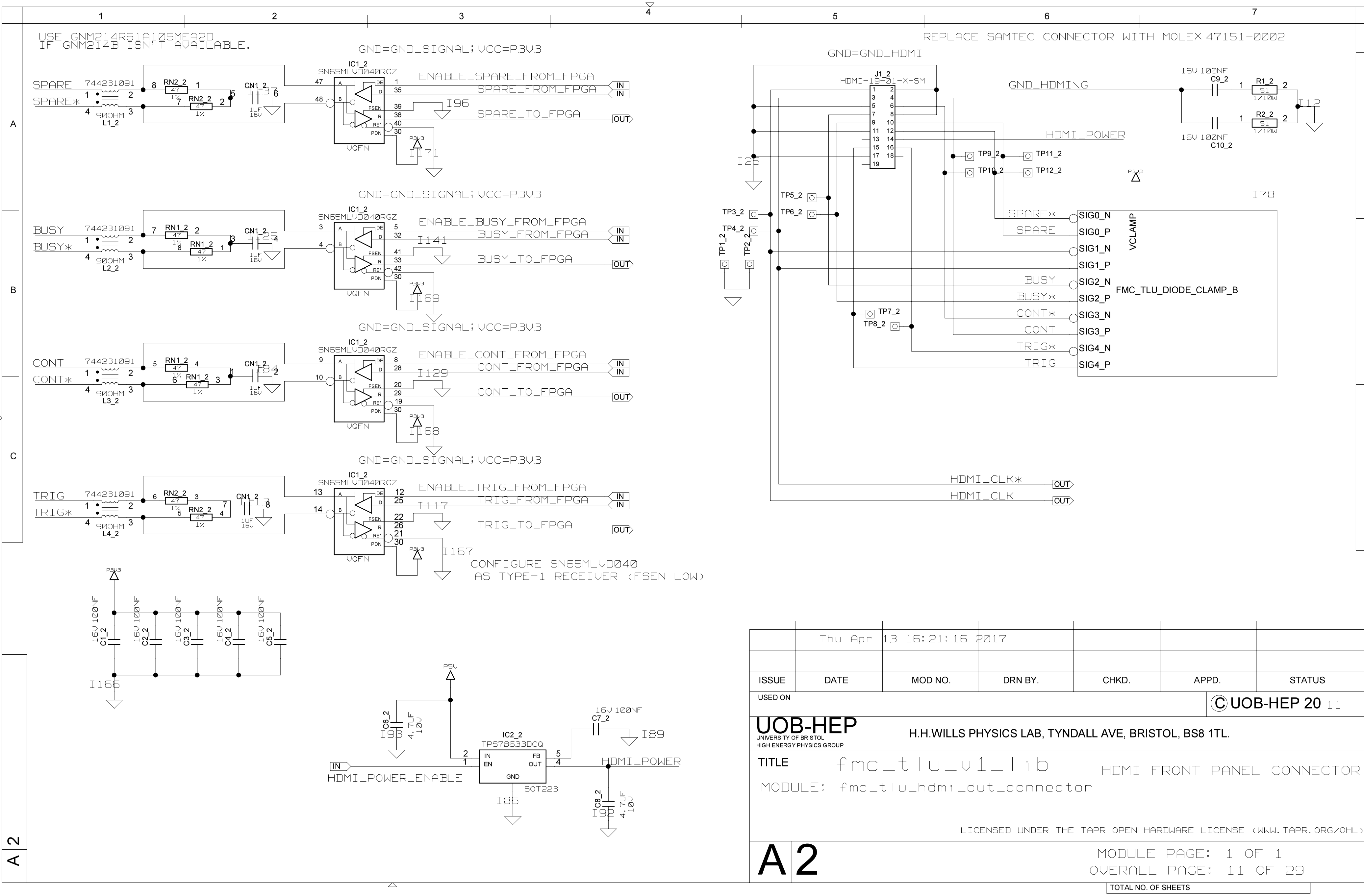
TITLE fmc_tlu_v1_lib

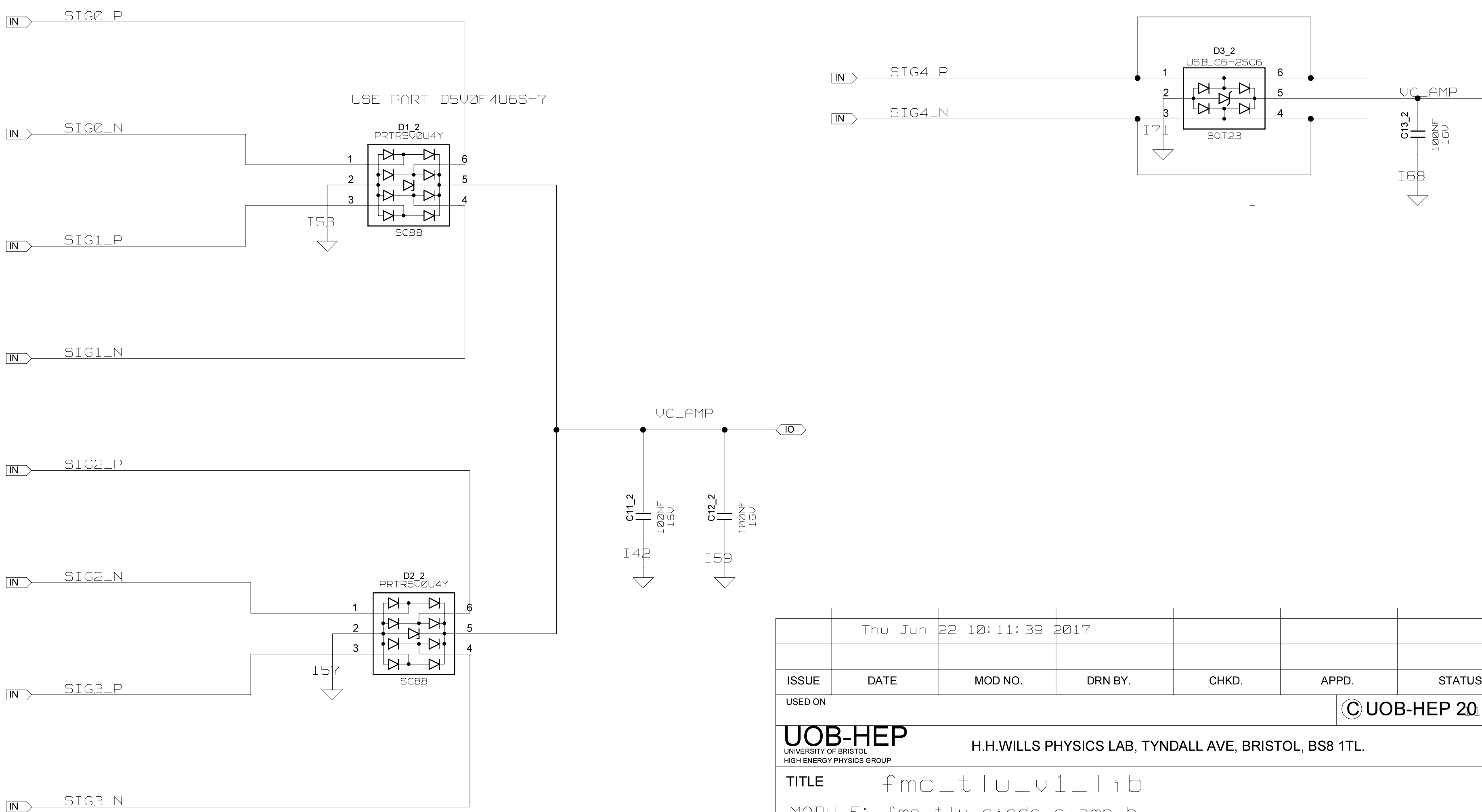
MODULE: fmc_tlu_diode_clamp_b

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 10 OF 29

TOTAL NO. OF SHEETS


USE GNM214R61A105MEA2D
IF GNM214B ISN'T AVAILABLE.
GND=GND_SIGNAL;VCC=P3V3
SN65MLVD040RGZ
ENABLE_SPARE_FROM_FPGA
SPARE_FROM_FPGA
SPARE_TO_FPGA
ENABLE_BUSY_FROM_FPGA
BUSY_FROM_FPGA
BUSY_TO_FPGA
ENABLE_CONT_FROM_FPGA
CONT_FROM_FPGA
CONT_TO_FPGA
ENABLE_TRIG_FROM_FPGA
TRIG_FROM_FPGA
TRIG_TO_FPGA
SPARE
SPARE*
BUSY
BUSY*
CONT
CONT*
TRIG
TRIG*
744231091
90OHM
CONFIGURE SN65MLVD040
AS TYPE-1 RECEIVER (FSEN LOW)
REPLACE SAMTEC CONNECTOR WITH MOLEX 47151-0002
GND=GND_HDMI
HDMI-19-01-X-SM
GND_HDMI\G
HDMI_POWER
FMC_TLU_DIODE_CLAMP_B
VCLAMP
SIG0_N
SIG0_P
SIG1_N
SIG1_P
SIG2_N
SIG2_P
SIG3_N
SIG3_P
SIG4_N
SIG4_P
HDMI_CLK*
HDMI_CLK
TPS78633DCQ
SOT223
HDMI_POWER_ENABLE
Thu Apr 13 16:21:16 2017
ISSUE
DATE
MOD NO.
DRN BY.
CHKD.
APPD.
STATUS
USED ON
© UOB-HEP 20 11
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
TITLE
fmc_tlu_v1_lib
HDMI FRONT PANEL CONNECTOR
MODULE: fmc_tlu_hdmi_dut_connector
LICENSED UNDER THE TAPR OPEN HARDWARE LICENSE (WWW.TAPR.ORG/OHL)
A2
MODULE PAGE: 1 OF 1
OVERALL PAGE: 11 OF 29
TOTAL NO. OF SHEETS



	Thu Jun 22 10:11:39 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

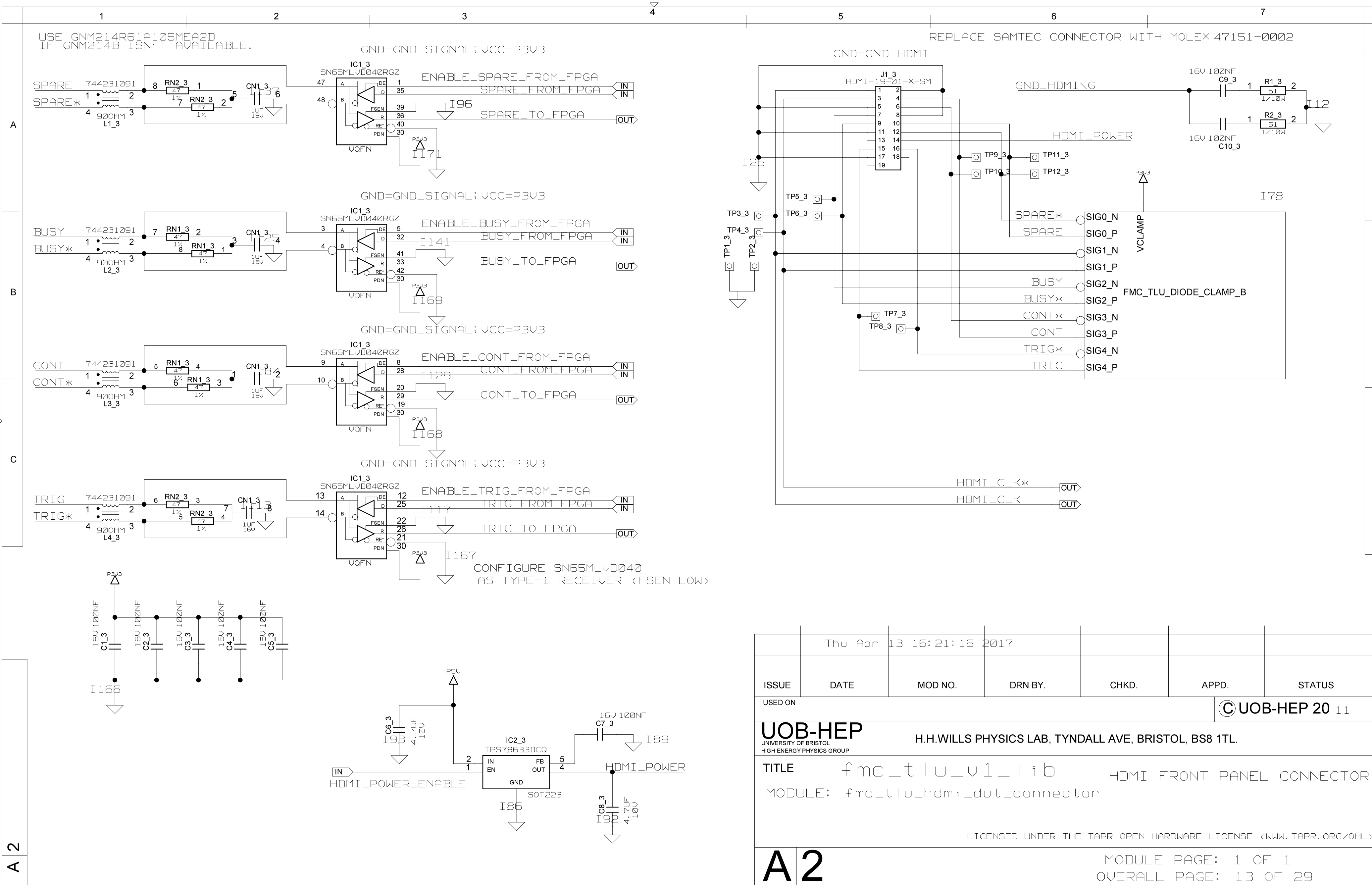
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_diode_clamp_b

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 12 OF 29
TOTAL NO. OF SHEETS



Thu Apr 13 16:21:16 2017

ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

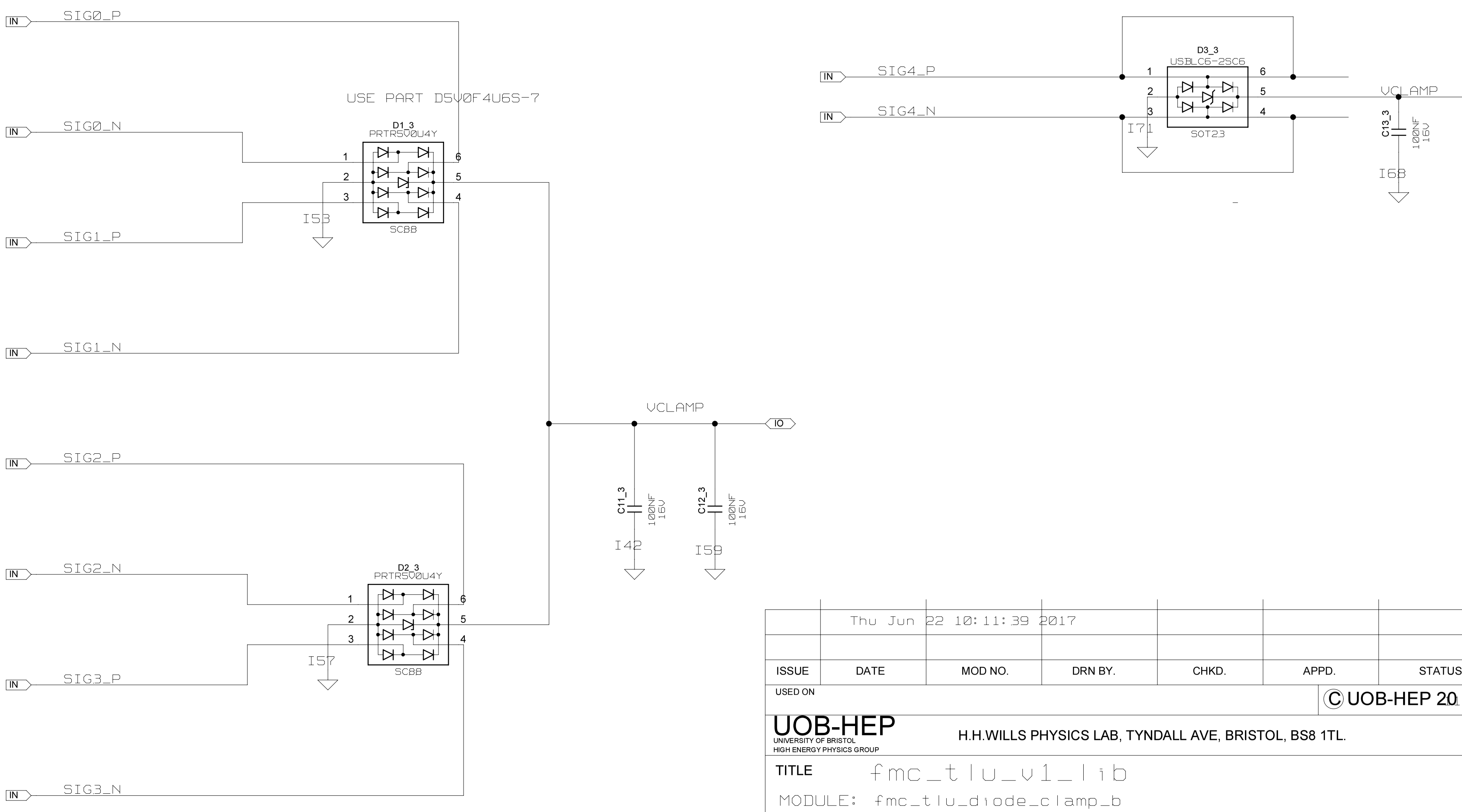
USED ON

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib HDMI FRONT PANEL CONNECTOR

MODULE: fmc_tlu_hdmi_dut_connector

A2



	Thu Jun 22 10:11:39 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

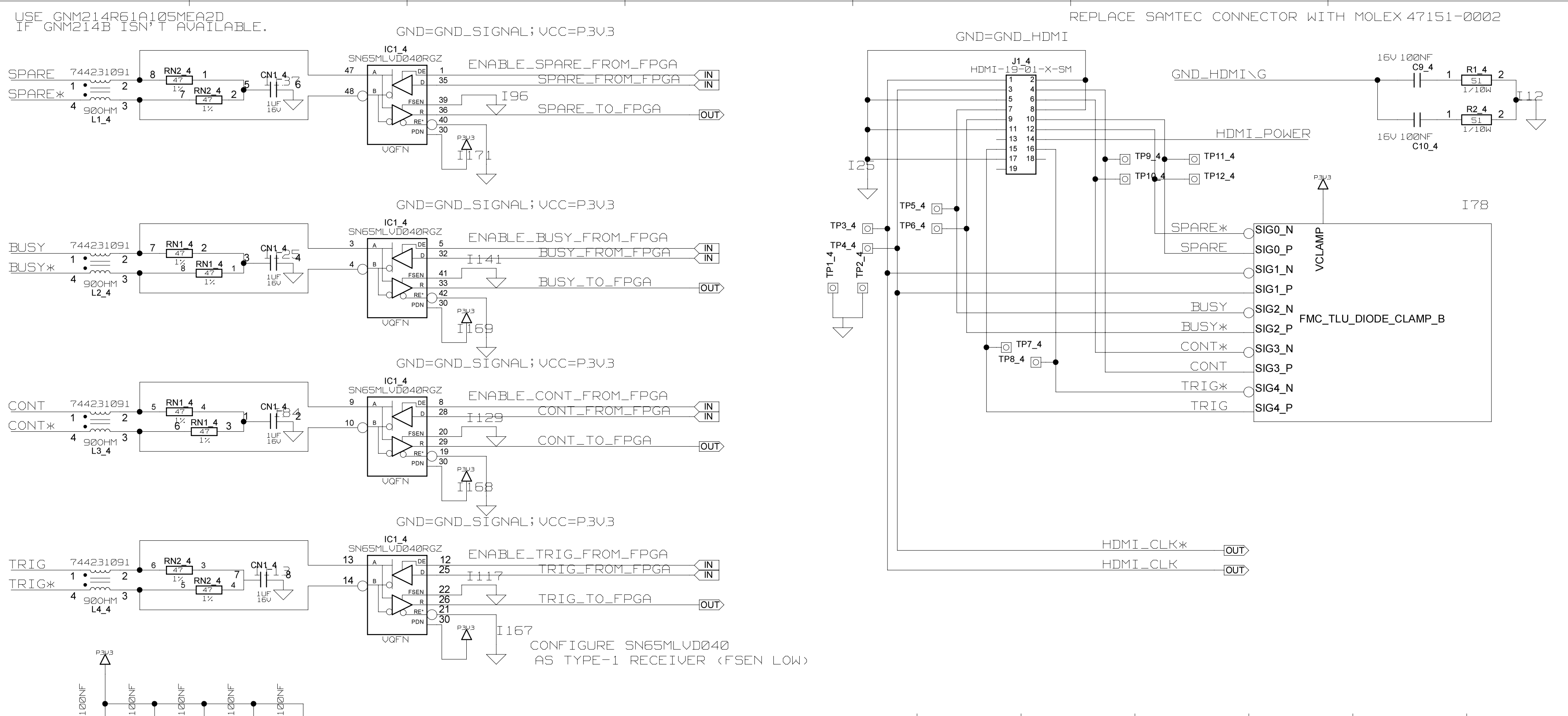
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_diode_clamp_b

A2



Thu Apr 13 16:21:16 2017

ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

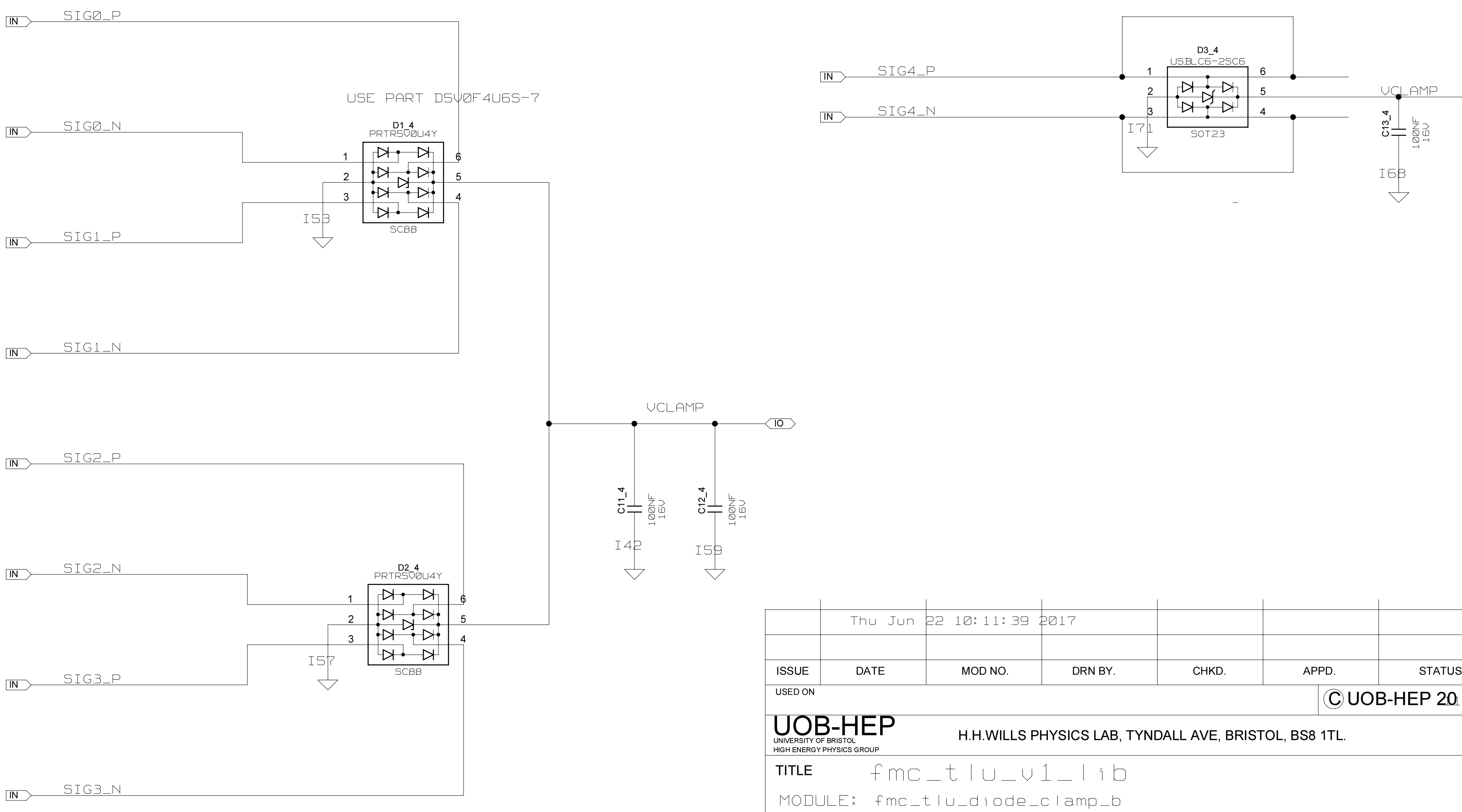
USED ON

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib HDMI FRONT PANEL CONNECTOR

MODULE: fmc_tlu_hdmi_dut_connector

A2



	Thu Jun 22 10:11:39 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

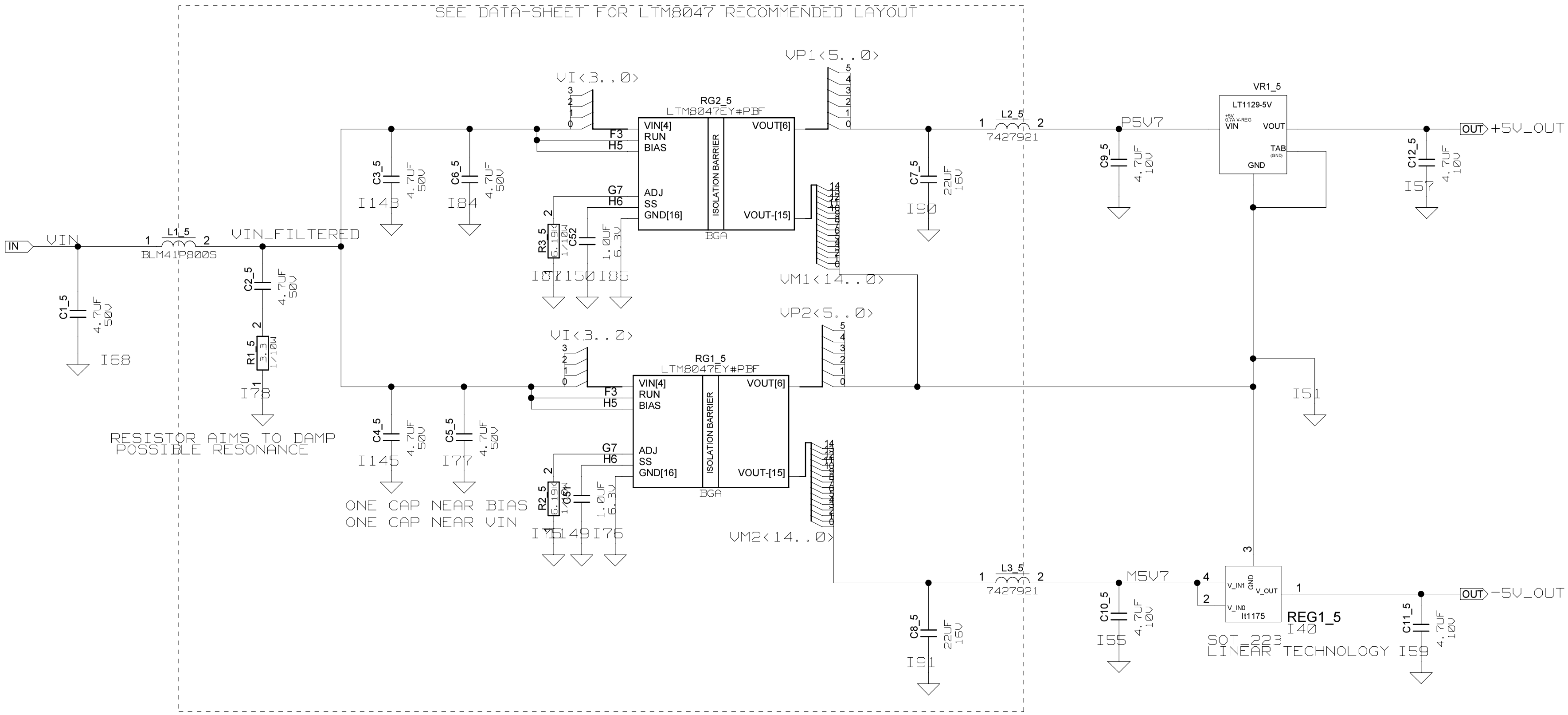
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_diode_clamp_b

A2



PUT DC-DC MODULES ON
COPPER AREA SURROUNDED BY
A MOAT (SPLIT IN PLANES)
CROSS THE MOAT IN ONE PLACE
WITH VIN , +5V , -5V AND GROUND

	Fri May 05 14:39:22 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_vsupply5v

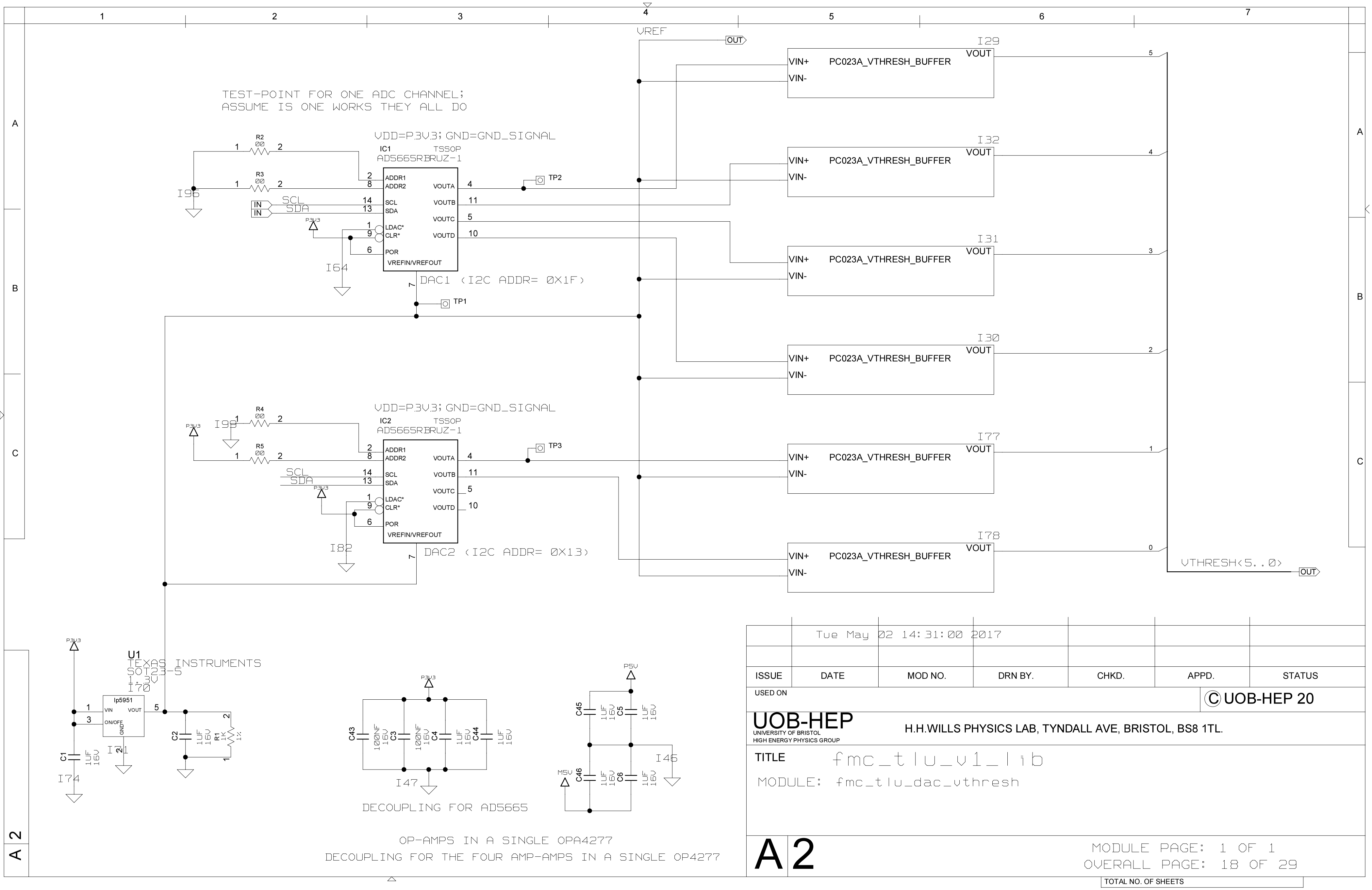
SUPPLY_DIGITAL

OVERALL PAGE: 17 OF 29

A2

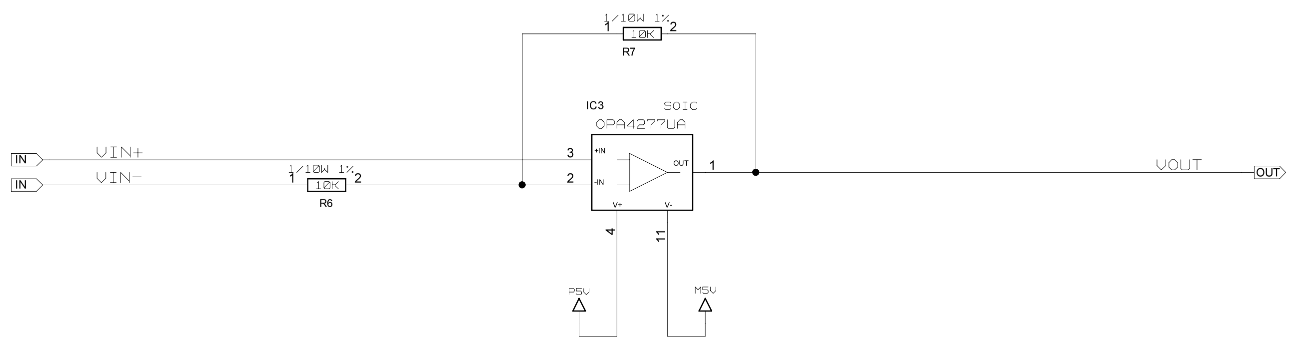
MODULE PAGE: 1 OF 1

TOTAL NO. OF SHEETS



TEST-POINT FOR ONE ADC CHANNEL;
ASSUME IS ONE WORKS THEY ALL DO
VDD=P3V3; GND=GND_SIGNAL
IC1
AD5665RBRUZ-1
TSSOP
ADDR1
ADDR2
SCL
SDA
LDAC*
CLR*
POR
VREFIN/VREFOUT
VOUTA
VOUTB
VOUTC
VOUTD
DAC1 (I2C ADDR= 0X1F)
TP1
TP2
TP3
R2
R3
R4
R5
0Ω
I96
I64
I98
I82
SCL
SDA
IN
P3V3
VDD=P3V3; GND=GND_SIGNAL
IC2
AD5665RBRUZ-1
DAC2 (I2C ADDR= 0X13)
VREF
OUT
I29
I32
I31
I30
I77
I78
VIN+
VIN-
VOUT
PC023A_VTHRESH_BUFFER
VTHRESH<5..0>
U1
TEXAS INSTRUMENTS
SOT23-5
1.3V
I70
lp5951
VIN
VOUT
ON/OFF
GND
C1
C2
1UF
16V
R1
1K
1%
I71
I74
C43
C3
C4
C44
100NF
I47
DECOUPLING FOR AD5665
C45
C5
C46
C6
P5V
M5V
I46
OP-AMPS IN A SINGLE OPA4277
DECOUPLING FOR THE FOUR AMP-AMPS IN A SINGLE OP4277
Tue May 02 14:31:00 2017
ISSUE
DATE
MOD NO.
DRN BY.
CHKD.
APPD.
STATUS
USED ON
© UOB-HEP 20
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
TITLE
fmc_tlu_v1_lib
MODULE: fmc_tlu_dac_vthresh
A2
MODULE PAGE: 1 OF 1
OVERALL PAGE: 18 OF 29
TOTAL NO. OF SHEETS

INVERTING BUFFER WITH X-1 GAIN



	Mon Mar 27 17:58:27 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

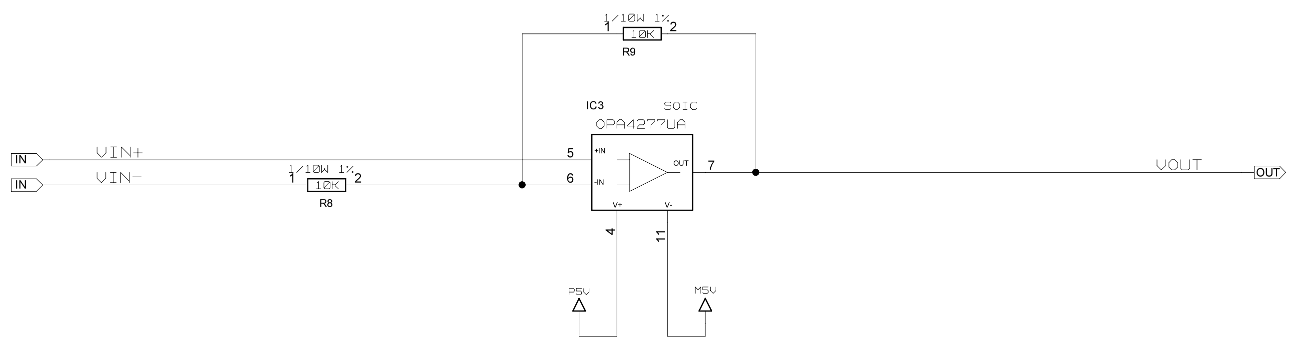
TITLE fmc_tlu_v1_lib

MODULE: pc023a_vthresh_buffer

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 19 OF 29
TOTAL NO. OF SHEETS

INVERTING BUFFER WITH X-1 GAIN



	Mon Mar 27 17:58:27 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

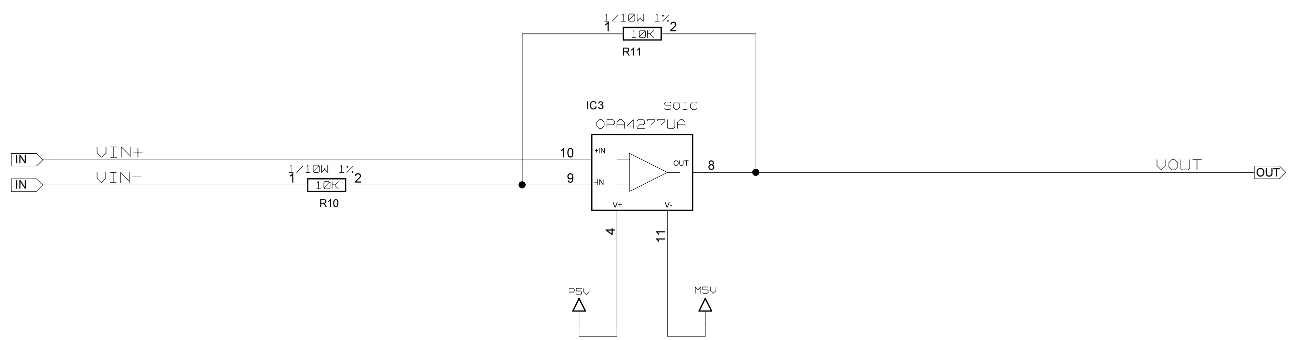
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: pc023a_vthresh_buffer

A2

INVERTING BUFFER WITH X-1 GAIN



	Mon Mar 27 17:58:27 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

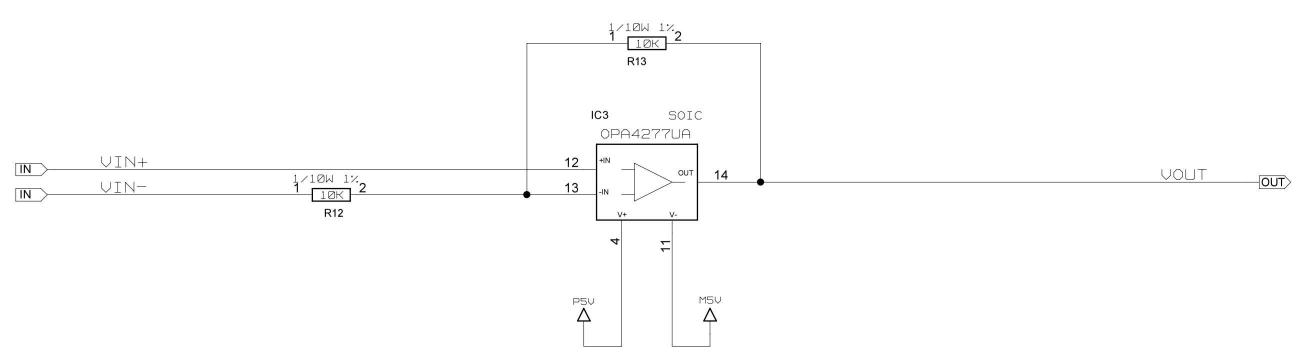
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: pc023a_vthresh_buffer

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 21 OF 29
TOTAL NO. OF SHEETS



	Mon Mar 27 17:58:27 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

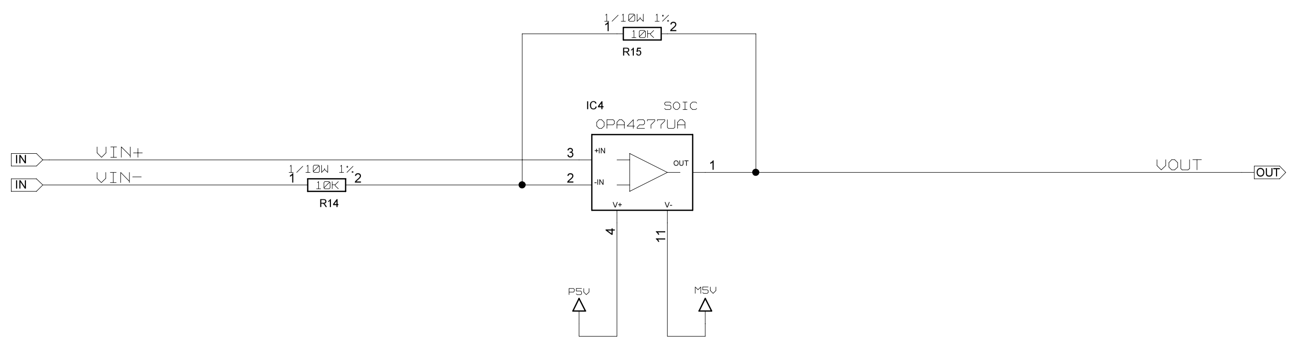
MODULE: pc023a_vthresh_buffer

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 22 OF 29
TOTAL NO. OF SHEETS

A2

INVERTING BUFFER WITH X-1 GAIN



	Mon Mar 27 17:58:27 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

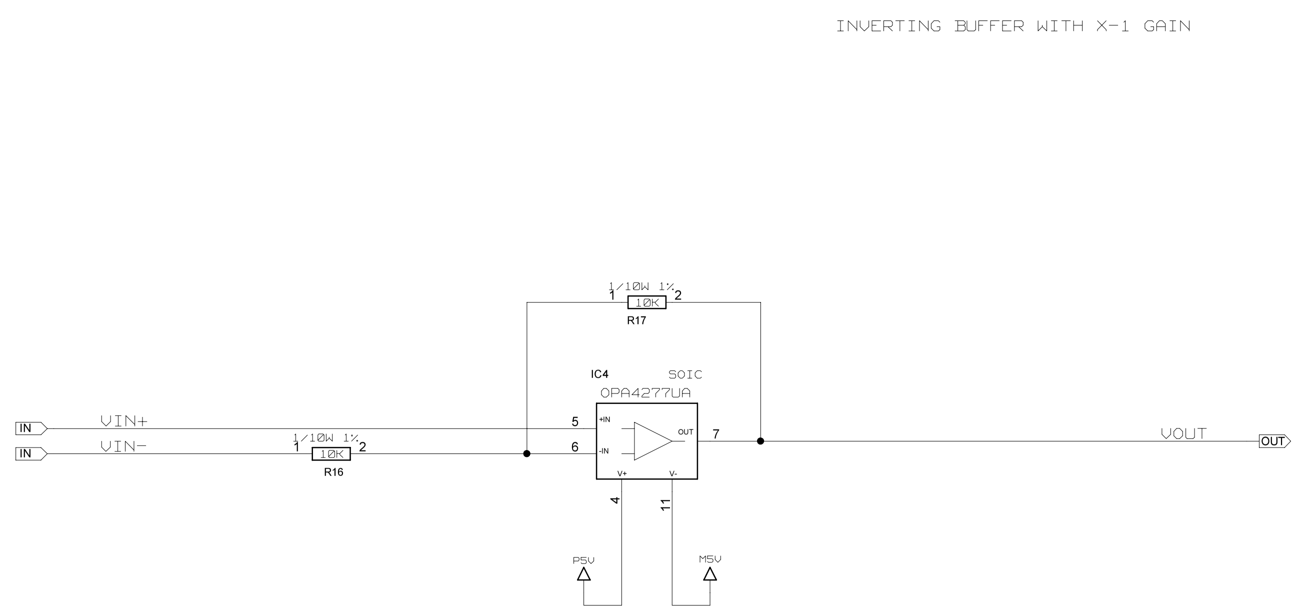
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: pc023a_vthresh_buffer

A2

A2



	Mon Mar 27 17:58:27 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

Ⓒ UOB-HEP 20

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

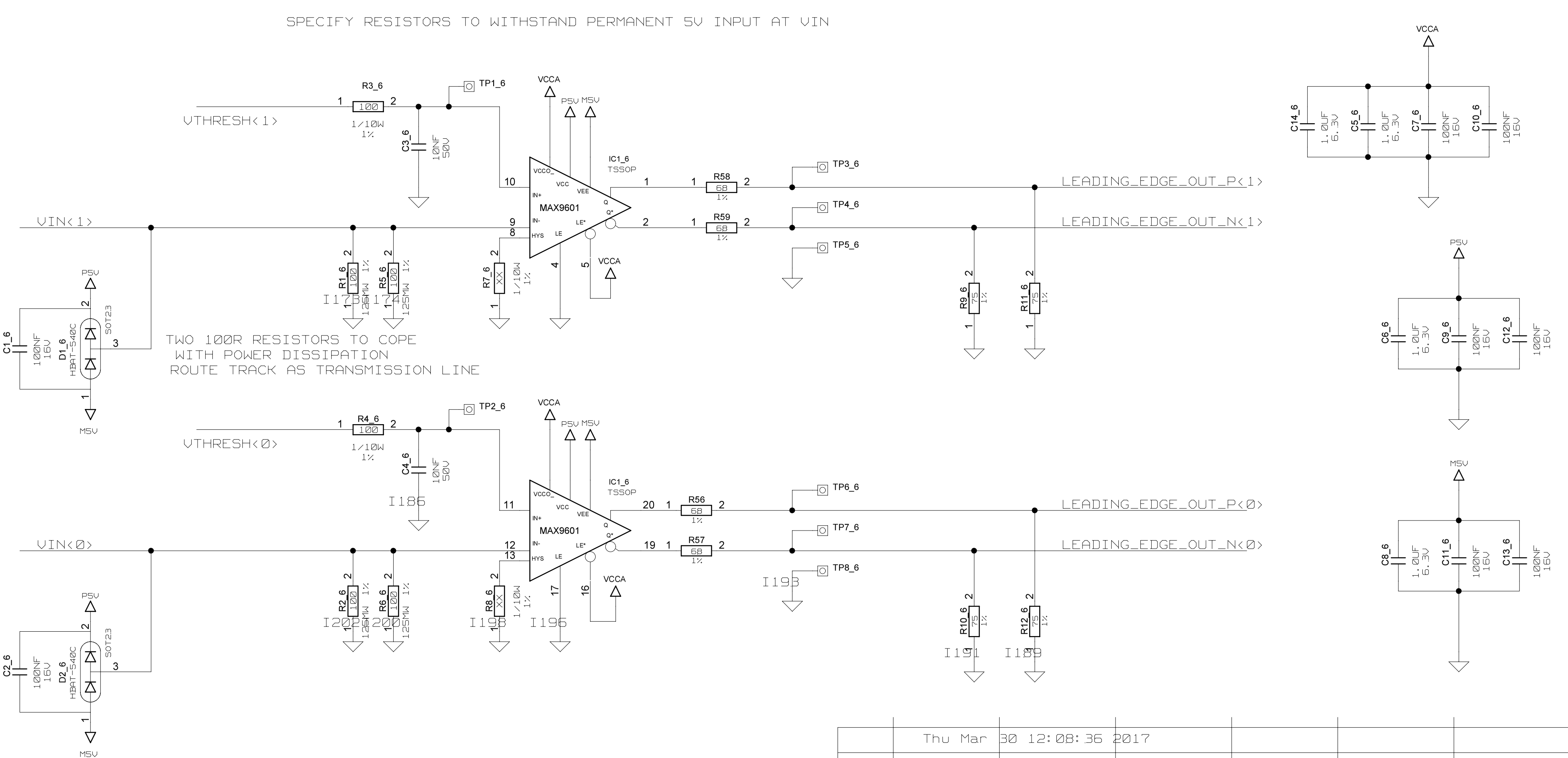
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: pc023a_vthresh_buffer

A2

MODULE PAGE: 1 OF 1
OVERALL PAGE: 24 OF 29
TOTAL NO. OF SHEETS



	Thu Mar 30 12:08:36 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20 12

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

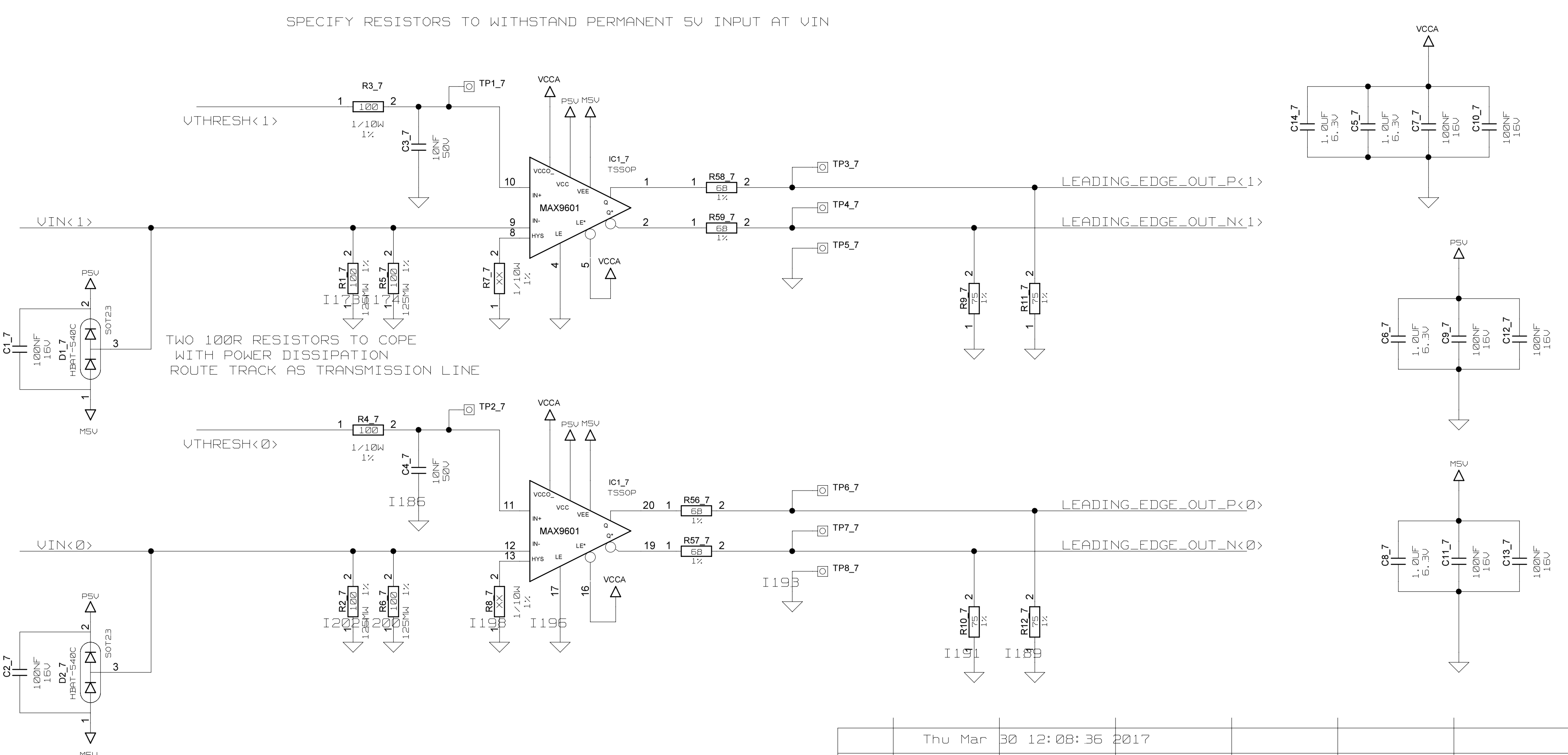
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

MODULE: fmc_tlu_threshold_discriminator_dual

A2

TOTAL NO. OF SHEETS



	Thu Mar 30 12:08:36 2017					
ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

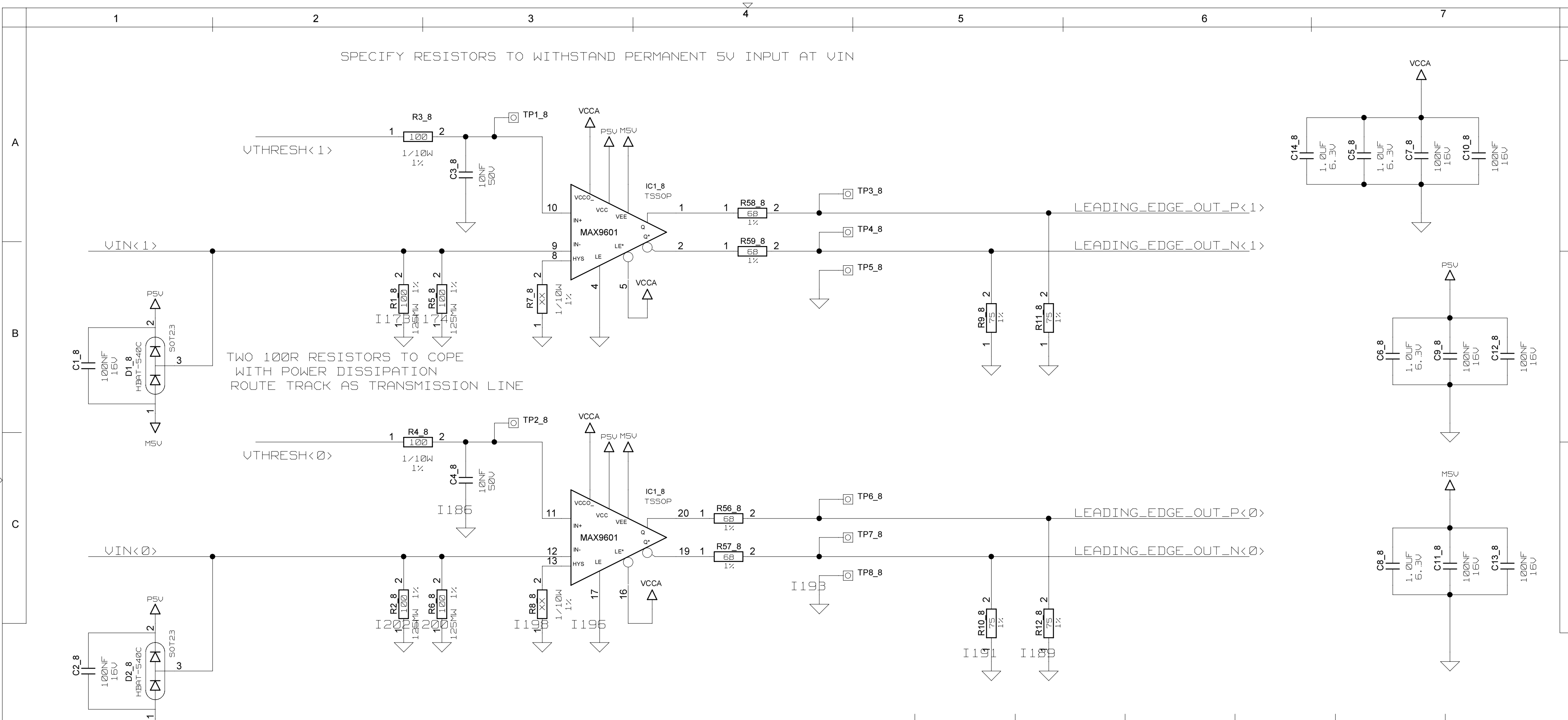
MODULE: fmc_tlu_threshold_discriminator_dual

A2

MODULE PAGE: 1 OF 1

OVERALL PAGE: 26 OF 29

TOTAL NO. OF SHEETS



Thu Mar 30 12:08:36 2017

ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

© UOB-HEP 20 12

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP

H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib

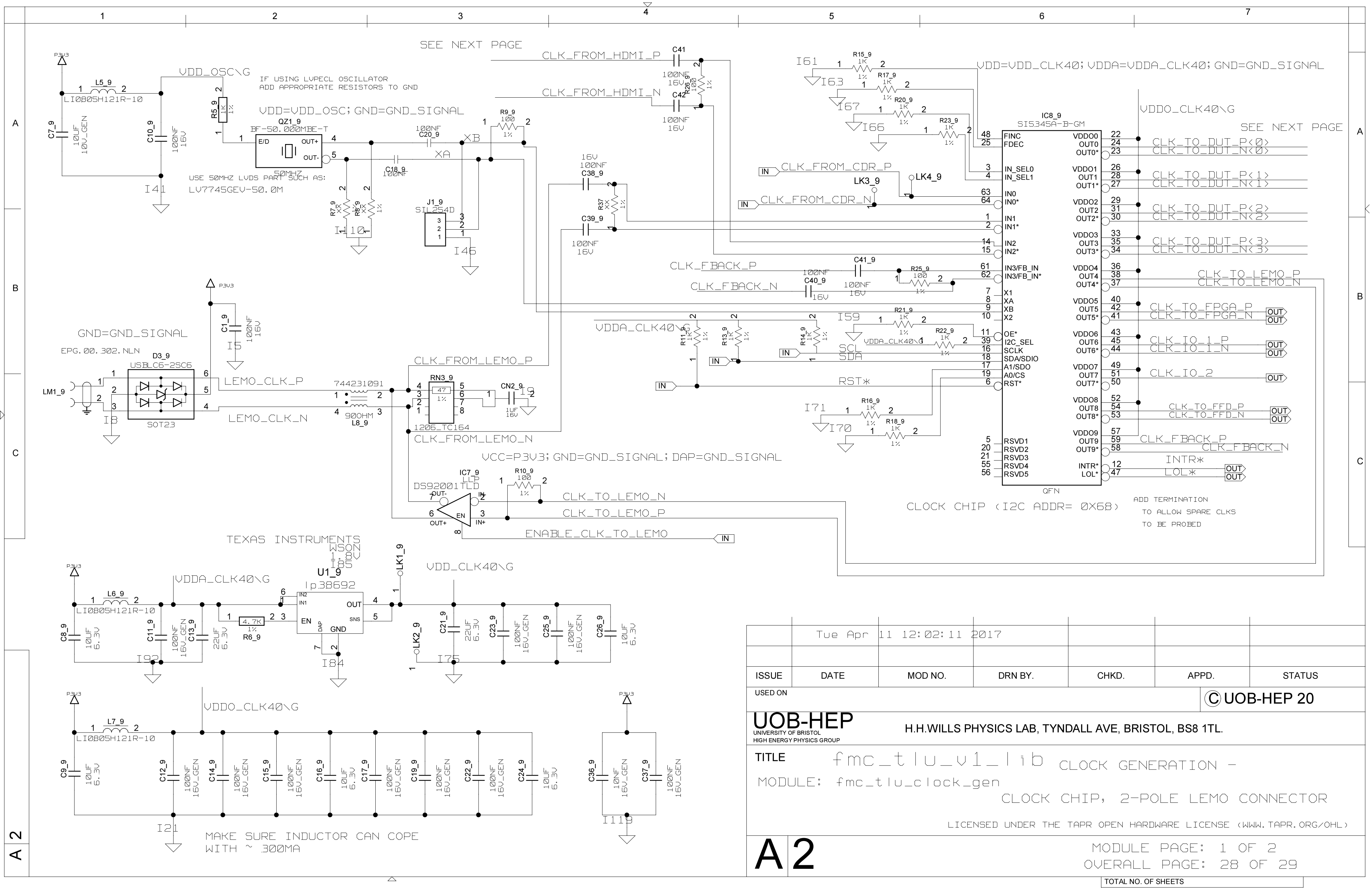
MODULE: fmc_tlu_threshold_discriminator_dual

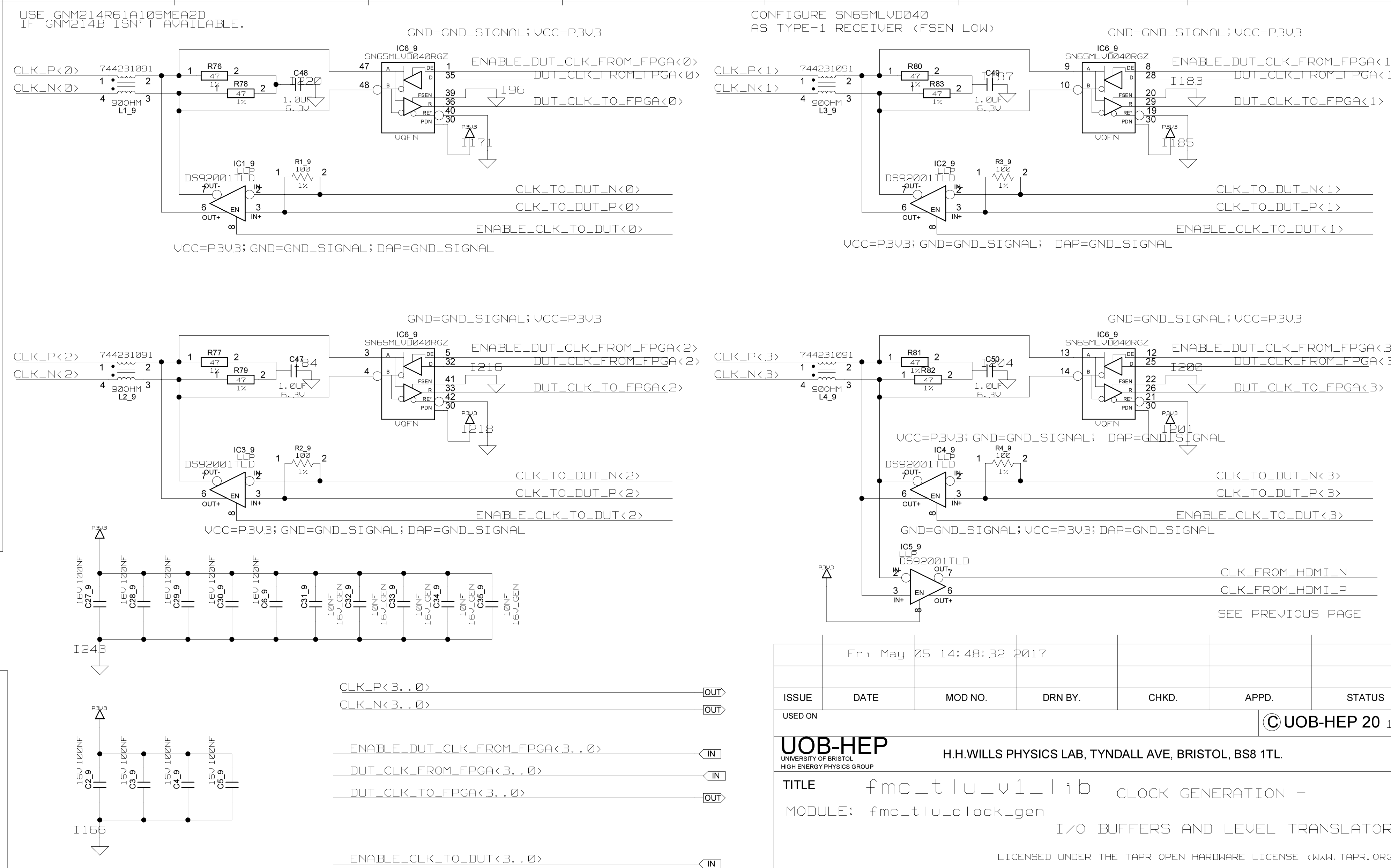
A2

MODULE PAGE: 1 OF 1

OVERALL PAGE: 27 OF 29

TOTAL NO. OF SHEETS


SEE NEXT PAGE
CLK_FROM_HDMI_P
CLK_FROM_HDMI_N
VDD_OSC\G
IF USING LVPECL OSCILLATOR ADD APPROPRIATE RESISTORS TO GND
VDD=VDD_OSC; GND=GND_SIGNAL
QZ1_9
BF-50.000MBE-T
50MHZ
USE 50MHZ LVDS PART SUCH AS:
LV7745GEV-50.0M
VDD=VDD_CLK40; VDDA=VDDA_CLK40; GND=GND_SIGNAL
VDDO_CLK40\G
SEE NEXT PAGE
IC8_9
SI5345A-B-GM
CLK_FROM_CDR_P
CLK_FROM_CDR_N
CLK_FBACK_P
CLK_FBACK_N
VDDA_CLK40\G
SCL
SDA
RST*
CLK_TO_DUT_P<0>
CLK_TO_DUT_N<0>
CLK_TO_DUT_P<1>
CLK_TO_DUT_N<1>
CLK_TO_DUT_P<2>
CLK_TO_DUT_N<2>
CLK_TO_DUT_P<3>
CLK_TO_DUT_N<3>
CLK_TO_LEMO_P
CLK_TO_LEMO_N
CLK_TO_FPGA_P
CLK_TO_FPGA_N
CLK_IO_1_P
CLK_IO_1_N
CLK_IO_2
CLK_TO_FFD_P
CLK_TO_FFD_N
CLK_FBACK_P
CLK_FBACK_N
INTR*
LOL*
ADD TERMINATION TO ALLOW SPARE CLKS TO BE PROBED
CLOCK CHIP (I2C ADDR= 0X68)
GND=GND_SIGNAL
EPG.00.302.NLN
USBLC6-2SC6
SOT23
LEMO_CLK_P
LEMO_CLK_N
744231091
90OHM
CLK_FROM_LEMO_P
CLK_FROM_LEMO_N
RN3_9
1206_TC164
VCC=P3V3; GND=GND_SIGNAL; DAP=GND_SIGNAL
IC7_9
DS92001TLD
CLK_TO_LEMO_N
CLK_TO_LEMO_P
ENABLE_CLK_TO_LEMO
TEXAS INSTRUMENTS
WSON
1.8V
U1_9
lp38692
VDDA_CLK40\G
VDD_CLK40\G
VDDO_CLK40\G
MAKE SURE INDUCTOR CAN COPE WITH ~ 300MA
Tue Apr 11 12:02:11 2017
ISSUE
DATE
MOD NO.
DRN BY.
CHKD.
APPD.
STATUS
USED ON
©UOB-HEP 20
UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.
TITLE
fmc_tlu_v1_lib CLOCK GENERATION -
MODULE: fmc_tlu_clock_gen
CLOCK CHIP, 2-POLE LEMO CONNECTOR
LICENSED UNDER THE TAPR OPEN HARDWARE LICENSE (WWW.TAPR.ORG/OHL)
A2
MODULE PAGE: 1 OF 2
OVERALL PAGE: 28 OF 29
TOTAL NO. OF SHEETS



Fri May 05 14:48:32 2017

ISSUE	DATE	MOD NO.	DRN BY.	CHKD.	APPD.	STATUS

USED ON

UOB-HEP
UNIVERSITY OF BRISTOL
HIGH ENERGY PHYSICS GROUP
H.H.WILLS PHYSICS LAB, TYNDALL AVE, BRISTOL, BS8 1TL.

TITLE fmc_tlu_v1_lib CLOCK GENERATION - I/O BUFFERS AND LEVEL TRANSLATORS

MODULE: fmc_tlu_clock_gen

A2