Warsaw University of Technology

FACULTY OF
ELECTRONICS AND INFORMATION TECHNOLOGY

The Institute of Electronic Systems

Master's diploma thesis

in the field of study Electronics
and specialisation Microsystems and Electronic Systems

Controller module for real-time quantum applications

Paweł Piotr Kulik
student record book number 252842

thesis supervisor
Grzegorz Henryk Kasprowicz, PhD

WARSAW 2019

Controller module for real-time quantum applications

Summary

In this master thesis project, the "Kasli" module was designed. It is used for real-time control of equipment used in quantum physics experiments (i.a. analog-to-digital and digital-to-analog converters, power supplies, signal generators). Kasli was built using printed circuit board (PCB) technology and houses an FPGA module which controls up to 12 connected devices (extensions), a jitter attenuator with clock distribution circuit which synchronizes all extensions and SFP connectors used for communication with user's computer or connection of other secondary Kasli modules. Many Kasli boards can be connected together in a tree topology when synchronized control of more than 12 extensions is required. Jitter attenuation circuit was measured to output a high-quality clock with RMS jitter below 400 fs in 10 Hz - 40 MHz range when provided with low-jitter input clock. With no reference signal connected RMS jitter of Kasli clock outputs was measured to be around 1 ps, depending on crystal oscillator performance which varies between measured boards. Phase noise of jitter attenuator circuit was checked with different power supply scenarios. Kasli was designed to be used with an ARTIQ (Advanced Real-Time Infrastructure for Quantum physics) control system which allows to control modules compatible with Kasli. Second revision of the board is currently in production and is used in many laboratories. Board schematics are licensed under CERN Open Hardware License.

Keywords: real-time control system, FPGA, ion trap, ARTIQ, Sinara, open hardware

Moduł sterujący do zastosowań kwantowych czasu rzeczywistego

Streszczenie

W ramach pracy zaprojektowany został układ „Kasli”, sterujący w czasie rzeczywistym aparaturą wykorzystywaną do eksperymentów fizyki kwantowej (np. układami przetworników analogowo-cyfrowych, cyfrowo-analogowych, zasilaczami, generatorami sygnałów itp.). Kasli został wykonany w technologii obwodu drukowanego, na którym został umieszczony m.in. układ FPGA, sterujący maksymalnie 12 modułami rozszerzeń, układ tłumiący jitter i dystrybuujący sygnał zegarowy, który synchronizuje rozszerzenia oraz złącza SFP wykorzystywane do komunikacji z komputerem użytkownika lub do podłączenia podrzędnych modułów Kasli. Wiele modułów Kasli można połączyć ze sobą w topologii drzewa, w przypadku gdy wymagane jest sterowanie więcej niż 12 rozszerzeniami. Pomiary wykazały, że układ tłumiący jitter jest w stanie wygenerować sygnał zegarowy o jitterze RMS poniżej 400 fs, kiedy podłączony jest wzorcowy sygnał o niskim jitterze. W przypadku braku sygnału wzorcowego zmierzono jitter RMS sygnału zegarowego na poziomie około 1 ps, zależnym od szumów oscylatora kwarcowego na zbadanych układach Kasli. W pracy zbadano również wpływ sposobu zasilania na szumy fazowe układów zegarowych zamontowanych na tym module. Kasli został zaprojektowany do pracy z systemem kontrolnym ARTIQ (Advanced Real-Time Infrastructure for Quantum physics, pol. Zaawansowana Infrastruktura Czasu Rzeczywistego dla Fizyki Kwantowej), który pozwala sterować urządzeniami kompatybilnymi z układem Kasli. Druga wersja modułu jest obecnie produkowana i jest wykorzystywana w wielu laboratoriach na świecie. Schematy układu Kasli są udostępnione na licencji CERN Open Hardware (CERN OHL).

Słowa kluczowe: system sterujący czasu rzeczywistego, system kontrolny, FPGA, pułapka jonowa, ARTIQ, Sinara, open hardware

Acknowledgements

Foremost I would like to thank my thesis supervisor, Grzegorz Kasprowicz, for giving me an opportunity to make a design for the scientific community and for his patience and unending will to share his knowledge.

Thanks to the ARTIQ community, particularly Robert Jördens and Thomas Harty, their feedback on my board design probably reduced the number of prototype revisions tenfold. Also thanks to Thomas Harty for borrowing me his Kasli v1.0 which enabled me to compare performance between the first and the second revision of the board. These measurements also would not be possible if helpful folks from the Microwave Circuits and Instrumentation Division didn't give me access to their apparatus.

I would also like to thank ARTIQ developers, particularly Robert Jördens and Sébastien Bourdeauducq from M-Labs. They developed software and gateware that made Kasli a useful device. Additionally, their support with ARTIQ saved me a lot of time and I greatly appreciate it.

Finally, I'd like to thank people who spent their time to check my thesis: Wojciech Zabołotny, Krzysztof Poźniak, and Joseph Britton as their feedback was crucial for me and my thesis.

załącznik nr 10 do zarządzenia
nr 46 /2016 Rektora PW

Politechnika Warszawska
Warsaw University of Technology

Warszawa, 28.02.2019
miejscowość i data
place and date

Paweł Piotr Kulik
imię i nazwisko studenta
name and surname of the student
252842
numer albumu
student record book number
Elektronika
kierunek studiów
field of study

OŚWIADCZENIE

DECLARATION

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.
Under the penalty of perjury, I hereby certify that I wrote my diploma thesis on my own, under the guidance of the thesis supervisor.

Jednocześnie oświadczam, że:
I also declare that:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- *this diploma thesis does not constitute infringement of copyright following the act of 4 February 1994 on copyright and related rights (Journal of Acts of 2006 no. 90, item 631 with further amendments) or personal rights protected under the civil law,*

- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- *the diploma thesis does not contain data or information acquired in an illegal way,*

- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- *the diploma thesis has never been the basis of any other official proceedings leading to the award of diplomas or professional degrees,*

- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- *all information included in the diploma thesis, derived from printed and electronic sources, has been documented with relevant references in the literature section,*

- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.
- *I am aware of the regulations at Warsaw University of Technology on management of copyright and related rights, industrial property rights and commercialisation.*

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

I certify that the content of the printed version of the diploma thesis, the content of the electronic version of the diploma thesis (on a CD) and the content of the diploma thesis in the Archive of Diploma Theses (APD module) of the USOS system are identical.

.....
czytelny podpis studenta
legible signature of the student

Contents

1. Introduction

Each year semiconductor companies encounter more problems in upholding famous Moore's law which predicts, that the number of transistors in integrated circuits doubles every two years. Increased spendings are needed to maintain this prediction and, effectively, to double the available speed of computation. That's because scaling down transistors is becoming more and more difficult with many quantum effects, that are not present in larger processes [52].

Meanwhile, advancements in physics and technology brought us closer to building a quantum computer big enough that it cannot be simulated classically. Quantum computing is a fundamentally different approach to computation which is theorized to be more powerful than its classical counterpart in certain tasks or even enable us to do computation that was otherwise impossible or highly impractical to do [9, 54]. For example, in [48] and [41] researches demonstrated molecular calculations of water and hydrogen molecules using quantum computers. Techniques used in this research could be used to solve more complicated chemical problems, that are impossible for current supercomputers, if more powerful quantum computers could be used. Quantum algorithms invented by Peter Shor and Lov Grover could break some of currently used encryption schemes [57, 29]. Because of this, there are post-quantum (but still classical) cryptographic schemes in the works, that will be resistant to quantum computers [10]. Quantum computers will cause some encryption types to become obsolete, but will also give a way to build a secure quantum encryption schemes [63, 28].

But quantum physics experiments are not only about quantum computers. These experiments also aim to build atomic clocks with better accuracy [38]. Cold-atom interferometry could be used for inertial navigation systems, which maintain Global Positioning System (GPS) precision and can be used in environments without access to satellites' signal, for example underwater [39]. Quantum physics experiments also aim to measure some of the fundamental properties of the universe [55]. Standard model prediction are tested this way [53]. Quantum simulation is also developed [8, 15, 56], proposed by Richard Feynman. It is an idea to simulate quantum systems, which are too difficult to simulate with classical computers, using another quantum system [22, 56].

It is clear that quantum computing and all technologies, that emerge during its development, hold great promise of significant progress in many areas of our lives.

1.1. Quantum computers

Quantum computers are built from a quantum version of classical bits – qubits. With each entangled qubit in a quantum computer, the complexity of a quantum state doubles, thus sufficiently big quantum computer is not possible to be simulated using a classical computer[1]. Operations on quantum computer involve manipulating amplitudes of each state, so the result we aim to achieve has the highest amplitude. In the end, we can only measure one result, but complex interactions in between are the power behind quantum computers [52].

Difficulties in scaling computer to a higher number of qubits include maintaining superposition and entanglement. Overall "performance" of quantum gates, memories etc. can be described using measures such as fidelity and coherence time. Fidelity is a measure of the overlap between an ideal quantum state and the state produced by a gate, memory or communication channel when

[1] E.g. 333 qubits correspond to $2^{333} \approx 1.7 \cdot 10^{100}$ (1.7 googol) possible basis states!

it's performing an operation on a quantum system [51]. Coherence time is a measure of how much time after preparing initial states do you have to perform an operation or series of operations on your qubits before they are substantially degraded by interaction with the environment. Both are critical in the construction of a universal quantum computer, which is the long term goal of many quantum computer experiments. That computer would be able to simulate an arbitrary physical system, including other quantum computers, which would enable it to run any algorithm. This is an analogy to Universal Turing Machine.

According to DiVincenzo, construction of a quantum computer requires seven criteria to be fulfilled with some restriction placed on potential qubits [18]. Currently, among other candidates for qubits are superconducting circuits and trapped ions. In ion trap quantum computing, information is stored in position of ion's electron on different energy levels (also called states). Superconducting circuits store information, depending on their type, as number of superconducting electrons in case of charge qubits, direction of current in case of flux qubits, or as oscillatory states in case of phase qubits. Each method of storing information in qubits has its own set of advantages and weaknesses. This project was done in collaboration with physicists working on trapped ions, so further sections will concentrate on electronics in context of trapped ions used as qubits [36].

1.2. Electronics used in ion trap experiments

In experiments where ions are used as qubits, various electronic circuits are needed i.a. for [49]:

- controlling lasers,
- controlling Acousto-Optic Modulators (AOMs), that change frequency of the lasers,
- controlling power supplies,
- generating RF frequency which traps ions,
- generating DC[2] voltage at the trap,
- reading ion states via photoemission.

This wide range of tasks often means, that laboratories have many, often incompatible, sets of hardware (like shown in figure 1.1) and controlling it in a unified way becomes a daunting task.

1.3. Examples of existing control systems and hardware families

The problem mentioned in the previous section is not a new one and there are already control systems, that could be used for controlling hardware during an experiment involving ion traps. Some of them are listed below however, systems that offer "full stack" solution are not described. These are systems, that offer a solution from qubit implementation to running quantum algorithms, like systems developed or in development by IBM [33], Google [27] or D-Wave [20].

[2] All stable voltages are called DC voltages in this thesis.

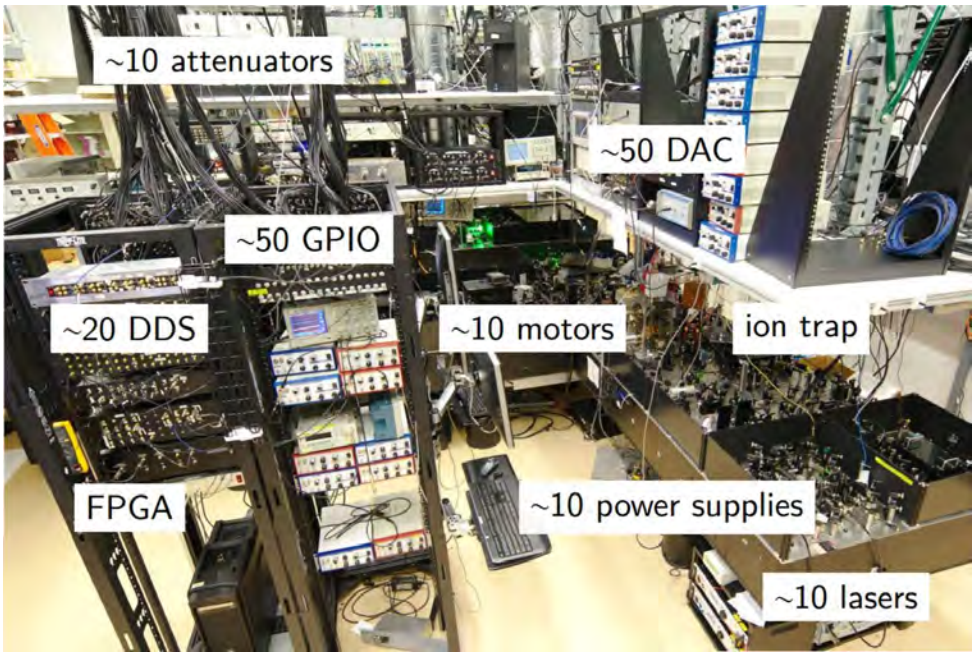


Figure 1.1: Example laboratory with used devices outlined [12]

Easy-phi system

One of the existing solutions is easy-phi. Easy-phi is an open electronics standard for scientific instrumentation. It was developed on University of Geneva by Group of Applied Physics in Optics for quantum optics experiments and authors claim, that it is suitable for other areas in physics. Whole standard is open-source and schematics and source code are available on GitHub [31]. Hardware is licensed under CERN Open Hardware License (CERN OHL). Easy-phi rack structure is shown in figure 1.2.

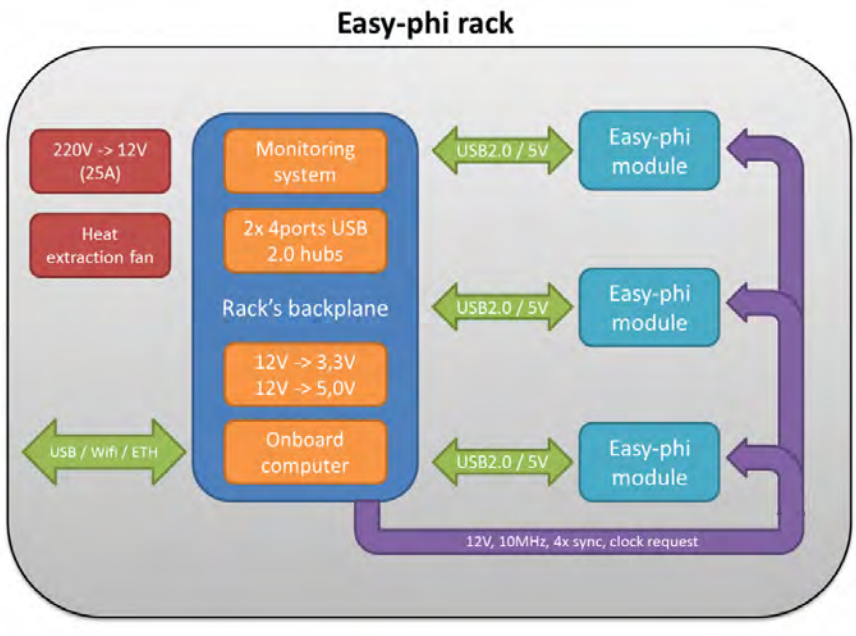


Figure 1.2: Easy-phi rack structure [30]

Each rack has an onboard computer, which enumerates all modules over a Universal Serial Bus (USB) 2.0 connection on the backplane. Modules are supplied from backplane with 5 V and 12 V power. Due to low throughput and high latency of USB 2.0 any high-speed or high-precision signals must use Quick-locking SMA (QMA) connectors on the front panel, which are defined in standard's description.

There are three main problems with this standard:

1. Achieving sub 1 ns synchronization is impossible without adapting existing modules and developing new ones.
2. Using front-panel connectors for high-speed data stream or high precision signals could quickly result in lots of tangled cables – which is something we want to avoid when using racks.
3. Easy-phi is not maintained since 2015 [31].

Labber Quantum

Labber is a control system designed by Scandinavian company Lab Control Scandinavia. The company partnered with Keysight Technologies to provide a development environment for experiments. Quantum Labber provides an "interface between quantum algorithms and qubit hardware" [40], however it seems to lack support for low latency branching or closed-loop feedback. These capabilities could be provided by Keysight software [37], as shown in block schematic in figure 1.3.

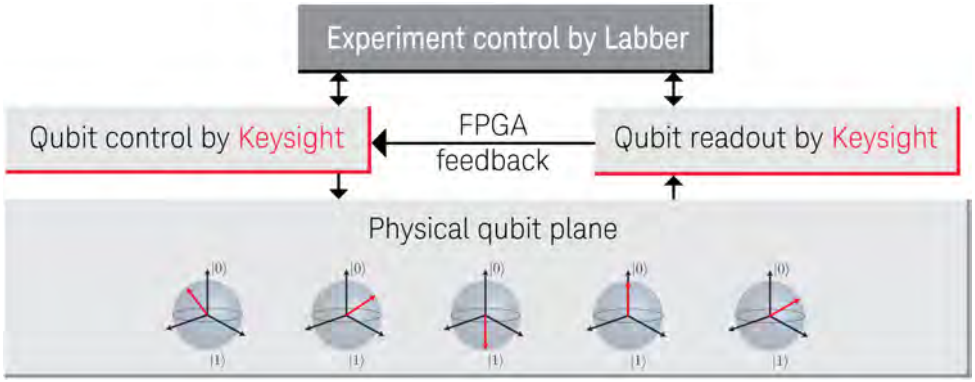


Figure 1.3: Quantum Labber with Keysight [37]

LabView

LabView, made by National Instruments (NI), is commonly used in many ion trap experiments (example case study is in [17]). It provides intuitive graphical programming language and support from NI. It can be interfaced with existing hardware or with PXI (PCI eXtensions for Instrumentation) or CompactRIO (Compact Reconfigurable I/O) systems proprietary to NI. There are several synchronization schemes available (see [50]). They provide synchronization from 1 ms in case of Precision Time Protocol (PTP) down to 20 ns, when using special synchronization modules [19].

LabView main disadvantage for users is relying on proprietary software, with no way to open code except for LabView Integrated Development Environment (IDE). This means that paid license is required even for reviewing old code. Another issue is high latency of programs written in LabView.

Zurich Instruments Quantum Computing Control System

Zurich Instruments offers Quantum Computing Control System (QCCS). This system consists of 4 main components (also shown in figure 1.4):

- Arbitrary Waveform Generator (HDAWG),
- Quantum Analyzer, capable of reading up to 10 superconducting qubits (UHFQA),
- System Controller, capable of controlling up to 18 instruments (PQSC),
- LabOne control software and Application Programming Interfaces (APIs) for several programming languages.

This system seems to be concentrating on needs of experiments using superconducting qubits with "instrumentation required for quantum computers of up to 100 qubits" [78].

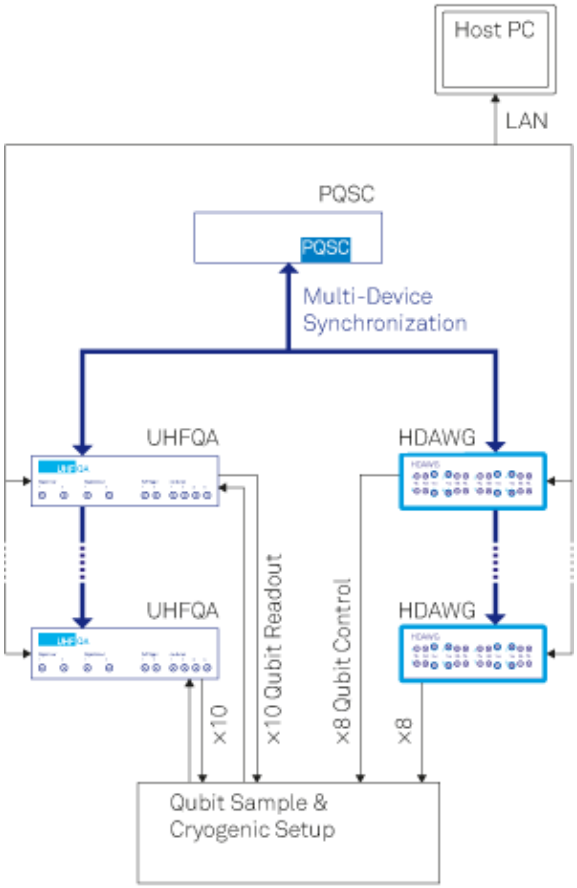


Figure 1.4: QCCS block schematic [78]

In-house systems

An example of an in-house solution is a digital servo, that was developed in National Institute of Standards and Technology (NIST). It provides two analog input channels and three analog output channels and is optimized for feedback control of lasers in atomic, molecular and optical physics experiments. The minimum latency of the servo is 320 ns [42].

Its hardware includes two Printed Circuit Boards (PCBs). On the first one, there is a Spartan-6 Field-Programmable Gate Array (FPGA), which is a controller part of the servo, and the second one provides input and output channels. The firmware of the servo, consisting of control logic and digital filters is written in Verilog. There is also a software part, running on user's PC, which provides the ability to set feedback transfer functions. Optionally this software also allows user to monitor the system. This software is not required for continuous operation, as once set, the device can operate autonomously [42].

It is a fully open source solution with designs and source code available online, however, it is limited in scope to only servo function. Its modular design allows the user to upgrade and adapt the controller or analog part as needed.

ARTIQ

Advanced Real-Time Infrastructure for Quantum physics (ARTIQ) was developed by M-Labs[3] in collaboration with NIST. It is a system of choice in groups that support this thesis project. ARTIQ provides high timing accuracy with resolution of 1 ns and sub-microsecond latency [12]. This is achieved by running code on an FPGA device. ARTIQ also features high-level programming language, based on Python, which is used to describe experiments. Time-critical parts of those experiments (called *kernels*) are executed on the FPGA and results are sent back to the user's computer, where data processing and other non-realtime tasks can take place. ARTIQ also includes a Graphical User Interface (GUI) (example shown in figure 1.5), a scheduler for experiments and databases for experiment results.

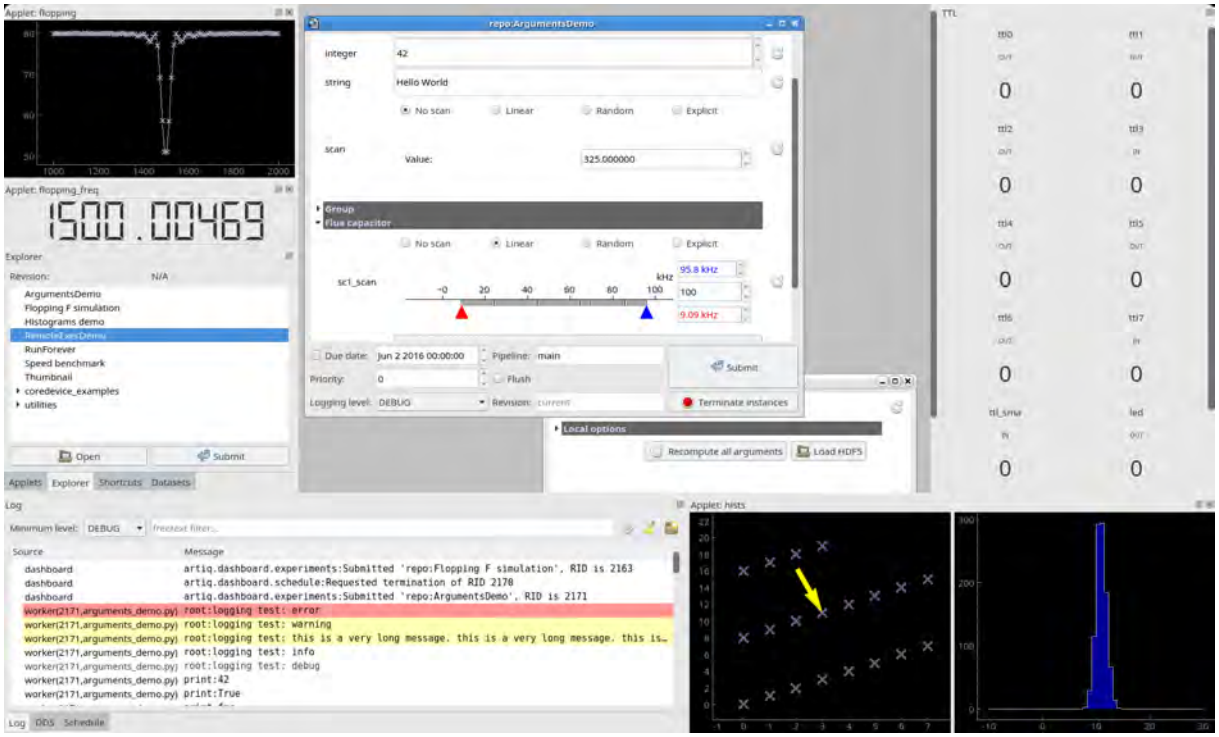


Figure 1.5: ARTIQ example GUI [12]

[3]Developer company working on ARTIQ/Sinara project, responsible for high-level ARTIQ python and low-level FPGA configuration.

ARTIQ is an open-source project [13]. While initiated by NIST, other organizations have also contributed or funded ARTIQ development, including University of Oxford and University of Maryland.

Sinara

Sinara is a hardware family designed to be used with ARTIQ. It aims to provide modular, flexible and well-tested hardware for use in experiments. All hardware was designed at the Institute of Electronic Systems. Currently, most of the hardware was developed with ion trap experiments in mind.

The motivation for creating this hardware family was to reduce duplication of work by different laboratories, which have similar needs but tend to create their own hardware. With all designs being licensed under CERN OHL, reproducing experiments conducted by laboratories using this hardware should be much easier. Open Hardware License also allows laboratories to modify hardware to their needs but requires publication of modified designs [16]. All Sinara designs are available on-line [60]. Open-source solutions can also be cheaper than current commercially available equipment.

An example of system architecture is shown in figure 1.6. It consists of user's computer, two MicroTCA crates and a Kasli box. MicroTCA crates house high-performance cards, which generate precise high frequency signals, while Kasli box generates lower frequency signals.

User's computer is running an ARTIQ program, which sends commands to the Metlino card in MicroTCA crate via Ethernet. Metlino is a MicroTCA Carrier Hub (MCH), which controls Advanced Mezzanine Cards (AMCs) inserted into the crate. Metlino is a source of time for any other card connected to it. Satellite crates may be connected to root crate in a daisy chain. Sayma is an Arbitrary Waveform Generator controlled by Metlino. It provides 8 channels of 1.2 GS/s 16-bit Digital-to-Analog Converters (DACs) and 8 channels of 125 MS/s 16-bit Analog-to-Digital Converters (ADCs). Sayma can also work in standalone mode, which is used mainly for development. Kasli box is another satellite crate.

All master and satellite cards are connected with Distributed Real-Time Input/Output (DRTIO) links, which transfer time and data between them. "The link is a high-speed duplex serial line operating at 1 Gbps or more, over copper or optical fiber." [11]. The link can also be established over MicroTCA backplane. DRTIO enables all real-time channels (e.g. TTL) on satellites to be synchronized and controlled by a master device. It also provides means of sending data between devices and user's PC, for example, measurement data. The protocol is optimized to minimize latency. Many devices can be connected in a tree topology, and each device deeper in the tree has increased latency. That latency is always constant and can be measured and compensated in ARTIQ kernel scripts [11].

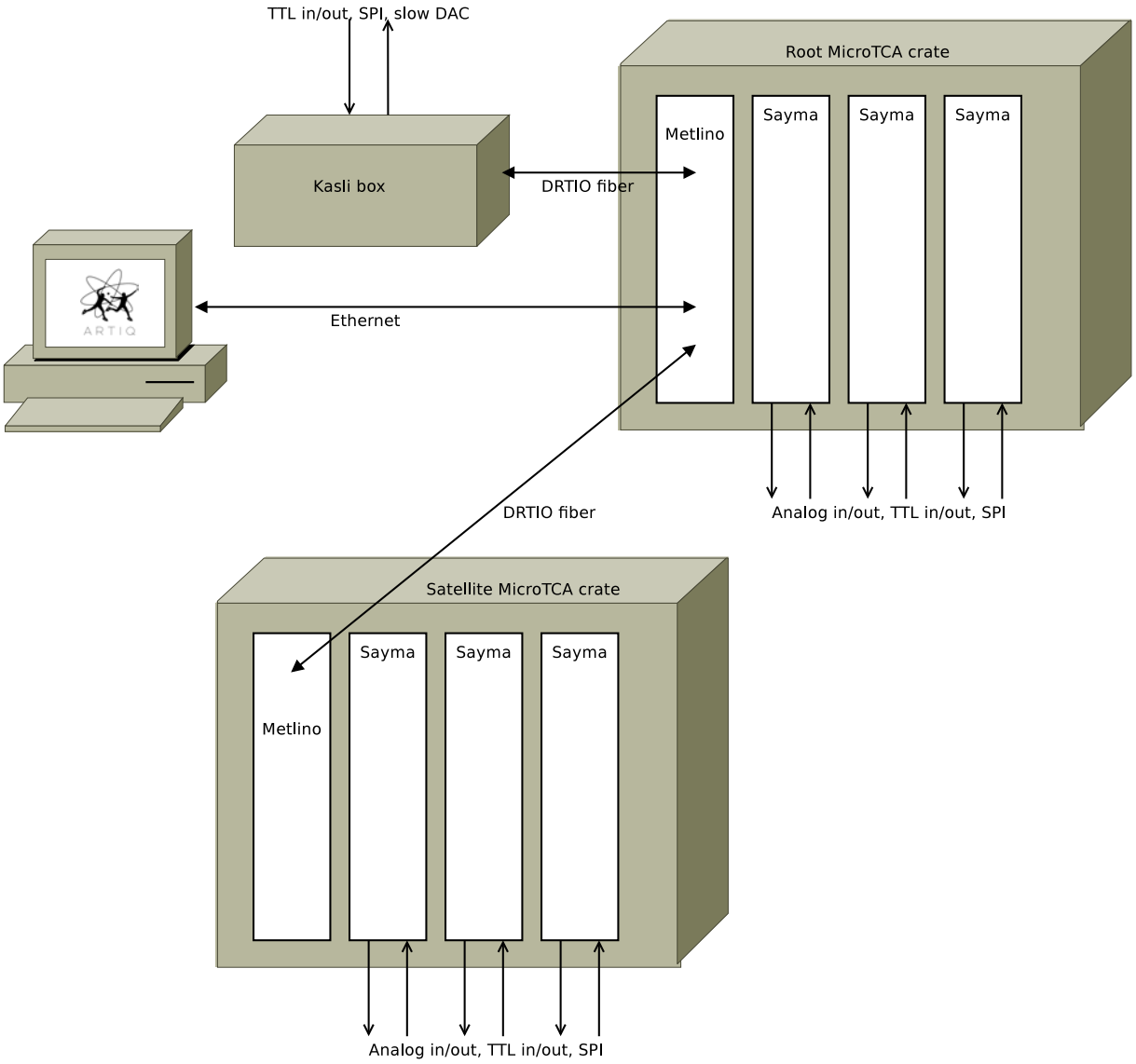


Figure 1.6: System block schematic

2. Motivation and goal

2.1. Motivation

The motivation for this project was to improve the Sinara hardware family as it aims to provide high-quality open-source equipment for ion trap experiments. Focus was on improving the low-cost solution that was planned – the Kasli box. It should provide a much cheaper alternative for less demanding tasks, like controlling power supplies or cooling systems, setting frequency of the lasers or setting precise DC voltage levels which are some of the tasks encountered in ion trap laboratories.

A modular architecture was planned for the Kasli box with several Eurocard Extension Modules (EEMs), each designed to perform a single function. Examples of functions include ADC, DAC, Direct Digital Synthesizer (DDS) or Digital Input/Output module (DIO). EEMs are designed to rely on a controller to perform computations or data signalling.

The idea of the Kasli box was to put all EEMs in a separate 3U crate to provide power and cooling. At the beginning stages of the Sinara project EEMs could only be controlled by KC705 (Kintex FPGA development kit) or Sayma boards, but not directly. The connection to these controllers (see figure 2.1) requires Very-High-Density Cable Interconnect (VHDCI) module, which is mounted on the FPGA Mezzanine Card (FMC) connector of Sayma or KC705 boards. VHDCI cable is connected to VHDCI carrier (shown in figure 2.2) and finally carrier splits control lines to individual EEMs connected to it with ribbon cables. Sayma or KC705 are then able to control lines connected to EEMs.

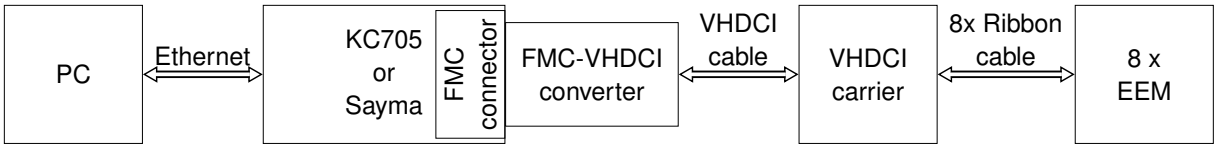


Figure 2.1: Block schematic of first EEM connections

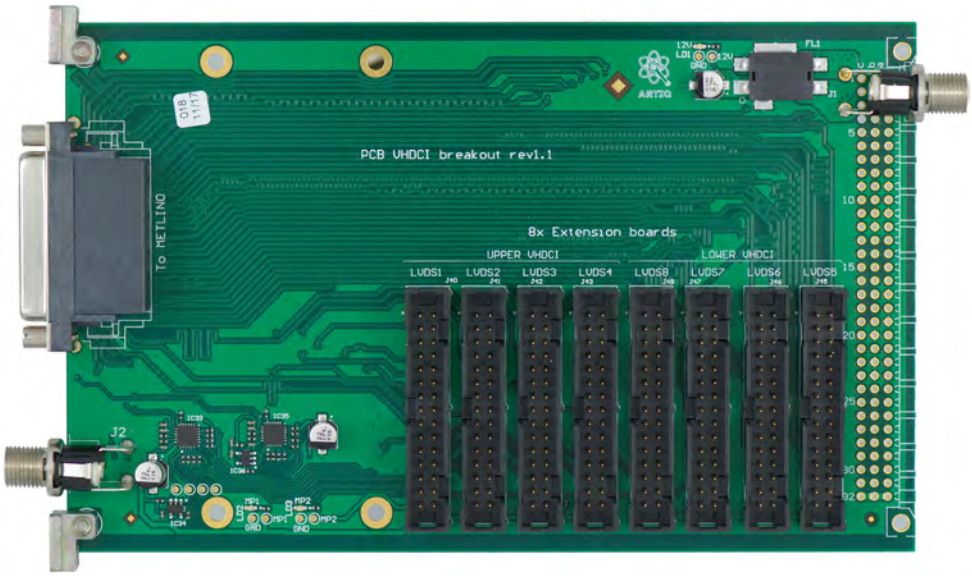


Figure 2.2: VHDCI carrier

Even though available EEMs were then limited to Sampler ADC and various DIO modules, more EEMs were planned and need for simplifying this connection chain was apparent. Kasli box was

meant to provide a cheaper alternative to high-end boards and requiring expensive Sayma board or KC705 development kit was a problem, especially since some parts of the experiment do not need to be synchronized to the rest of the experiment.

2.2. Goal

The goal of this thesis project was to design a standalone EEM carrier board, which would enable users to fully utilize EEMs in a low-cost modular system without the overhead of current expensive controller options. It should simplify connection scheme showed in figure 2.1. This EEM carrier should be able to control all EEMs directly, replacing KC705 or Sayma and all modules required to interface EEMs to them. The final connection diagram should look like the one presented in figure 2.3. It was decided by a funder, that this carrier shall be named Kasli, after the Russian town.

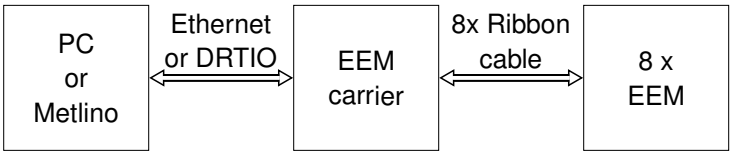


Figure 2.3: Block schematic of planned EEM connections

2.3. Technical assumptions

First and foremost, Kasli board should be compatible with ARTIQ. It should allow to connect at least as many EEMs as VHDCI carrier and control them with the same timing resolution of 1 ns as MicroTCA ecosystem and KC705. This means that the clock source the Kasli is using should have low jitter. This device should be able to operate in stand-alone mode, independently of other parts of the experiment. It should also be able to synchronize to other devices if there's need to. In case, when there are more devices to control than can be connected to one Kasli, there should be a way to connect many carriers together to control more EEMs, synchronized to the same source. Some EEMs may require an external reference clock signal, that cannot be supplied with a ribbon cable. In that case, Kasli should be able to provide that reference clock with another type of cable.

3. Module architecture and component choices

3.1. Module architecture

Core components of the board which fulfill requirements presented in section 2.3 are:

- FPGA with Random Access Memory (RAM) for compatibility with ARTIQ and hard real-time interactions with hardware,
- as many Small Form-factor Pluggable (SFP) connectors as possible for daisy-chaining multiple boards,
- jitter attenuator circuit to provide low-jitter reference clock to extensions,
- as many EEM connectors as possible for extensions.

In addition to those components, there are additional features that were requested by potential users during the planning stage:

- reference clock input,
- USB connector with Universal Asynchronous Receiver-Transmitter (UART) and JTAG (Joint Test Action Group) for debugging purposes and upgrading firmware,
- clock fanout to a supply clock signal to EEMs,
- 3U form factor to put Kasli with other EEMs in a single crate,
- power supplied from external 12 V power supply.

Proposed block schematic is shown in figure 3.1. In the middle of the schematic, there is an FPGA module, which interfaces with RAM. SFP 1 connector is connected to the FPGA. This connector serves as the main connection to Ethernet or to other master DRTIO card. Additional SFP connectors can be used to connect slave devices. Having more than one slave SFP connector will decrease depth of DRTIO tree, decreasing latency. SFP data lines should be connected to FPGA multi-gigabit transceivers. USB lines should be connected to some kind of converter, which would allow translation to UART and JTAG. These lines should then be connected to the FPGA. There should also be a dedicated JTAG connector (not shown on schematic), compatible with Xilinx Platform Cable, for initial development on the FPGA.

The FPGA can recover a clock from DRTIO input data stream, though this clock can have higher jitter than it is desired. A jitter cleaner circuit should be able to output a low-jitter clock, using clock recovered from DRTIO as a reference. This low-jitter clock (marked as "cleaned clock" on block schematics) could then be sent to EEMs through dedicated connectors and also used to drive logic used to control EEMs. To perform as an EEM carrier, Kasli should have as many Insulation-Displacement Connector (IDC) connectors as can be connected to the FPGA (same connectors as in VHDCI carrier), each with 8 Low-Voltage Differential Signalling (LVDS) lines and one I^2C bus.

It is not likely to fit more than 8 EEM connectors and all other components on a 3U board with dimensions of VHDCI carrier. It was decided, that LVDS and I^2C signals for additional EEMs will be routed to a backplane connector (same as in VHDCI carrier). That way, if a backplane is defined at later stages in the project there will be 32 LVDS lines and 4 I^2C buses to use. If not, a simple adapter board can be designed to break out to 4 30-pin IDC connectors (same as on Kasli).

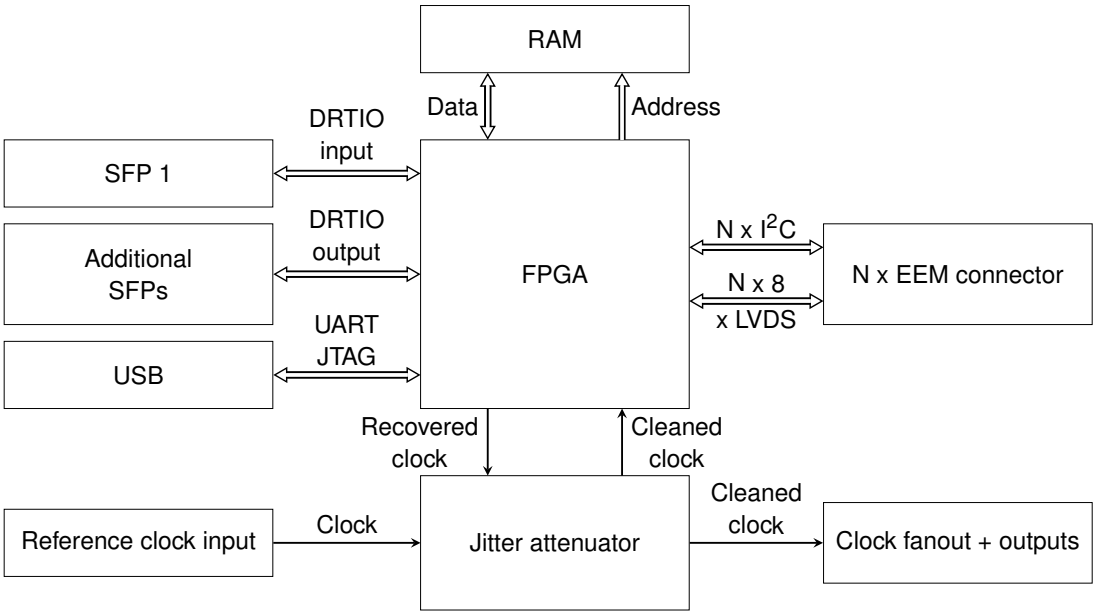


Figure 3.1: Block schematic of Kasli

3.2. EEM connector

Quoting Sinara wiki page: "EEM connectors provide a standardized means of connecting EEM peripherals to a carrier, such as Kasli, which provides power and real-time digital I/O. Connectors are 2x15 100mil pitch male pin-header (pinout below). Wiring between boards is typically done using ribbon cable (50mil wire pitch). (..) Each EEM peripheral has an Electrically Erasable Programmable Read-Only Memory (EEPROM) on its I^2C bus for identification." [60] Connector pinout is shown in table 3.1.

Table 3.1: EEM connector pinout

Function	Pin numbers
Ground	1, 4, 7, 10, 13, 16, 19, 22, 25
12 V supply	28, 29
3.3 V supply	30
I^2C	26 (SDA), 27 (SCL)
LVDS_0	2 (P), 3 (N)
LVDS_1	5 (P), 6 (N)
LVDS_2	8 (P), 9 (N)
LVDS_3	11 (P), 12 (N)
LVDS_4	14 (P), 15 (N)
LVDS_5	17 (P), 18 (N)
LVDS_6	20 (P), 21 (N)
LVDS_7	23 (P), 24 (N)

3.3. Component choices

Many of the components were used in other boards from the Sinara hardware family. Some component choices may not be optimal in the scope of just Kasli board (e.g. they can have higher parameters than necessary and higher price), but using the same component in many boards across the

whole hardware family has its benefits. In many cases, software development time is significantly reduced because the code is already written for components used elsewhere. Also, the performance of these components is already known and there are less unknown factors. It's also cheaper for a manufacturer to stock a higher number of the same components as opposed to a lower number of different components.

3.3.1. FPGA

ARTIQ was already running on KC705 development kit and Sayma also has a Xilinx FPGA. Choosing Xilinx' parts cuts down on development time since most of the primitives (like special buffers) are already included and used in ARTIQ code. Also, our laboratory has special pricing available from Xilinx.

Artix-7 was chosen as the cheapest (at the time of design of the module) FPGA family from Xilinx. M-labs decided to use Artix-7 100T variant with -2 speed grade after estimating resource usage and project requirements. -2 speed grade has a maximum switching frequency of F_{MAX} = 550 MHz. Since Kasli is meant to work in laboratories and most of them have controlled temperature (to minimize thermal drift of used equipment) any temperature range of the ones available are suitable. Commercial (C) temperature range is preferred, as it is the cheapest option, however manufacturer may decide to mount part in other temperature range in case it has better availability.

Artix-7 100T variant can be purchased in following packages: CSG324, FTG256, FGG484, FGG676[4]. First two packages were rejected because they do not have any GTP transceivers[5], which are needed for high-speed communication with DRTIO and Ethernet. FGG676 provides only 15 more I/O pins compared to FGG484 in 100T FPGA variant. FGG676 package also provides 4 more GTP channels than FGG484, however, limited place on the board (mostly taken by IDC connectors) does not allow to put more than 4 SFP connectors on the front panel, so 4 additional GTP channels cannot be used either way. Ultimately 20% increase in cost and worse availability led to choosing FGG484 package. The exact part number of the FPGA is XC7A100T-2FGG484C.

The FGG484 package also gives an upgrade path to a 200T variant, should 100T have insufficient resources and allows to choose cheaper variants (down to 15T) if there is a need for a less capable and less costly carrier. These variants are pin-compatible with the chosen variant, so replacing the FPGA comes down to just replacing one Bill of Materials (BoM) line [71].

3.3.2. RAM

The Synchronous Dynamic Random-Access Memory (SDRAM) Integrated Circuit (IC) that was used in Sayma board is no longer manufactured. Instead a similar part from Micron was chosen. It is a 4 Gbit, 16-bit memory in 96-pin BGA package, which is capable of up to 1866 Mbit/s data rate [44]. The package was chosen for compatibility with Sayma. Its part number is MT41K256M16TW-107:P. Artix-7 FPGA achieves up to 1066 Mbit/s [26], however slower SDRAM parts from Micron are no longer available in 96-ball package. This SDRAM IC is able to work with 1.35 V or 1.5 V power supply.

[4] They differ by a number of pins (denoted as a number in package name) and by spacing (called pitch) between each Ball Grid Array (BGA) ball.

[5] GTP - Xilinx' name for multi-gigabit transceivers found in Artix-7 FPGAs, capable of 6.6 Gbit/s of transfer speed.

The latter was used to keep compatibility with older revisions of this IC. This model's datasheet refers to [43] when designing with 1.5 V power supply and operating conditions were checked against this datasheet.

Reference voltage regulator from Texas Instruments was used to supply a reference voltage to the SDRAM IC and termination resistors. It is the same part as in Sayma board. It provides termination voltage of half of the SDRAM power supply. The device was connected according to an example design with small changes [62]. Reference voltage was not connected to the FPGA since its two reference voltage inputs had to be used for other SDRAM signals. Instead, an internal reference voltage in the FPGA was used.

On all address and control lines 51 Ω termination resistors were placed. SDRAM memory part used in this project provides dynamic On-Die Termination (ODT) on DQ (data) and DQS (strobe) lines, which allows controller to connect or disconnect termination depending on the direction of data flow. Artix-7 FPGA provides uncalibrated split termination [68] which is only active when data line is an output. This means that DQ and DQS lines have dynamic termination on both ICs and no additional resistors are needed.

Memory signals were connected according to the user guide published by Xilinx [77] and checked with Memory Interface Generator (MIG). This tool (pin assignment screen shown in figure 3.2) allows user to instantiate a memory controller IP core and either automatically assign signals to FPGA pins, or load pin assignment made by a user. The tool automatically selects I/O standard and appropriate termination settings. After loading user-specified pinout, MIG checks assignments against rules described in [77], allowing the user to find errors that are easy to miss otherwise.

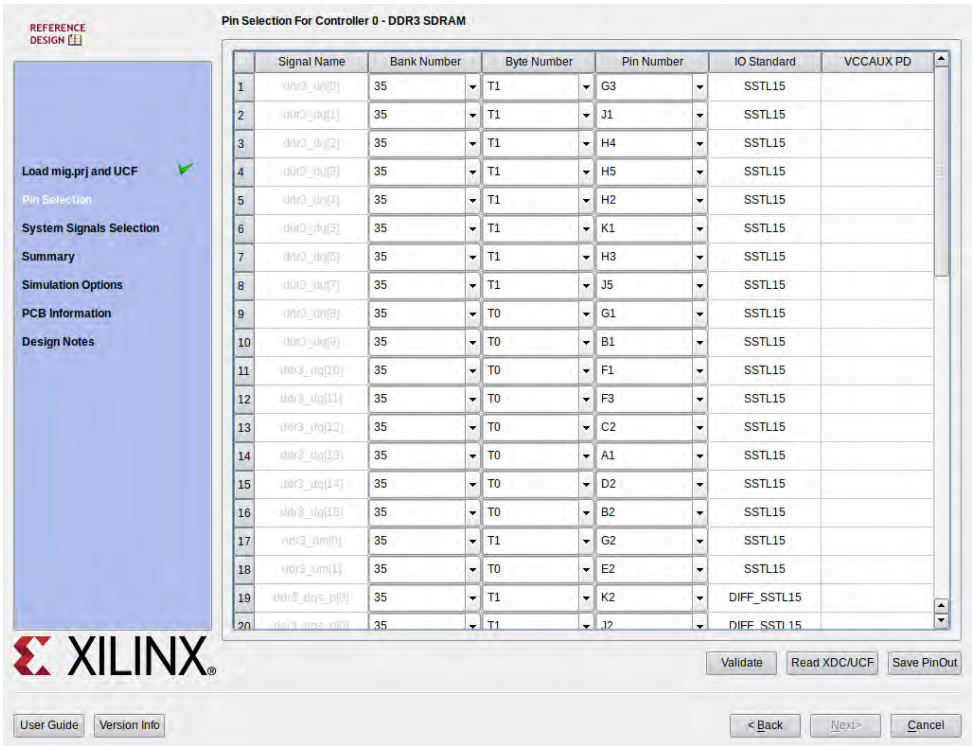


Figure 3.2: Memory Interface Generator – pin assignment

3.3.3. Jitter attenuator circuit

The jitter attenuator circuit plays a vital role in ensuring that ARTIQ is capable of providing a reliable 1 ns timing resolution without significant jitter, that would make this kind of precision useless. For that reason, jitter attenuation scheme is similar as in Sayma board, with Si5324 as the main part. Detailed block schematic is shown in figure 3.3.

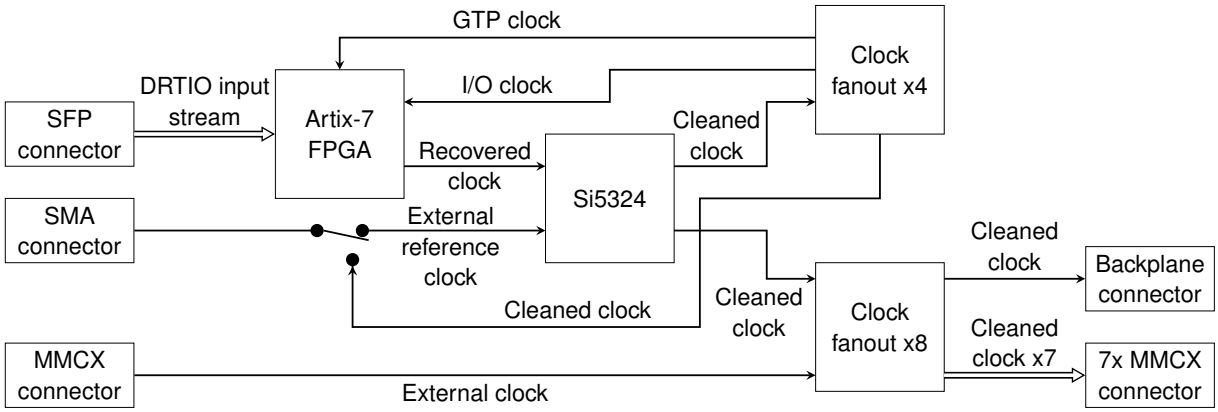


Figure 3.3: Block schematic of planned jitter attenuation circuit

Si5324 is a precision clock multiplier and jitter attenuator. It is programmable by an I^2C bus and has two clock inputs and two clock outputs. Outputs can be configured as LVDS, Low-Voltage Positive Emitter-Coupled Logic (LVPECL), Current Mode Logic (CML) or CMOS I/O standards. It is suitable for applications requiring sub 1 ps jitter [58]. It can take an input clock and output a clock with known frequency in relation to an input frequency, but much lower jitter (down to Si5324 and external oscillator noise floor).

Two clock fanouts were used, ADCLK944 (4-output fanout) and ADCLK948 (8-output fanout). These are, again, used in Sayma board. ADCLK944 is used to provide clock signals to FPGA GTP transceiver, I/O bank used to clock logic controlling EEMs and SMA connector. SMA connector can be used as an input, which provides external reference clock for Si5324, or as an output, which provides clean clock synthesized by Si5324. ADCLK948 was used to provide 7 clock signals on Micro-Miniature Coaxial (MMCX) connectors and one signal on the backplane connector. It can also be used to fan out an external clock source, from an MMCX input connector. ADCLK parts were used because they provide fanout with very low additive phase noise, which does not degrade the jitter parameters of signals synthesized by Si5324. Both fanout ICs add less than 75 fs jitter to the input signal [1, 4]. ADCLK944 and ADCLK948 both can accept AC-coupled LVDS and LVPECL signals on inputs, which Si5324 can provide.

Outputs of both fanout ICs are in LVPECL standard. Each output was biased with 200 Ω resistors, though, resistors on unused negative lines of differential pairs were set to not be placed during assembly to reduce power consumption of fanout ICs [1, 4]. GTP reference clock input has an internal level bias, so when AC-coupling this signal no further bias is needed [65]. FPGA I/O clock needs to have LVDS levels. This means, that this clock needs to be DC-biased using resistors [32].

Clocks from SMA and MMCX input connectors are single ended. Both the Si5324 and ADCLK948 parts can accept single-ended signals, but in case it's performance would be less than desired by physicists, (e.g. slew rate of single ended signals is lower than of differential signals with the same output stage and ADCLK944 and ADCLK948 performance degrades with lower slew rates [1, 4])

transformers can be mounted, that provide symmetrical output and attenuate common mode noise referenced to ground. These transformers by default are bypassed with two jumpers[6], as shown in figure 3.4. The input stage of ADCLK948 and Si5324 was by default terminated with 51 Ω resistors (built in case of ADCLK948) to ground. If transformers are mounted, then termination scheme changes to provide 100 Ω differential termination, as shown in figure 3.5.

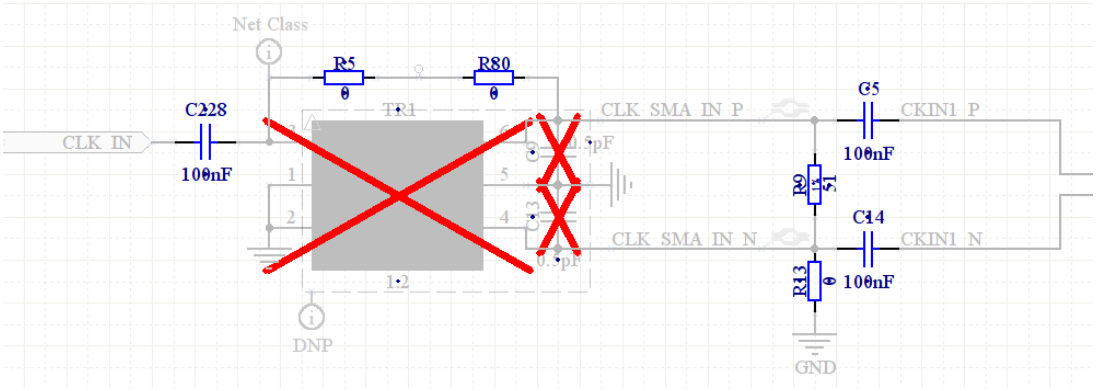


Figure 3.4: Transformer connections on Si5324 input – default connection

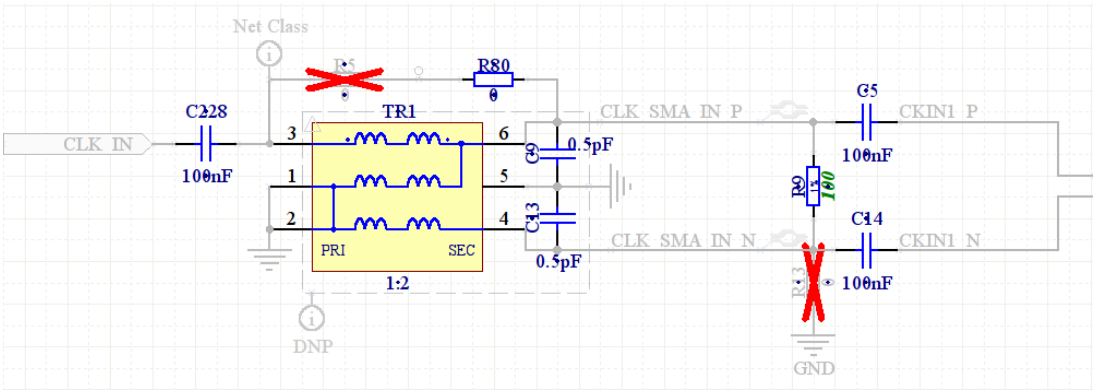


Figure 3.5: Transformer connections on Si5324 input – alternative connection

3.3.4. USB circuit

FT4232H integrated circuit was chosen for interfacing with the FPGA and other devices on Kasli via USB. It is an IC that has four 8-bit buses that can also work as a serial interface, controlled from user's PC. It is used on Sayma and scripts to use its functions are already developed. FTDI shares libraries, that allow users to control each bus in bit bang or Multi-Protocol Synchronous Serial Engine (MPSSE) mode. Open source libraries for python are also available. The IC was connected according to the device datasheet [23] with additional Electrostatic Discharge (ESD) protection diodes on USB lines.

The FT4232H IC was connected to draw power from USB. Small buck converter (TLV62565 from Texas Instruments) was used to convert from 5 V, supplied by USB, to 3.3 V required by the IC. This way the FT4232H does not draw power from the main power supply and is powered only when necessary. All FT4232H signals, that are interfacing with other circuitry on board, are connected

[6] Each jumper is close to pad to minimize stubs, that could act as antennas.

to level shifting ICs (SN74AVC2T245 and TCA9517 from Texas Instruments were used) to avoid latch-up effect, when FT4232H is not powered [34], and to translate signals to other voltages.

Channel A was connected to JTAG signals. All JTAG lines were pulled up with 10 kΩ resistors to match pull-up on the FPGA [69]. JTAG from FT4232H is disabled when Xilinx Platform Cable pulls down pin 13 of JTAG connector [73], which was connected via transistor to `output enable` pins of level shifter ICs. FT4232H is also supported by openOCD, an open source application, which can be used to program and debug JTAG peripherals including 7-Series FPGAs by Xilinx. This means that USB connection can replace Xilinx platform cable when ChipScope or Integrated Logic Analyzer (ILA) are not used in an FPGA project. These technologies are not used in ARTIQ, so no platform cable is required.

Channel B was connected to the FPGA UART via level shifting IC. Channel C was connected to the I^2C bus, also via level shifting IC designed for I^2C bus. Two pins of FT4232H were connected together. This IC uses one input and one output line alternately for bidirectional data line [24]. Channel D was connected to 10-pin connector (8 data bits + ground + power, connector is not mounted by default) in case any user needs it.

3.3.5. FPGA connections

At this point, when all major components were known, FPGA connections could be assigned and power requirements were known.

SDRAM was connected to bank 35. SDRAM was using 1.5 V power supply so this FPGA bank must also use this voltage.

EEMs are controlled by LVDS signals. In 7-series there are two types of I/O banks: High Range (HR) and High Performance (HP). Artix-7 FPGA has only HR banks, which support LVDS only when I/O bank is supplied with 2.5 V. This FPGA has internal differential termination required by LVDS receivers [68].

JTAG lines have internal pull-ups so no external pull-up resistors are needed [69]. It was decided that 4x Serial Peripheral Interface (SPI) memory flash will be used to store FPGA bitstream. This required 4 pins from bank 14, which had to be powered from 2.5 V power supply. This meant, that memory part was needed, that could interface with 2.5 V logic. 128 Mbit flash memory was used. This was more than minimum memory size recommended by Xilinx, however, the memory will also be used to store ARTIQ firmware. The part number of the used memory IC is S25FL128S. It is supported by Vivado Design Suite[7] [75]. This also means, that bank 0 needed to be powered from 2.5 V power supply to match signal level forced by using LVDS signals on bank 14.

100T variant of Artix-7 FPGA in FGG484 package has 285 available I/O pins, out of which 274 can be used as differential I/O (137 differential pairs) [66]. Bank 35 was used by SDRAM and it has 50 I/O out of which 48 can be differential pins. This left 235 I/O pins with 226 differential pins. Single ended signals, that must be routed to FPGA, are shown in table 3.2 (SDRAM signals are not included). When subtracted from available pins this left 197 I/O pins out of which 196 differential (assuming non-differential pins are used first). 196 differential pins gives 98 differential pairs. 2 differential pairs were used by clock output recovered from GTP and by clock input cleaned by Si5324. This left 96

[7] Vivado is an IDE for Xilinx FPGAs. It allows synthesising VHDL or Verilog source code into a bitstream, which can be used by the FPGA to perform desired functions.

differential pins. Each EEM requires 8 LVDS signals. This meant that 96/8 = 12 EEMs could be connected to the remaining pins. One single ended pin was left unused in addition to any unused pins in bank 35.

Table 3.2: Single-ended signals to connect to the FPGA

Description	Count
SFP control signals x 3	$8 \cdot 3 = 24$
UART	2
I^2C	2
FPGA config + SPIx4	6
50 MHz clock	1
USB presence	1
ADCLK948 clock selection	1
User LED	1
Sum	**38**

Voltage requirements based on planned I/O usage are shown in table 3.3 [72].

Table 3.3: Voltages required by Artix-7 FPGA

Name	Description	Voltage [V]
V_{CCINT}	Internal supply voltage	1.00
V_{CCBRAM}	Block RAM supply voltage	1.00
$V_{MGTAVCC}$	Analog supply voltage for the GTP transmitter and receiver circuits	1.00
$V_{MGTAVTT}$	Analog supply voltage for the GTP transmitter and receiver termination circuits	1.20
V_{CCAUX}	Auxiliary supply voltage	1.80
V_{CCADC}	XADC supply relative to GNDADC	1.80
$V_{CCO_{35}}$	Supply voltage for HR I/O bank 35	1.50
V_{CCO_0}, $V_{CCO_{13}}$, $V_{CCO_{14}}$ $V_{CCO_{15}}$, $V_{CCO_{16}}$, $V_{CCO_{34}}$	Supply voltage for HR I/O banks 0, 13, 14, 15, 16 and 34	2.50

To work properly the FPGA power supplies have to be decoupled. In table 3.4 required quantities and values of capacitors are shown, based on Xilinx user guide [67]. Some 100 nF capacitors were also added so that at least every two BGA balls used for power supply share a dedicated capacitor.

Table 3.4: Required PCB capacitor quantities

Capacitor value	330 µF	100 µF	47 µF	4.7 µF	0.47 µF
V_{CCINT}	1	0	0	6	8
V_{CCBRAM}	0	1	0	0	2
V_{CCAUX}	0	0	1	3	5
V_{CCO_0}	0	0	1	0	0
$V_{CCO_{13}}$, $V_{CCO_{14}}$ $V_{CCO_{15}}$, $V_{CCO_{16}}$, $V_{CCO_{34}}$ (per bank)	0	0	1	2	4

3.3.6. Other components

- Level shifters that were used for JTAG, I^2C bus and UART were also used to translate SFP control lines.
- Each EEM connector has its own I^2C bus. This prevents conflicts when identical modules are connected. Each SFP connector and Si5324 also have separate buses. Two TCA9548 ICs were used as I^2C bus switches. I^2C topology is shown in figure 3.6.

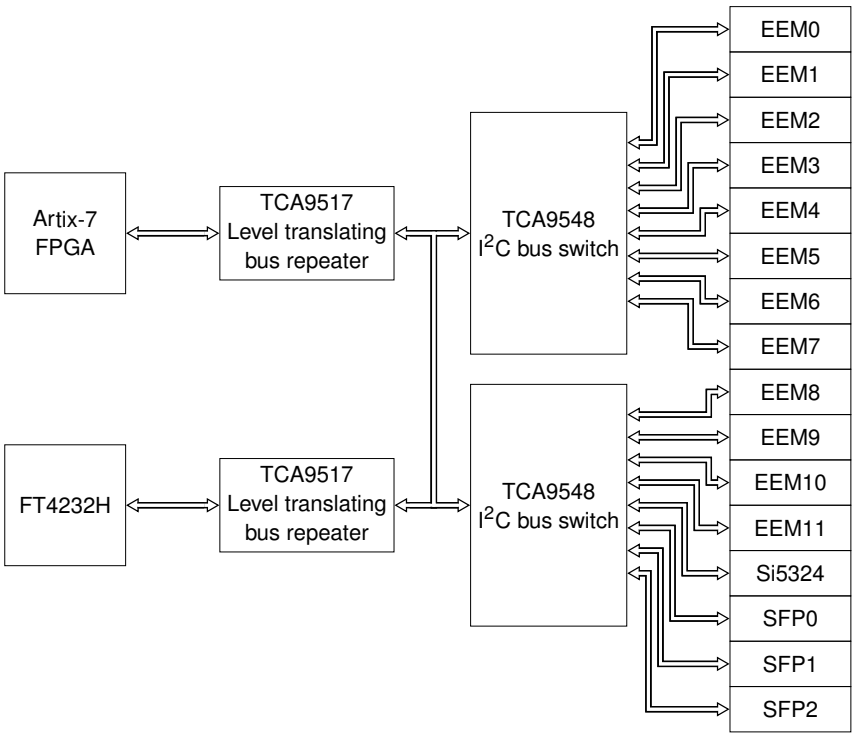


Figure 3.6: Block schematic of I^2C buses in Kasli

- Each EEM has 3.3 V power supply line, that is meant for identification via I^2C bus. Current draw from this supply was limited by a 125 mA reversible fuse, shared between groups of four EEMs.
- 50 MHz oscillator for FPGA I/O and 125 MHz LVDS oscillator for GTP transceivers were added.
- Backplane connector and IDC connectors for EEMs are the same as in VHDCI carrier.

3.3.7. Power supply

The required voltages are listed in table 3.5 along with the maximum current draw of components estimated on values provided in their datasheets. Artix-7 power draw was estimated using power estimator spreadsheet [76]. Based on this estimation of current draw ADP5052 IC was chosen as a 4-channel buck converter with a LDO. Its 2 channels are capable of sourcing 4 A (which is needed for 3.3 V and 1.0 V rails) and 2 channels are capable of sourcing 1.2 A. Channel 5 is a 200 mA LDO, which is suitable for 1.2 V rail for the GTP transceiver termination [5]. Buck designer spreadsheet was used to calculate required values of elements connected to the IC [6].

Table 3.5: Current draw of ICs used in the project

Description	Part Number	Current draw on voltage rail [mA]					
		3.3 V	**2.5 V**	**1.8 V**	**1.5 V**	**1.2 V**	**1.0 V**
LED SFP0-2	HSME-C120x3	27					
LED USER1, Power Good, FPGA done, P3V3	KPH-1608CGCKx4	12					
Clock multiplier/ jitter attenuator	Si5324	279					
Clock fanout buffer	ADCLK944B	138					
Clock fanout buffer	ADCLK948B	330					
Extensions, powered by Kasli (I^2C power)x12		240					
DDR3 Termination regulator	TPS51200	40					
SFP+ 6.25 Gbit x3		900					
Level-Shifting I^2C Repeater	TCA9517 x4	20					
I^2C Switch	TCA9548 x2	80					
Level-Shifting Bus Transceiver	SN74AVC2T245 x9	216					
50 MHz Oscillator	636L3I050M00000	25					
FPGA memory	S25FL128SAGBHIA	100					
RAM x16, rev E	MT41K256M16				274		
Artix-7 FPGA	XC7A100T		767	971	155	167	3838
Low-Dropout regulator (LDO) 1.2 V	ADP5052		167				
1.8 V buck	TLV62565	595					
Sum		**3169**	**934**	**971**	**429**	**167**	**3838**

Input values for the Buck Designer spreadsheet are shown in table 3.6. Inductors rated for currents equal to or larger than I_{OUT} were used for each channel.

Table 3.6: Values entered in Buck Designer spreadsheet

Channel number	V_{IN} [V]	V_{OUT} [V]	I_{OUT} [A]	V_{ripple}
1	12.0	3.3	3.5	2%
2	12.0	1.0	4.0	1%
3	12.0	2.5	1.1	2%
4	12.0	1.5	0.8	2%
5 (LDO)	2.5	1.2	0.2	–

ADP5052 can provide power for 5 out of 6 required voltage rails. It was decided to use TLV62565 buck for 1.8 V since it capable of sourcing up to 1.5 A and is already used in this project. Providing power rail with higher current draw from an external buck offloads the ADP5052 and evens out heat distribution. An additional 300 mA LDO for 1.2 V was added, but is not mounted by default. This LDO can be populated if current draw on 1.2 V rail is higher than 200 mA.

Power sequencing on this board is required by the Artix-7 FPGA as (aside from level-shifter ICs) it is the only part that uses multiple voltage rails. Power sequencing minimizes current draw during power-on. Xilinx recommends that power rails are ramped in order: V_{CCINT}, V_{CCBRAM}, V_{CCAUX}, V_{CCO}. Supply rails that have identical voltage can be ramped simultaneously. Power sequence required by GTP transceiver: V_{CCINT} $V_{MGTAVCC}$ $V_{MGTAVTT}$. V_{CCINT} and $V_{MGTAVCC}$ can be ramped simultaneously. In this project there is a power sequence of 12.0 V (from external power supply),

1.0 V, 2.5 V, 1.5 V & 1.2 V, 3.3 V, 1.8 V. GTP power sequence is maintained. While V_{CCAUX} is ramped after V_{CCO}, voltage differences of less than 2.625 V between these rails are acceptable [72].

Power to this board has to be supplied from an external power supply. Same Electromagnetic interference (EMI) filter and power jack were used as in VHDCI carrier. Power jack has a barrel lock on it to secure the connection. Physicists requested that the power could be supplied from the front or from the back of the board. This power jack is rated up to 5 A [61]. Additionally, a Zener diode was used to protect Kasli in case higher voltage was supplied than can be handled by ADP5052 or EEMs.

Table 3.7 shows a summary of maximum current draw and calculations of power draw. These values are worst-case scenario. In the sum of power draw 1.8 V and 1.2 V power were not added, because they are already included in 3.3 V and 2.5 V power draw, respectively. Total 12 V current draw from the external power supply is 20.92 W/12 V = 1.74 A. Power jack is rated up to 5 A, which leaves 5 A – 1.74 A = 3.26 A for EEM power. EEMs are supplied with 12 V via ribbon cable.

Table 3.7: Maximum power calculation

Power rail	**3.3 V**	**2.5 V**	**1.8 V**	**1.5 V**	**1.2 V**	**1.0 V**
Sum of current draw [A]	3.169	0.934	0.971	0.429	0.167	3.838
Buck converter limit [A]	3.5	1.1	1.0	0.8	0.2	4.0
Power [W]	10.46	2.34	1.75	0.64	0.20	3.84
Efficiency	0.88	0.78	0.89	0.76	1	0.74
Power [W]	11.88	3.00	1.96	0.85	0.20	5.19
Sum of power draw [W]			**20.92**			

3.4. Final block schematic

Final block schematic is shown in figure 3.7. It was not possible to fit 4 SFP connectors on the front panel of the board, so instead one GTP link (RX and TX lines) is connected to a straight SATA connector. Initially there was a plan to mount two SMA connectors on a front panel, however lack of space and possible difficulties with connecting SMA and USB cables prompted users to forgo the second SMA connector and use jumpers to choose desired SMA function.

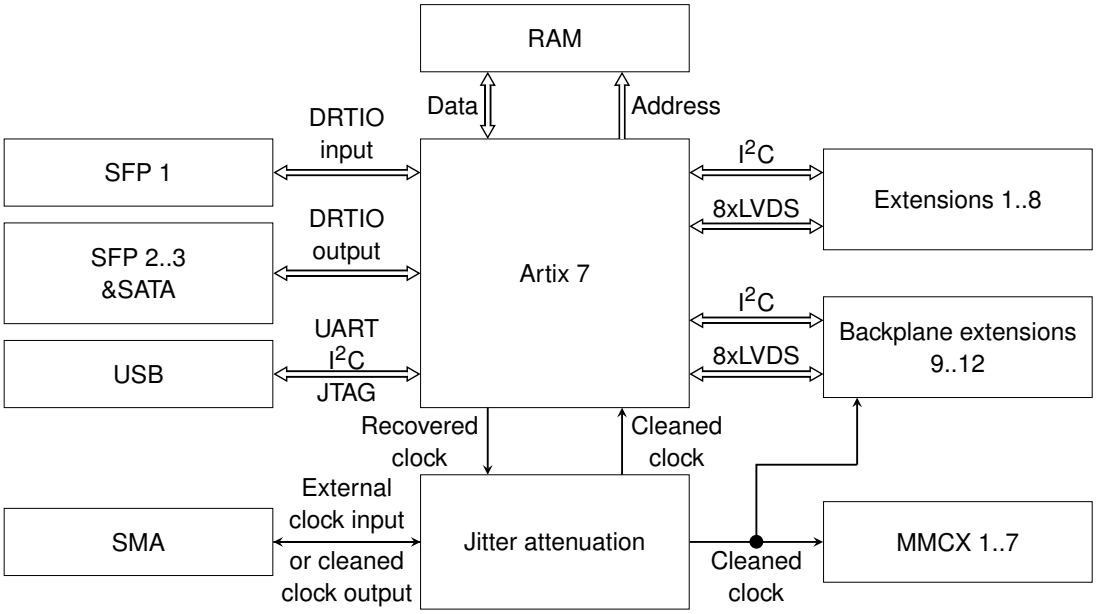


Figure 3.7: Final block schematic of Kasli

4. Board development

4.1. Implementation

Electrical schematics were done based on architecture of the module, described in the previous section. They are shown in appendix A.

4.1.1. Impedance controlled lines

In this project there are 4 types of high-speed signals that need to have impedance controlled traces:

- SMA and MMCX clock signals, DDR3 address and data signals – 50 Ω single-ended impedance,
- USB differential traces – 90 Ω differential impedance,
- LVDS signals and DDR3 differential clocks – 100 Ω differential impedance,
- LVPECL clocks – 100 Ω differential impedance, 60 Ω single ended impedance.

4.1.2. Board stackup

PCB stackup was designed to be produced in Brandner EE company. It has a manufacturing technology comparable to other PCB manufacturing companies which means, that design fabricated and working on Brandner boards will most probably be accepted by other manufacturers. Xilinx estimates that 4 signal layers are needed to route every signal in FGG484 package [74]. Thus 6 layer stackup was chosen with 4 signal layers, one dedicated ground plane and one dedicated power plane. Stackup was designed to get small enough impedance-controlled signal traces to be able to escape with a differential pair from under the FPGA package between pads. At the same time it has to be able to conduct currents powering EEMs and not be thicker than approximately 1.6 mm to fit guiding rails of the 3U crate. Appropriate values were calculated using Saturn PCB toolkit and Brandner's web page, where stackup can be created. Resulting stackup is shown in figure 4.1.

Layer	Name of layer	Type	mm	Vias
1	External copper	18µm + GalvCu 35µm	0.053	
	Prepreg	2116 (0,11mm)	0.110	
2	Innerlayer foil	18µm	0.018	
	Innerlayer	High Tg 0,2mm 18µm/18µm	0.200	
3	Innerlayer foil	18µm	0.018	
	Prepreg	2116 (0,11mm)	0.110	
	Etched innerlayer	FR4 VK0.51 (0.51mm)	0.510	
	Prepreg	2116 (0,11mm)	0.110	
4	Innerlayer foil	18µm	0.018	
	Innerlayer	High Tg 0,2mm 18µm/18µm	0.200	
5	Innerlayer foil	18µm	0.018	
	Prepreg	2116 (0,11mm)	0.110	
6	External copper	18µm + GalvCu 35µm	0.053	
		Material thickness (mm) ± 10%	**1.528**	

Figure 4.1: Board stackup chosen on the Brandner page

Layer 1 (Top), Layer 3, 4 and 6 (Bottom) are signal layers. Layer 2 was used for power supplies and layer 5 is ground. This stackup results in the following line widths and spacing for impedance controlled traces:

- 50 Ω single ended lines: 0.19 mm on top and bottom layers, 0.24 mm on layers 3 and 4.
- USB differential traces: 0.135 mm width, 0.15 mm spacing on top and bottom layers, 0.15 mm width and 0.12 mm spacing on layers 3 and 4.
- LVDS lines: 0.095 mm width, 0.125 mm spacing on top and bottom layers, 0.115 mm width and 0.12 mm spacing on layers 3 and 4.
- LVPECL lines 0.095 mm width, 0.125 mm spacing on top and bottom layers, 0.165 mm width and 0.31 mm spacing on layers 3 and 4.

4.1.3. Component placement

The board has dimensions of other 3U boards in Sinara hardware family: 162.3 mm × 100 mm. Component placement is shown in figures 4.2 and 4.3. Most of the space on the board was taken by the IDC connectors and SFP cages. The FPGA was placed above IDC connectors, in the middle, which allows for natural routing of EEM signals in 3 directions from the FPGA. Above it, there is a power supply section. Above SFP cages there are SMA, USB and power connectors. Power jack was placed on the top of front panel, so that routing 12 V power supply can be done in the upper section of the board. SMA connector was placed furthest away from the power connector to minimize noise pickup from the power supply. Jitter attenuator circuit was placed on the bottom side of the board. It minimizes the influence of EMI which may be radiated by EEMs. Also, the jitter attenuator circuit was placed furthest away from power supply circuits. On the top side, FT4232H and translator circuits were placed.

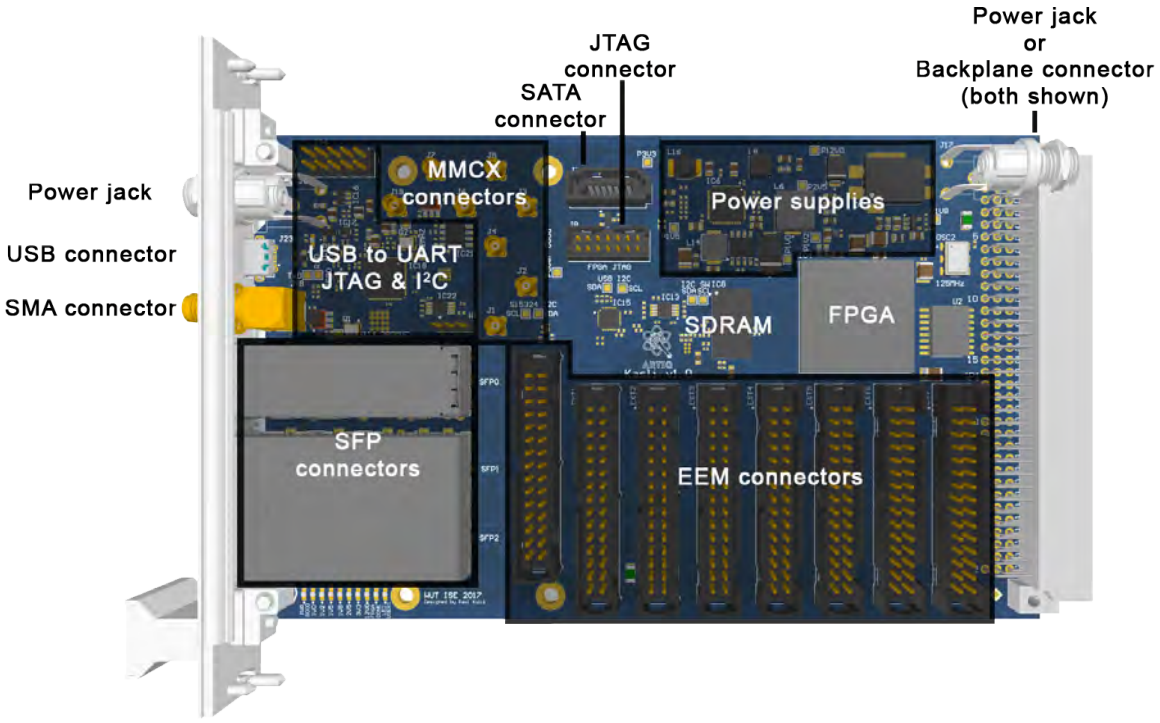


Figure 4.2: Component placement – top side

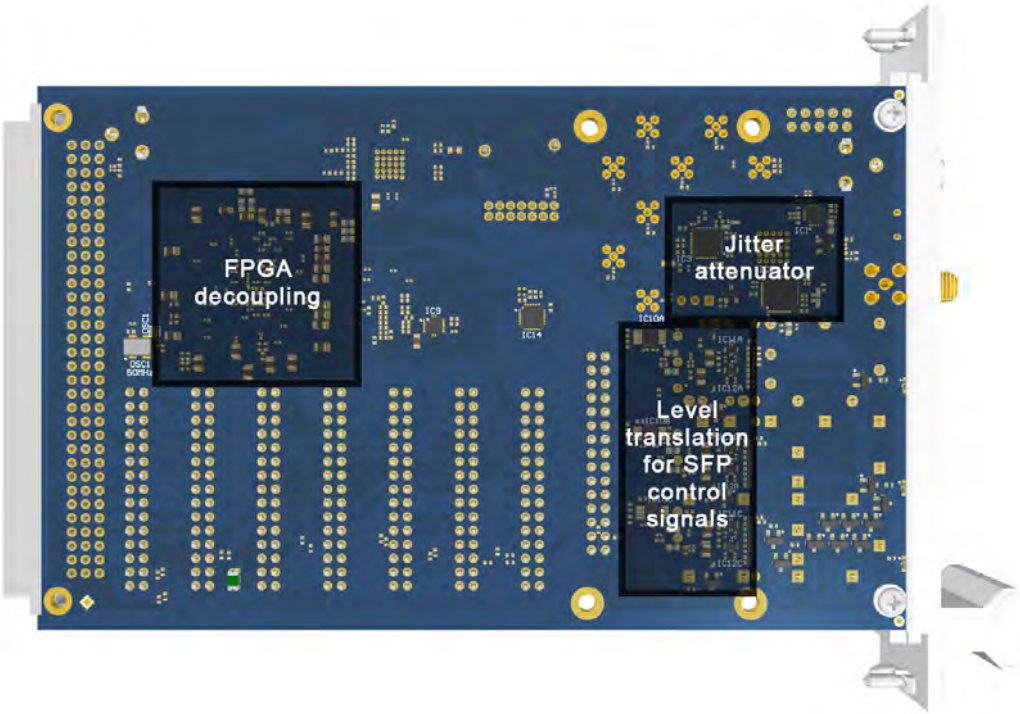


Figure 4.3: Component placement – bottom side

4.1.4. Power supplies and decoupling

When designing power supplies, care was taken to ensure traces are wide enough for expected currents. Multiple vias were used to change layers. Power was routed to integrated circuits before any signals were. This approach ensures that there is an appropriate trace to each power pin.

12 V plane took much space on the board since power jacks used to supply it are both on front and on the back of the board. This means that 12 V and its return current plane have to go through the upper half of the board (as shown on figure 4.4, layers are shown with multiple colors).

Under the FPGA, 2 layers were used to supply 2.5 V, 1.8 V, 1.5 V, 1.2 V and 1.0 V. Bottom layer and the second layer (meant for power) were used as is shown in figures 4.5 and 4.6. 100 nF 0201[8] decoupling capacitors were placed under the FPGA – the closest place to BGA power pins. 0402 and bigger capacitors, which would not fit between vias, were placed around the FPGA, with care being taken not to place elements higher than 2 mm on the bottom side of the board, so that chance of accidentally scraping them off while inserting the board into a crate is lesser. Figure 4.7 on page 37 shows capacitors under the FPGA.

The second layer on other parts of the board was divided between 2.5 V and 3.3 V. Circuits interfacing with FPGA use 2.5 V and the rest uses 3.3 V. 2.5 V also extends to all EEM connectors, so that the reference plane for all LVDS lines is continuous.

[8] 0201 and other four number codes denote the size of an Surface Mount Device (SMD) capacitors and resistors, measured in hundredths of an inch – 0201 means that the capacitor is a 0.02 in by 0.01 in rectangular.

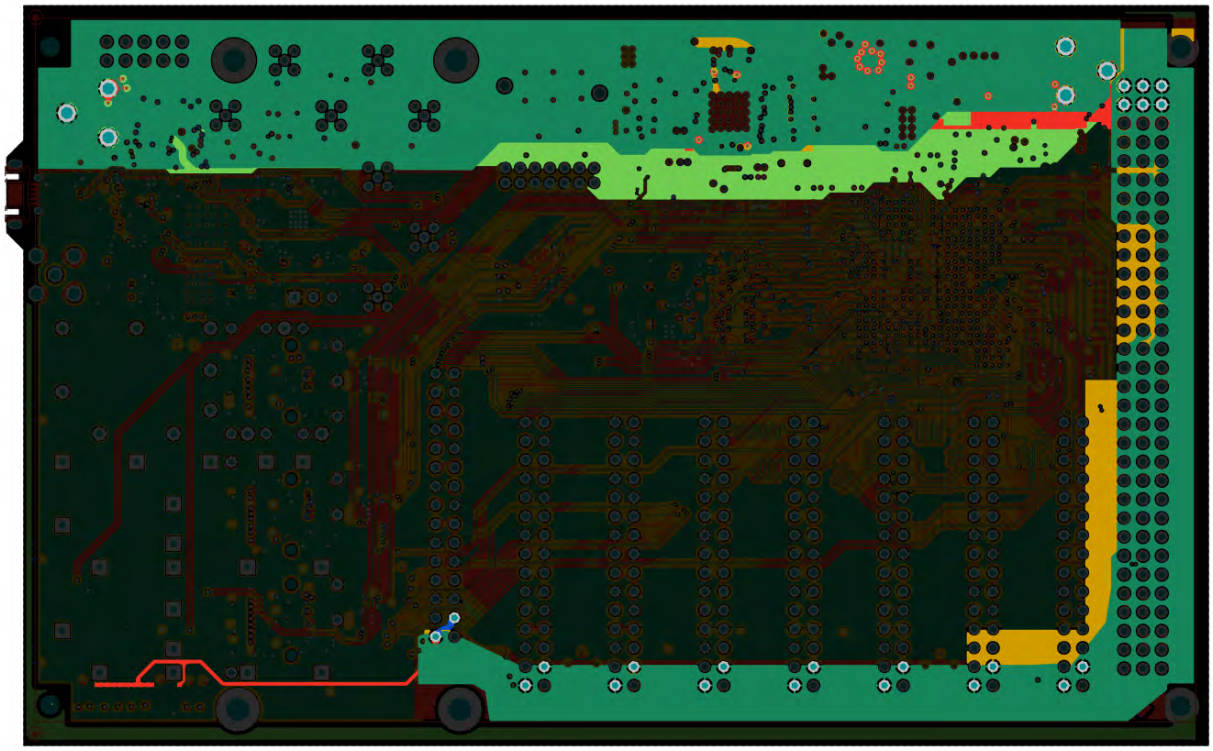


Figure 4.4: 12 V power supply

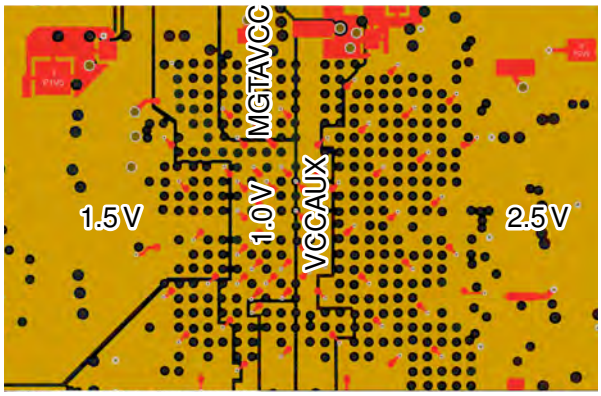


Figure 4.5: FPGA power supplies – middle layer

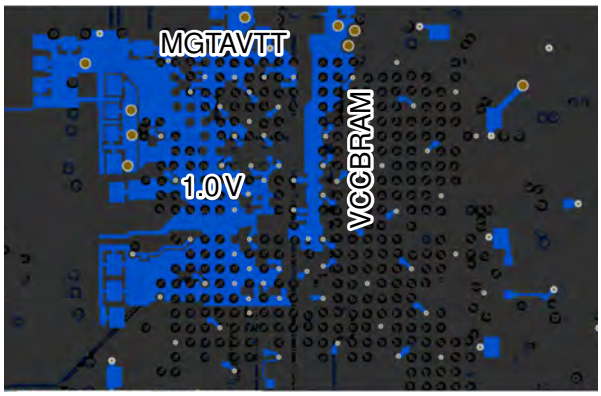


Figure 4.6: FPGA power supplies – bottom layer

4.1.5. Multi-gigabit lines

These are the lines that were routed after routing the power supplies. They conduct signals of the highest frequencies, so they need to have the most spacing and most direct route from SFP and SATA connectors to the FPGA transceivers. All these traces were routed on layer 4, which has a continuous ground reference plane (layer 5). Minimum spacing of those lines was defined according to rule of thumb:

$$3 \cdot d = 0.125\,\text{mm} \cdot 3 = 0.375\,\text{mm},$$

where d is spacing of traces between differential pair.

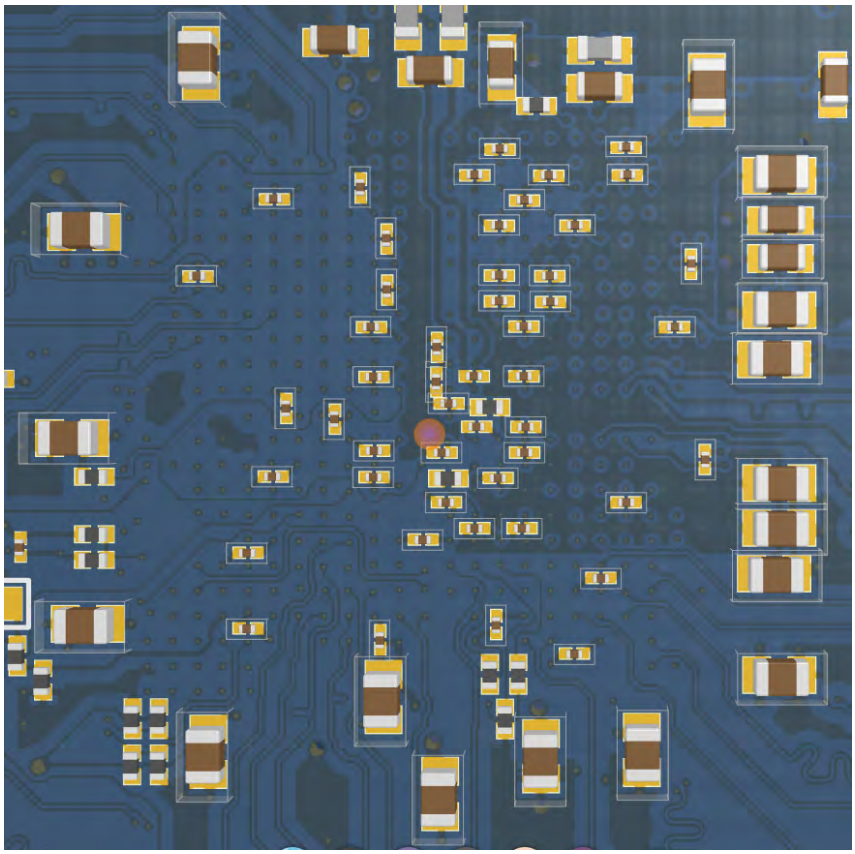


Figure 4.7: Decoupling capacitors (brown) under the FPGA

4.1.6. Clock lines

Clock lines in this project are:

- all clock fanout traces, which go to MMCX connectors,
- cleaned clock signals, which go to GTP transceiver and to I/O clock capable input,
- differential clock going to backplane connector,
- clock signals between Si5324 and ADCLK clock fanouts.

They are shown in figures 4.8 and 4.9. Colors were inverted in some images for better readability.

These lines must stay clear of any noisy planes and signal lines to avoid introducing any additional noise above additive noise of Si5324 and ADCLK clock fanouts. Clock lines, that go to 7 MMCX connectors from ADCLK948 (shown in figure 4.9) were length-matched to $\Delta s = 1\,\text{mm}$, to make sure, that all these clock outputs are phase-aligned. Built-in tools in Altium Designer were used to calculate trace lengths. Maximum skew:

$$\Delta t = \frac{\Delta s}{c} = \frac{1\,\text{mm}}{15\,\text{cm/ns}} = 6.66\,\text{ps}$$

125 MHz clock signal, which is planned to be used in ARTIQ has 8 ns period. ADCLK948 typically has rise and fall time equal to 75 ps and 9 ps skew between outputs (maximum 25 ps) [4]. This means, that skew introduced by difference in trace lengths is smaller than skew introduced by clock fanout, both of which are 3 orders of magnitude smaller than clock period.

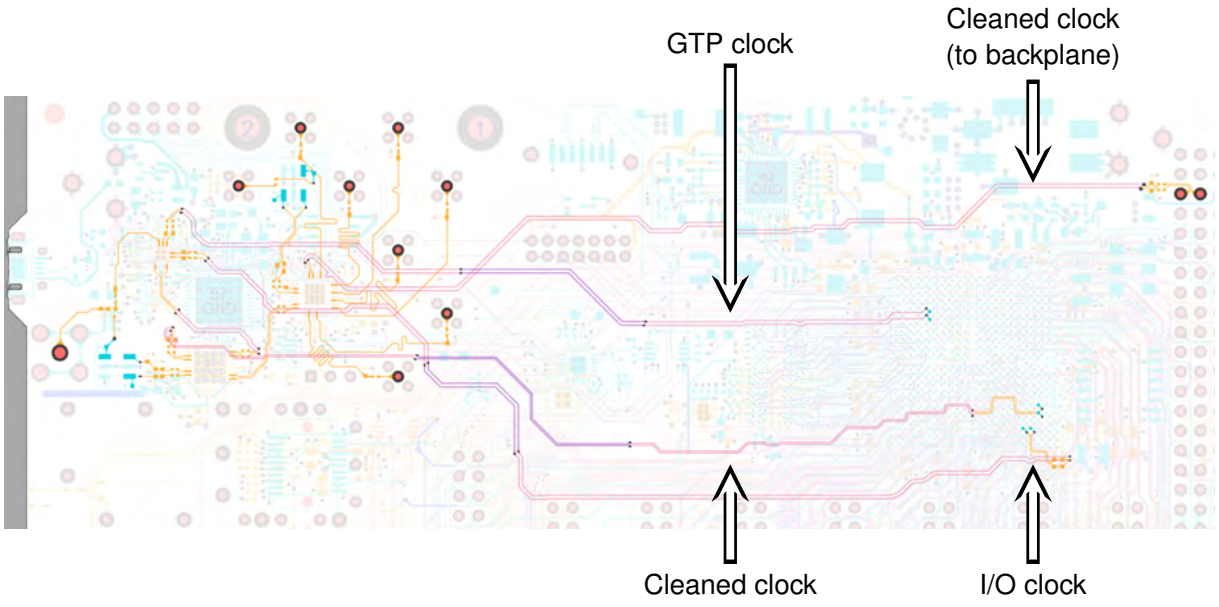


Figure 4.8: All clock lines (highlighted) on PCB

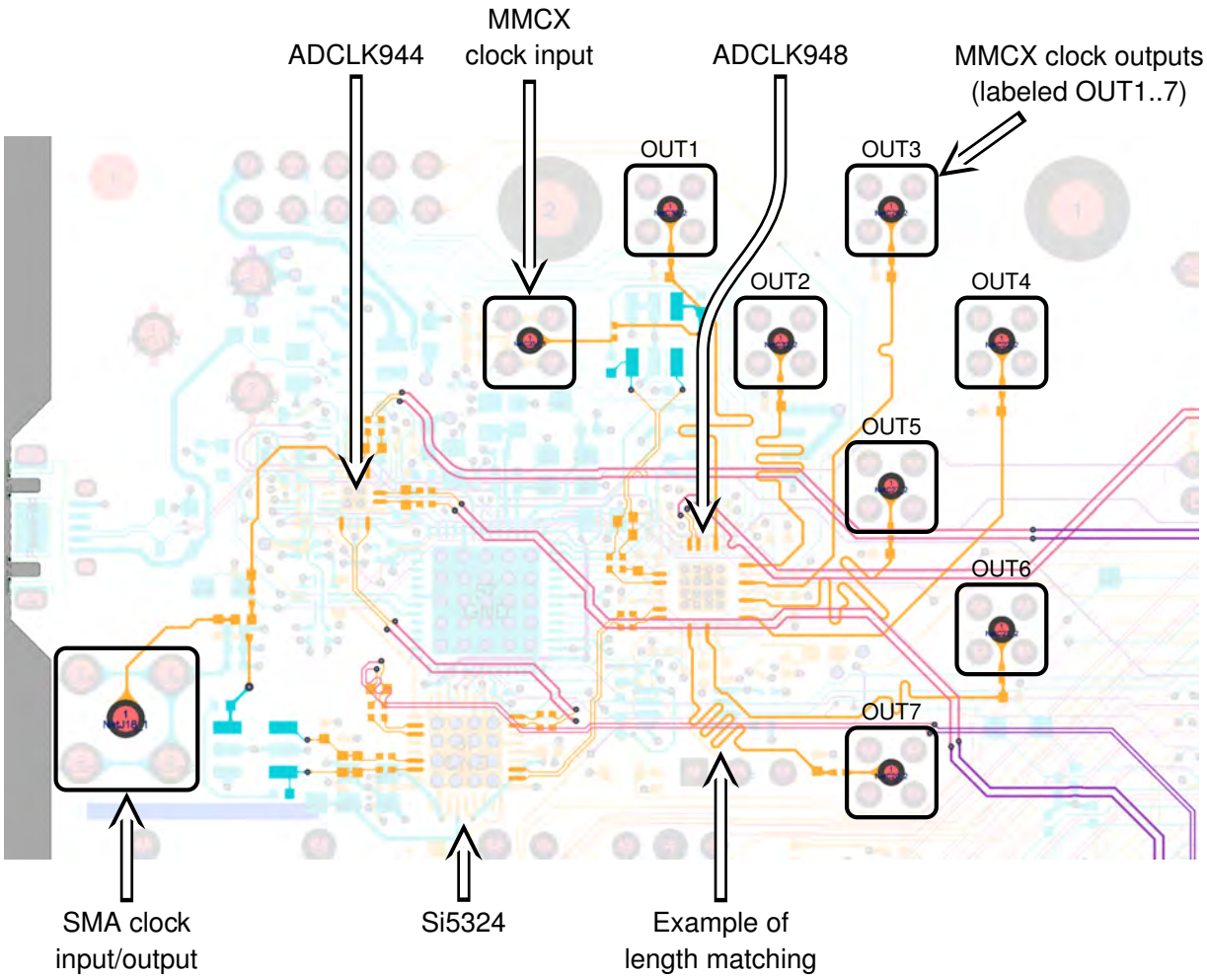


Figure 4.9: Clock lines in jitter attenuator circuit area

4.1.7. DDR3 memory

Pin assignments that were done during schematics phase were not optimal for routing since they did not include a position of FPGA pins, leading to many crosses and suboptimal paths, as shown in figure 4.10. In bank 35, used for DDR3, signals there are 4 byte groups. One had to be used for lower byte of data (DQ0..7 signals), strobe (DQS) and mask (DM), one for higher byte, strobe and mask signals, and the rest for address (A0..14), clock and control signals (clock enable, reset, RAS, CAS, write enable, ODT, bank address). Signals can only be swapped in lower data, upper data, and address signals groups. Whole signal groups were swapped between different byte groups, to find an optimal solution. Generally, the aim was to minimize crosses, by making sure that pin assignments on the FPGA loosely follow signals on the SDRAM package – address lines closer to the EEM connectors, the lower byte in the middle and the higher byte closer to power supplies. Then an automatic pin/net optimizer was used, which swaps signals in assigned groups to minimize cross count and length from FPGA pin to SDRAM pin. This provides a good starting point for routing. During routing, manual pin swapping was used to further optimize traces. Result is shown in figure 4.11. All changes were in accordance with rules in [77] and were checked with MIG.

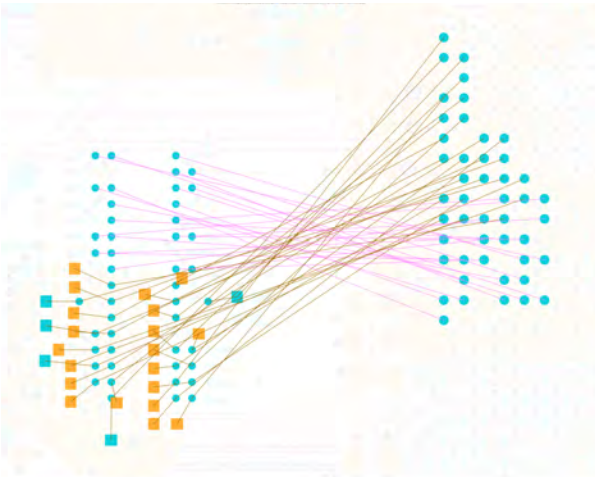


Figure 4.10: SDRAM signals before pin swapping

Figure 4.11: SDRAM signals after pin swapping

DDR3 traces have 50 Ω characteristic impedance (in accordance with [26]). Line widths were presented in section 4.1.2. Most signals were routed on top layer (shown in figure 4.12), as it meant fewer vias in signal path, meaning fewer impedance discontinuities.

Maximum expected data rates are 800 Mbit/s. Length was matched according to Xilinx' guide [26]:

"Xilinx recommends the following:

- All DQ/DM nets should be matched to their associated DQS nets to within $\pm$15 ps for DDR2/DDR3 interfaces at 800 Mbit/s

  (...)

- For unidirectional signals, all ADDR/CMD/CTRL signals must be matched to the CLK signal. It is a good design practice to match each transmission line segment (FPGA to DRAM1, FPGA to DRAM2, FPGA to fly-by termination resistor, etc.) to within a reasonable tolerance of $\pm$25 ps."

This, assuming c = 15 cm/ns in FR4 material ($\epsilon \approx 4$) gives

$$\Delta s = 15\,\text{cm/ns} \cdot 15\,\text{ps} = 2.25\,\text{mm}$$

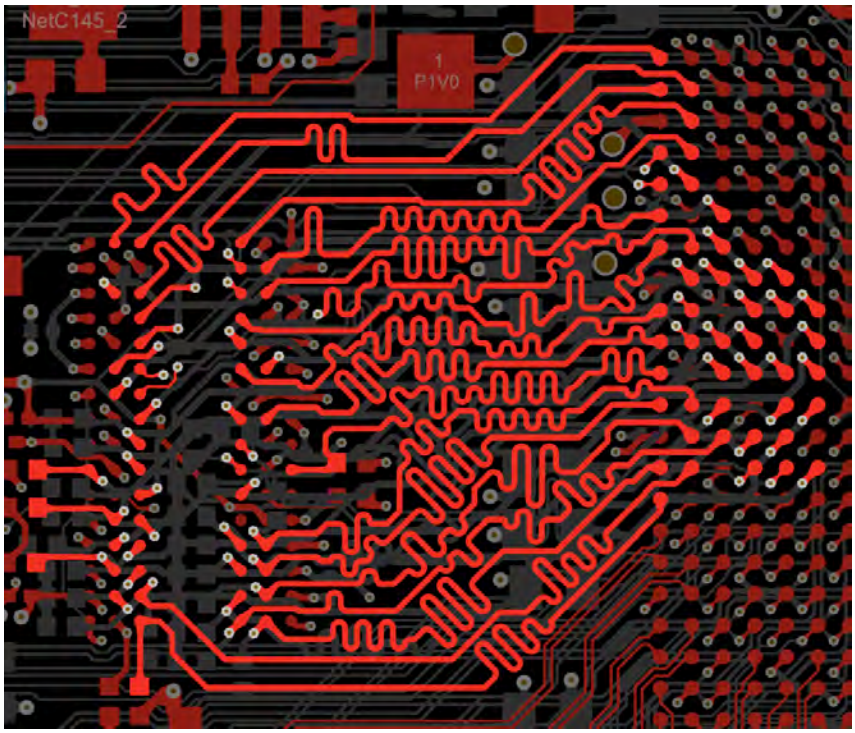


Figure 4.12: DDR3 signals on top layer

for data lines and

$$\Delta s = 15\,\text{cm/ns} \cdot 25\,\text{ps} = 3.75\,\text{mm}$$

for address lines.

The termination was done in a fly-by style which is required for DDR3 memories. In this style, all termination resistors are placed after the memory IC and the only discontinuities that are left, are short stubs going directly via vias to the IC (see figure 4.13). This style also ensures the possibility of sharing address lines between many ICs and terminating them in one place.

Figure 4.13: DDR3 fly-by termination style

4.1.8. EEM signals

In figure 4.14 all EEM LVDS lines are shown. Each 8 differential pairs going to EEM connector were length matched to 1.5 cm (100 ps assuming c = 15 cm/ns in FR4). 32 differential pairs were routed to backplane connector.

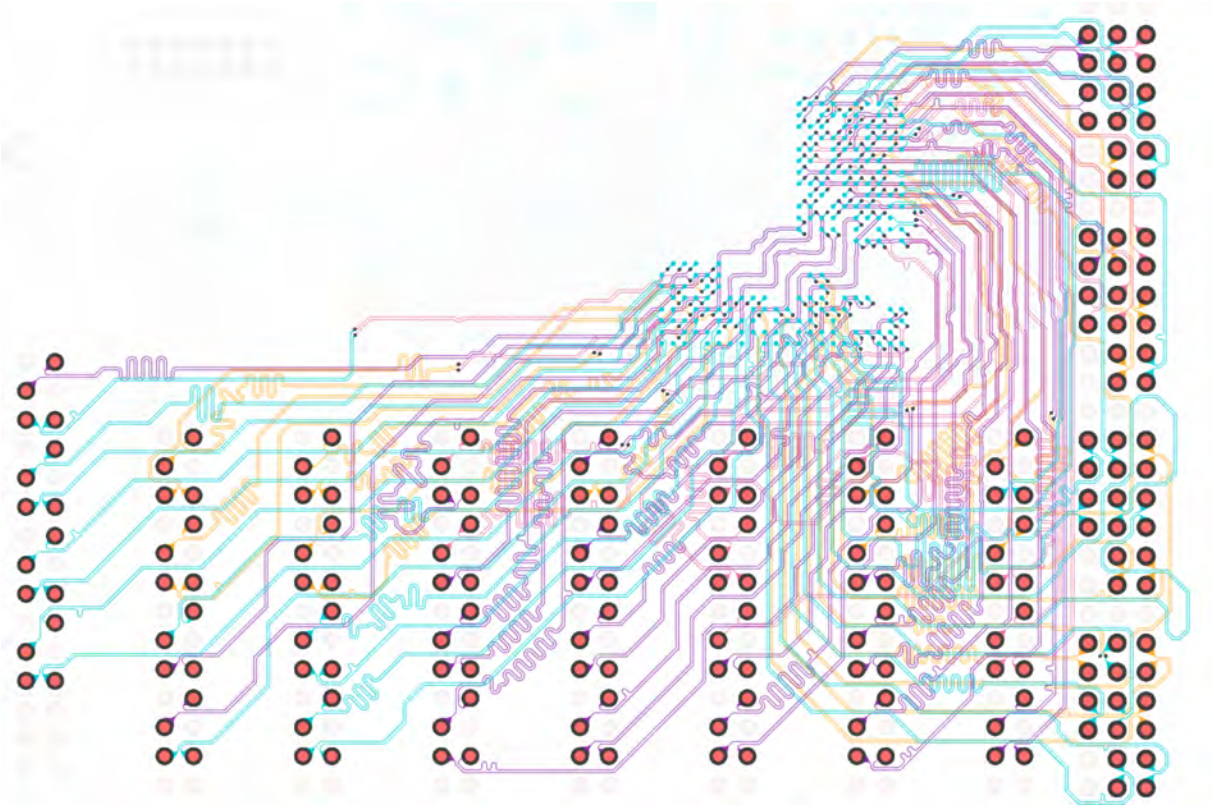


Figure 4.14: All EEM signals

4.1.9. Test points

A test point is an unmasked part of the copper, which can be easily touched with a probe connected to an oscilloscope or a multimeter. In figure 4.15 several test points are shown, that were placed in easy to reach places to aid debugging of the board. All power rails, oscillators, UART and I^2C lines are connected to them.

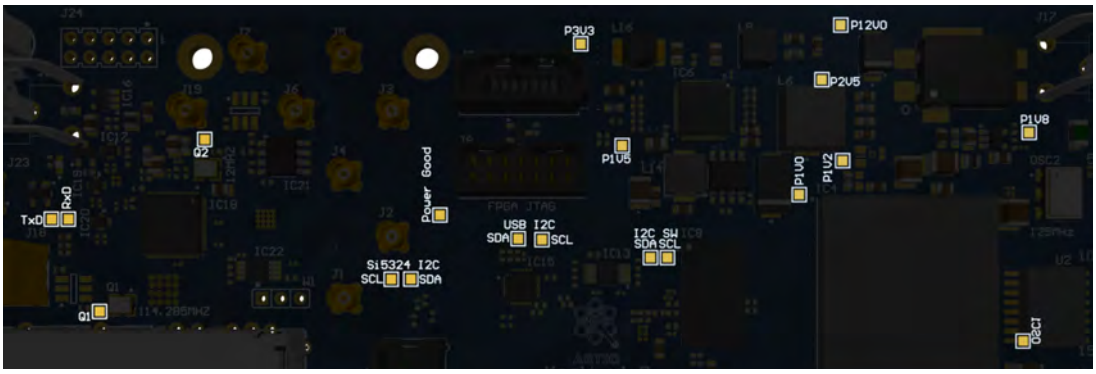


Figure 4.15: Testpoints on the PCB

4.1.10. Other

- The nets with relatively slow signals were routed last. This includes signals like I^2C, UART, General Purpose Input Output (GPIO) for SFP control, JTAG.
- Board outline was changed slightly so that USB cable would be possible to connect to short micro USB connector if front panel was mounted. Thus there's a bump in board outline. Otherwise, the board is rectangular.

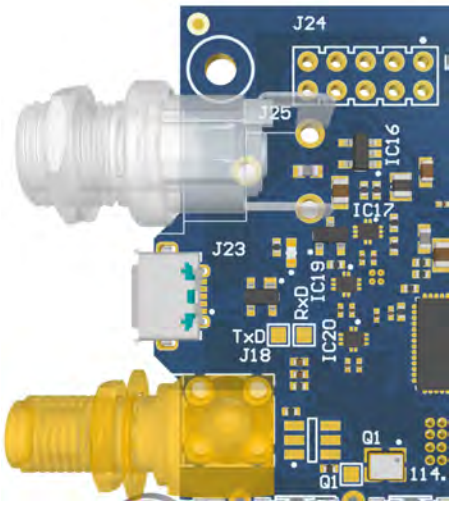


Figure 4.16: Board outline

- Copper balancing was done by filling unused parts of the board with polygon fill connected to the ground. This way each part of the board is etched to the same extent. It had to be done on all layers. Copper balancing on all signal layers is shown in figure 4.17.

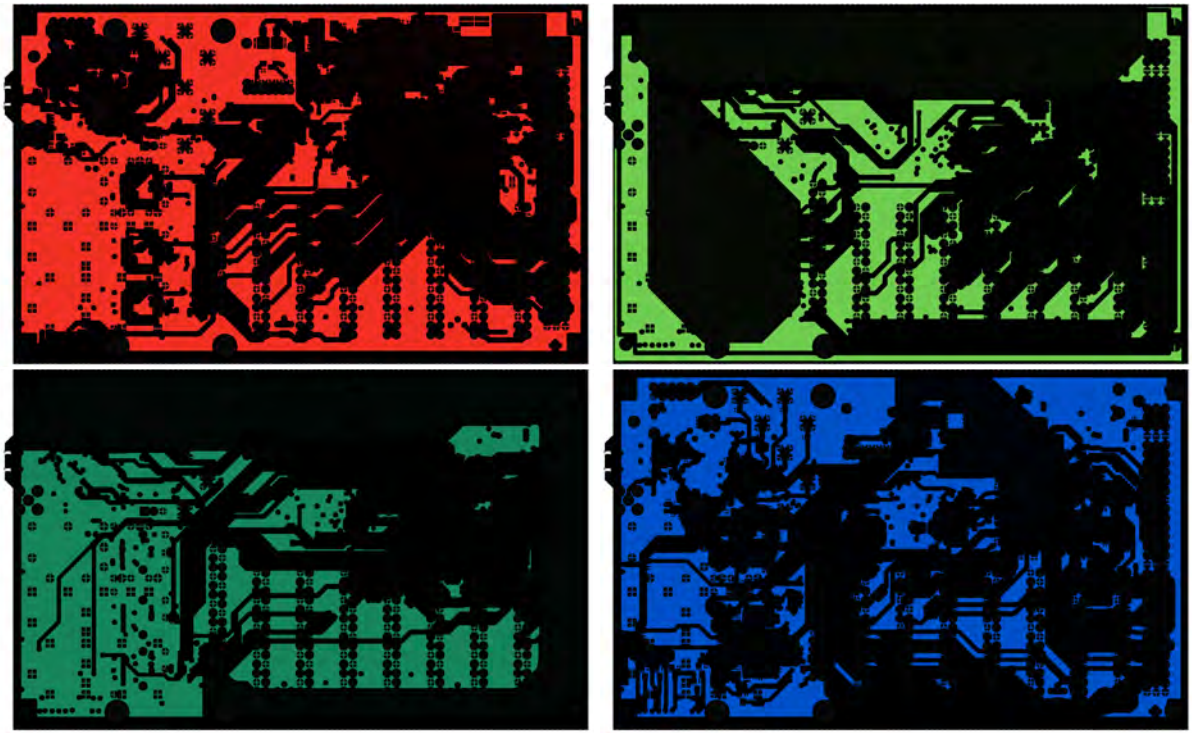


Figure 4.17: Copper balancing on all signal layers

- Net lengths in differential pairs also have to be equalized, due to a difference in length on turns and different starting points. Maximum 5 mm difference is allowed. An example of net length equalizing is shown in figure 4.18.
- Teardrops are shown in figure 4.18. They are added to minimize effects of tolerances in drills, increase the mechanical strength of the pads and to provide a smooth impedance transition to the pad.

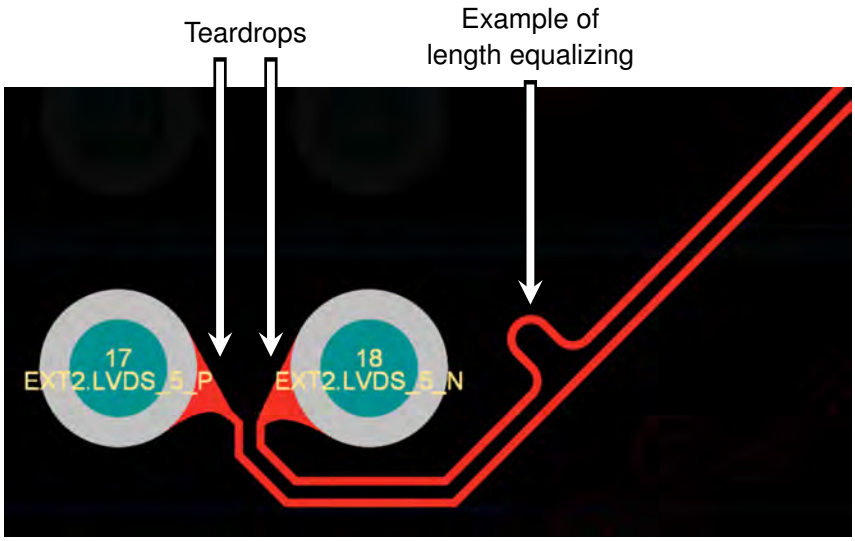


Figure 4.18: Example of teardrops and equalizing of net length within differential pair

4.1.11. Backplane adapter

For interfacing with backplane connector, an adapter board was made. It houses 4 EEM connectors, an alternate power supply input, and clock output. It allows user to connect up to 12 EEM extensions, if total current on 12 V rail is not higher than 5 A (power connector limit). 3D render of this board is shown in figure 4.19.

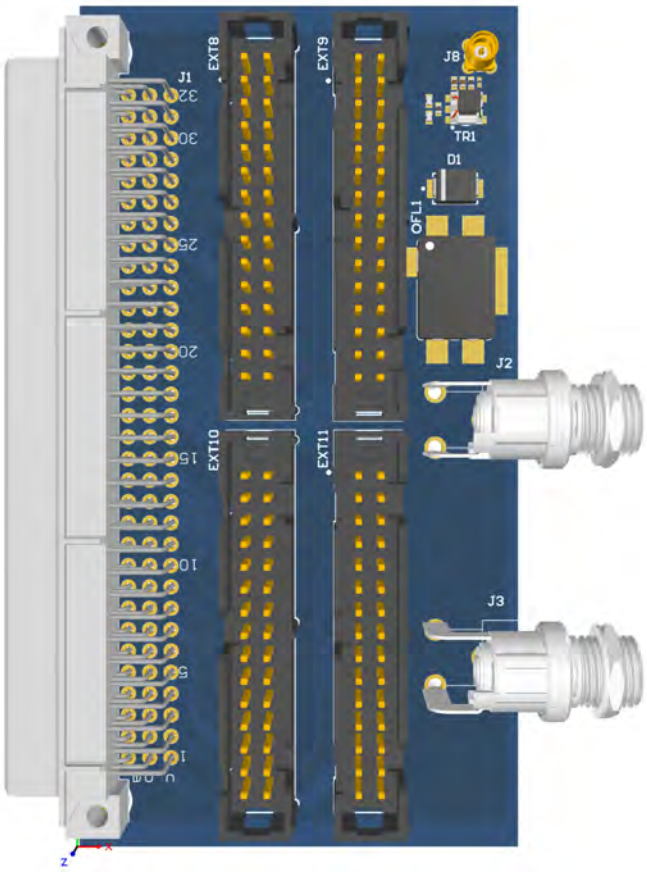


Figure 4.19: Backplane adapter board

4.2. Simulations

All simulations were done using Hyperlynx software. Altium Designer has an option to export a board file to a format readable by Hyperlynx.

For power supply DC simulation each IC was assigned to draw current based on table 3.5.

For AC (signal) simulations Input/output Buffer Information Specification (IBIS) models were used. These are models that contain information about I/V (current/voltage) response of chip pins. They are usually published by a manufacturer since they do not contain any sensitive information about the IC. They allow users to make simulations to check for over/undershoots, signal rise and fall times and non-monotonic signal transitions, all of which may suggest signal integrity issues with the PCB project. IBIS models were assigned by reference designator with an exception of Artix-7 FPGA, which had most of the signals assigned from IBIS model generated from Vivado project and GTP drivers and receivers were assigned manually (this required special models, which are not included in the generated model). Additionally, full simulation also required setting up termination resistors' value (on DDR3 signals and on LVDS and LVPECL clocks) and capacitors value. Differential pairs also had to be set up manually.

All signal lines were assigned to signal classes with proper stimuli[9] shown in table 4.1.

Table 4.1: Signal classes and stimulus

Signal class	Stimulus
GTP	6.25 Gbit/s PRBS
DDR3	800 Mbit/s PRBS
EEM signals	600 Mbit/s PRBS
USB	480 Mbit/s PRBS
Clocks	125 MHz oscillator
DDR3	800 Mbit/s PRBS
I^2C & GPIO	1 Mbit/s PRBS[10]

These stimuli were used in GTP and clock simulations. DDR3 simulation used its own stimuli however stimuli from table 4.1 was used for crosstalk simulation.

4.2.1. Multi-gigabit and clock lines

These simulations were made to make sure, that clock fanouts and FPGA's gigabit transceivers send and receive proper signals without any reflections, which could be caused by improper termination or wrong characteristic impedance of the trace. ADCLK944, ADCLK948, and Si5324 IBIS models were obtained from manufacturer sites [2, 3, 59]. Artix-7 IBIS model can be exported from the Vivado project. Project with SDRAM controller was used. For gigabit transceivers IBIS-AMI models were used, which can be downloaded from Xilinx' site after approval [70]. FPGA pins had to be assigned manually the driver or receiver model. On the far side of the track (at the connector) those models were also used so that only board traces were simulated, not connectors and cables. Clock chips had their drivers and receivers assigned automatically after assigning models by reference designator.

[9] With an exception of 50 MHz clock signal

For simulations several stimuli were used. For gigabit transceivers Pseudo-Random Bit Sequence (PRBS) was used at 1 Gbit/s, 3 Gbit/s and 6.25 Gbit/s. For clocks 100 MHz, 125 MHz and 400 MHz (as the upper range of Si5324 output) oscillating signals were used. For slow signals (e.g. Light-Emitting Diodes (LEDs), UART, I^2C) which were used for cross-talk simulation PRBS 1 Mbit/s signal was used, as the lowest bit rate possible to choose in Hyperlynx.

Sample eye diagram simulation is shown in figure 4.20. Overall no problems with crosstalk or signal integrity were spotted, although, based on geometry analysis, some control lines were too close to gigabit lines. Their rerouting required some time and was pushed back to the v1.1 version of the board. It was not critical, as in simulations even 12 Gbit/s transfers had proper eye diagram openings. Transceiver input requires minimum 150 mV of differential voltage swing to register change, which is easily achieved. 1% (of signal period) gaussian jitter was added to the signal.

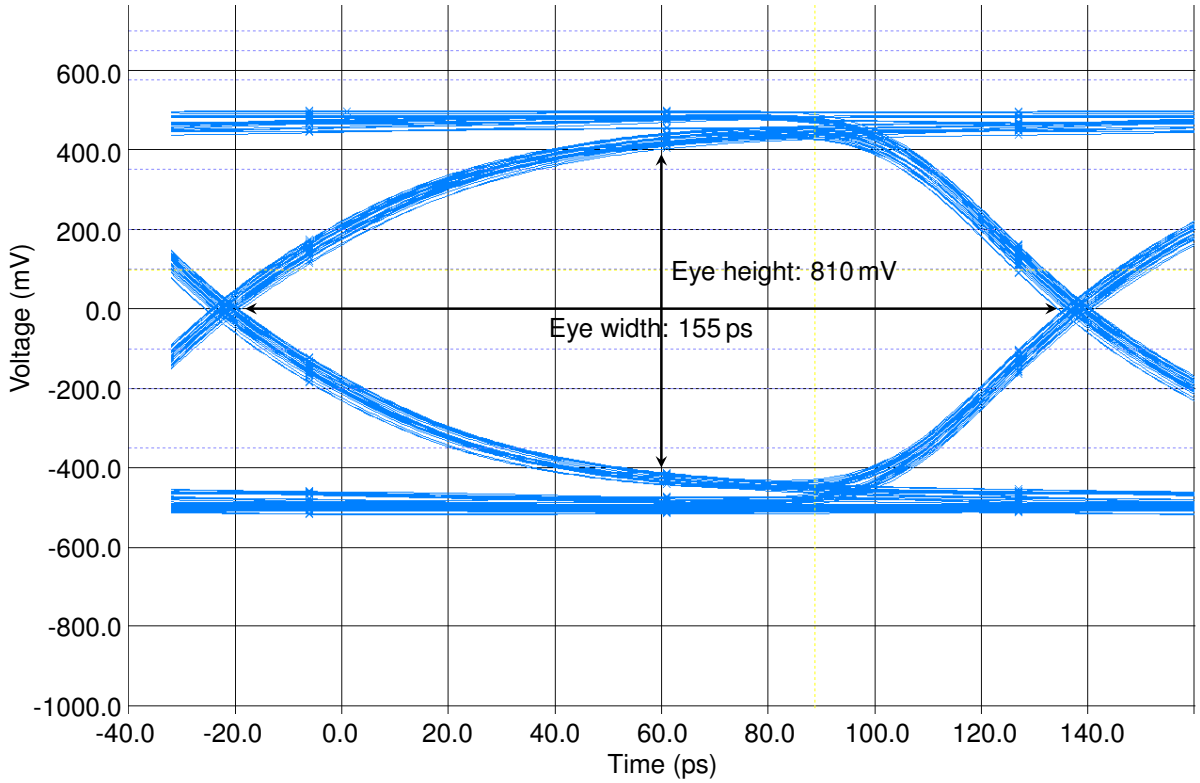


Figure 4.20: GTP simulation – receiver in FPGA, signal from SFP2 connector

One unexpected result was that lines with capacitors in series (DC block) had differential voltage swing shifted by a constant value. This was traced to simulation starting from uncharged capacitors. Once several microseconds were simulated it was evident, that differential voltage was gradually loosing this DC shift, as is shown in figure 4.21.

In figure 4.22 all MMCX outputs are shown. These 7 signals have less than 10 ps difference in simulation, which is much less than signal period and thus it is not visible in the figure.

4.2.2. DDR3 memory

For memory simulation, the IBIS models of the memory controller (FPGA) and memory are needed. IBIS model of the memory part was downloaded from manufacturer's site [45]. For the FPGA, the

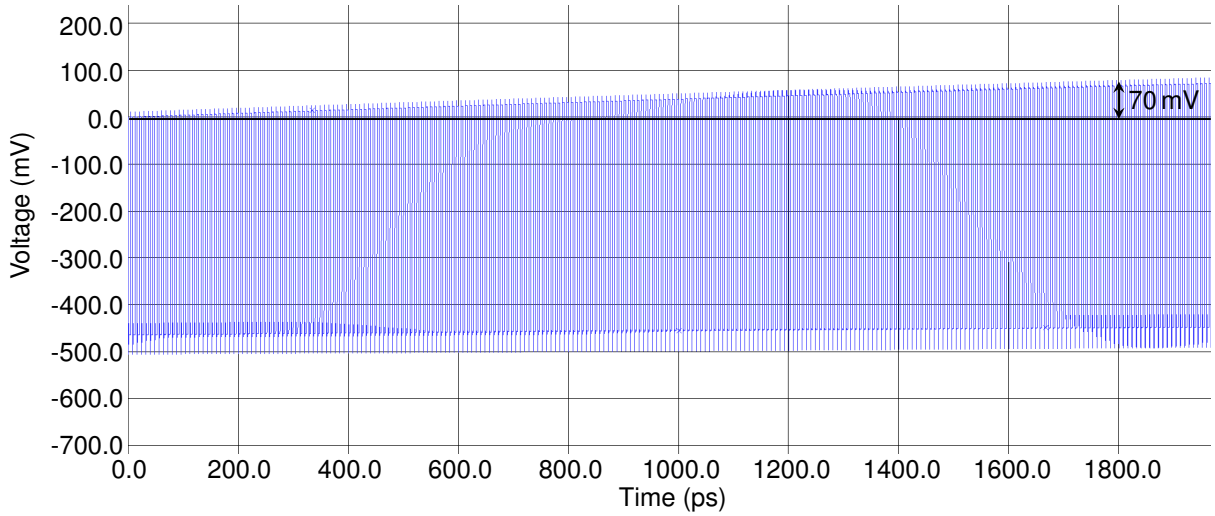


Figure 4.21: MMCX clock output in longer timescale – visible equalization of DC level

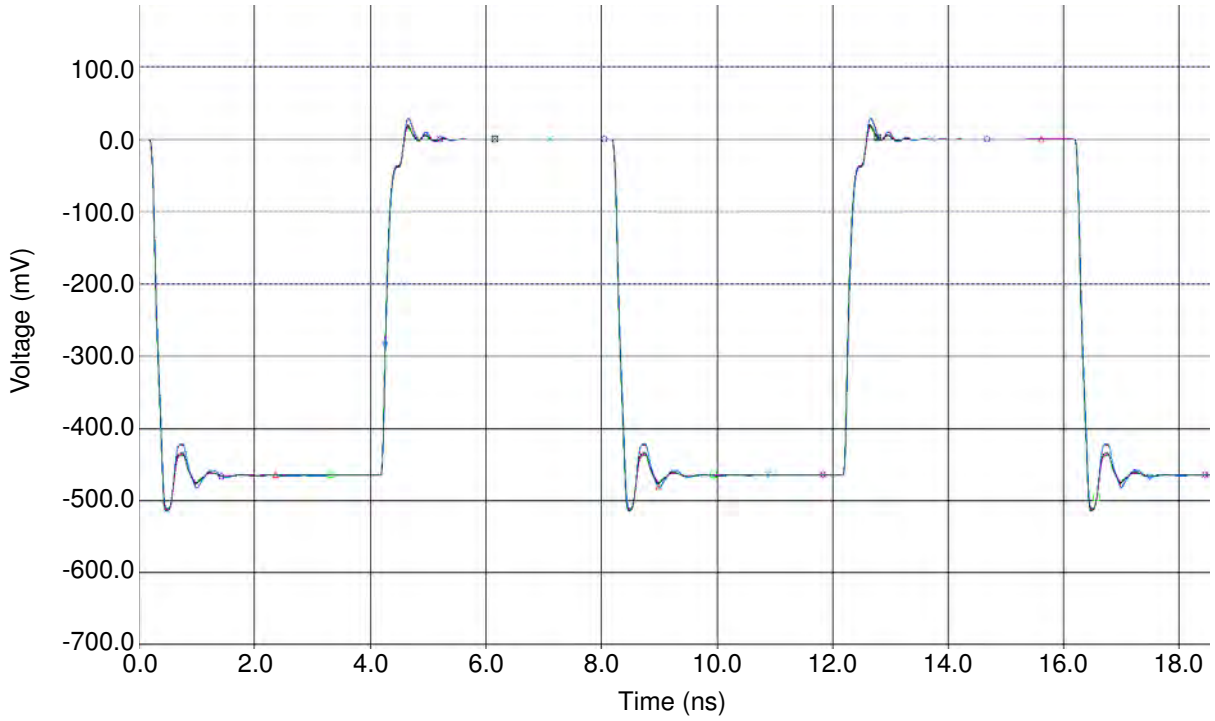


Figure 4.22: MMCX clock outputs simulation

same model as gigabit transceivers and clocks, exported from Vivado project was used. In Hyperlynx DDRx wizard tool was used, which takes default values (as per JEDEC standard). Crosstalk and power-aware models were used in the simulation. Transfer speed was simulated at 400 MHz clock speed. Dynamic ODT was used.

Overall all control, address, and data lines when writing were working properly in simulations, however, in read from memory simulation, there were problems with non-monotonic signals, one of which is shown in figure 4.23.

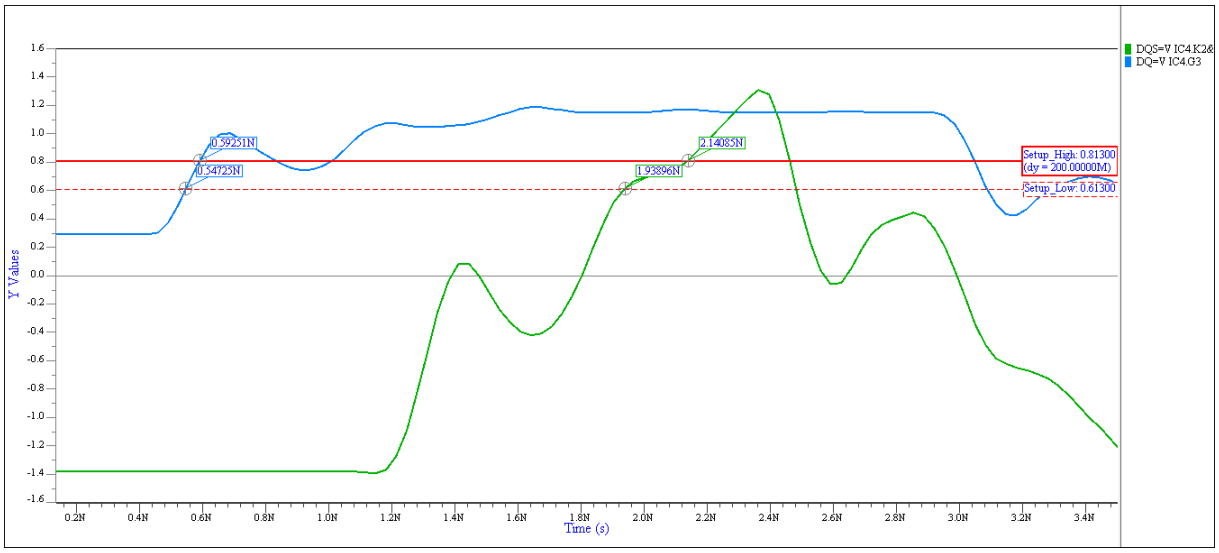


Figure 4.23: DDR3 DQ setup simulation without input termination in FPGA

These results were unexpected, given that the write simulations were excellent. This problem was traced to Xilinx IBIS models not exporting I/O models with uncalibrated split termination, which was designed to be used during memory read operations. This meant, that without proper termination reflections were occurring, resulting in the non-monotonic signals, which could not be interpreted properly by the simulator. To fix this, receivers from SDRAM were used in place of FPGA ones. IBIS file of FPGA was modified to include a model selector to switch between the original output model and input model from SDRAM with on-die termination enabled. After this operation DDR3 simulation was passed. Simulation with proper waveforms is shown in figure 4.24, this is the same trace as in figure 4.23.

4.2.3. Power supplies

Current values from table 3.5 were distributed evenly between all power supply pins of ICs found in this project. Additionally, ferrite beads were assigned to have 25 mΩ resistance [47], which allows Hyperlynx to treat many separate copper polygons and traces as one net with resistance bridges for purposes of simulation. For 12 V rail two furthest EEM connectors (EXT0 and EXT1) were set to draw 3.2 A.

12 V line was heavily optimized to get rid of hot-spots (sample hot-spots, that were present before optimization, are shown in figures 4.25 and 4.26), which could potentially burn copper and cause chain reaction as other ways that current can go would be stressed more.

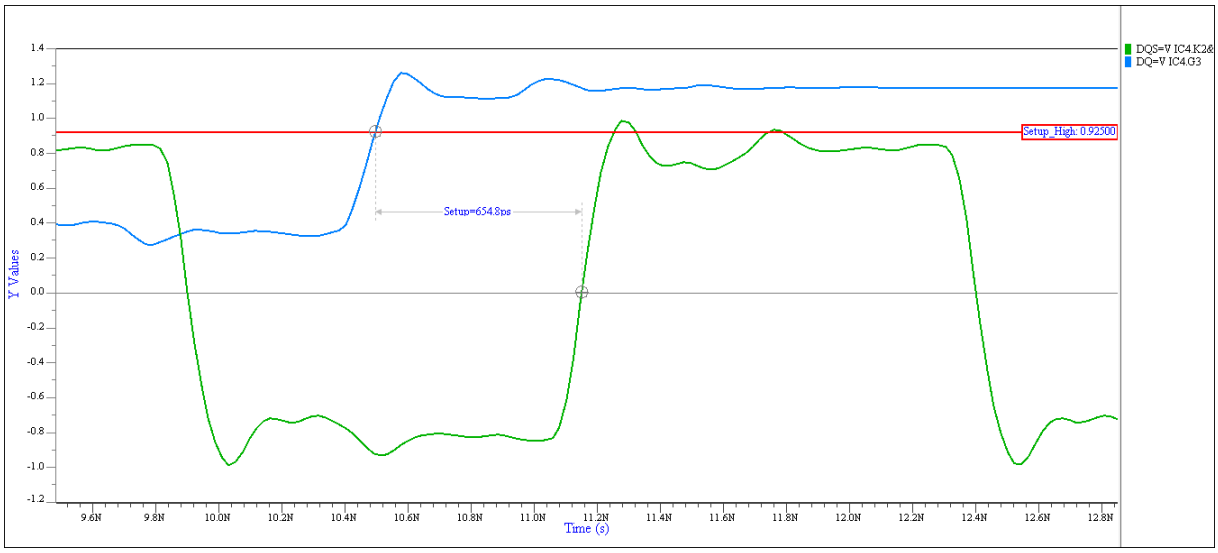


Figure 4.24: DDR3 DQ setup simulation with different input buffer

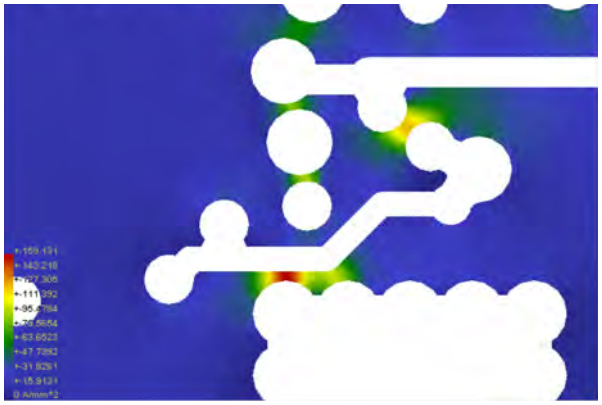


Figure 4.25: Two hot spots near vias and trace

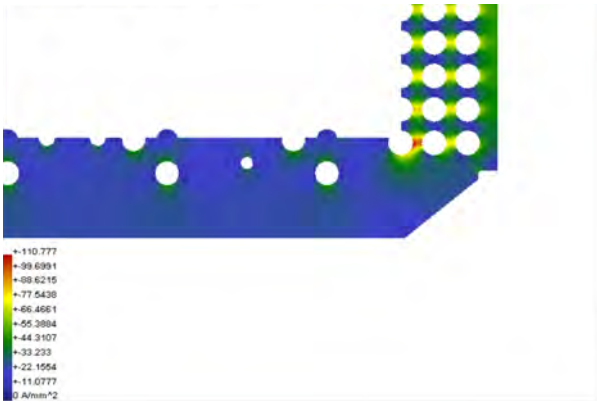


Figure 4.26: Hot spot between through-hole pins

After optimization current density was reduced by 100 A/mm^2 on 12 V rail, as shown in figure 4.27. The aim was to reduce maximum current density to 50 A/mm^2 where possible. This was done by blocking very narrow paths (cutting out copper from polygon fill) and repositioning vias and traces.

Additionally, care was taken to reduce voltage drop to no more than 50 mV on all supply rails other than 12 V. This was done by widening traces and making multiple paths for current. Sample simulation is shown in figure 4.28. In this simulation, it is visible that the voltage drop is equal to 197.7 mV. A high current draw means that there's a significant drop on inductor used by the buck converter. However, the voltage on power supply pins is higher due to voltage after inductor being stabilized to 1 V.

4.2.4. Thermal simulation

For thermal simulation information about each chip power draw was used, alongside with thermal conductivity of each chip's package (found in most datasheets) and dimensions of the package (exported from Altium Designer as an IDF file).

For simulation there were several assumptions about the environment:

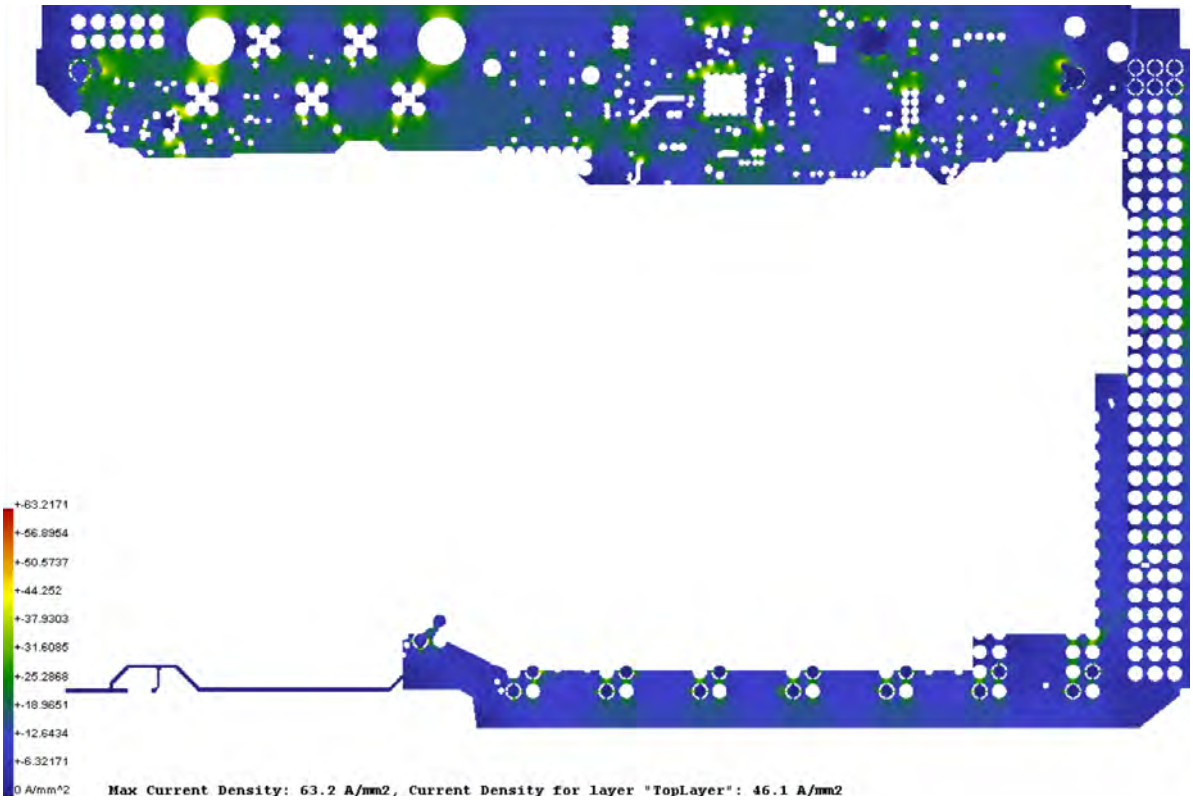


Figure 4.27: 12 V rail - current density simulation

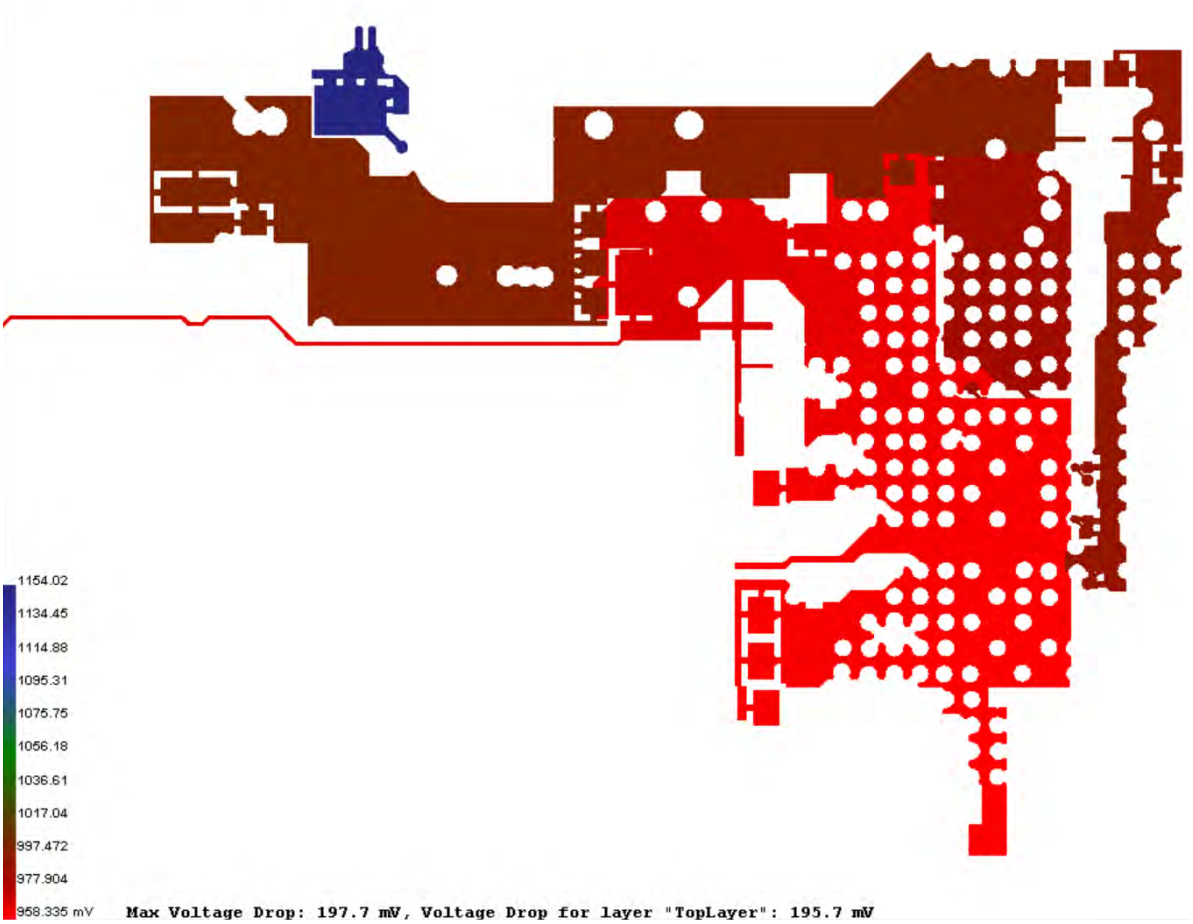


Figure 4.28: 1 V rail - voltage drop simulation

- board is placed in rack vertically,
- forced airflow is coming from the bottom of the rack,
- adjacent boards emanate 8 W,
- room temperature is 22 °C.

Three simulations were done:

- no forced airflow,
- 50 cm/s forced air flow,
- 130 cm/s forced air flow.

Simulation results with different cooling scenarios are presented in figures 4.29 to 4.31. Without forced cooling board reaches temperature of nearly 80 °C. This means, that the board will need to be cooled while under full load.

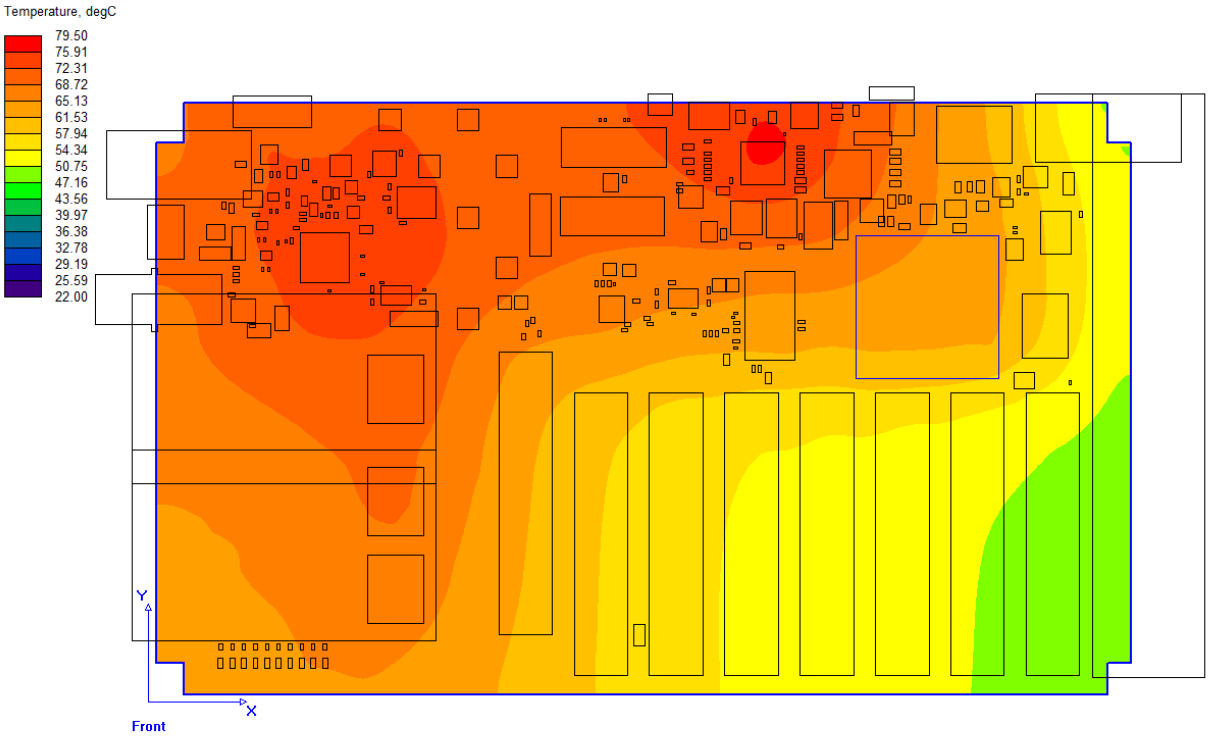


Figure 4.29: Thermal simulation – no forced cooling

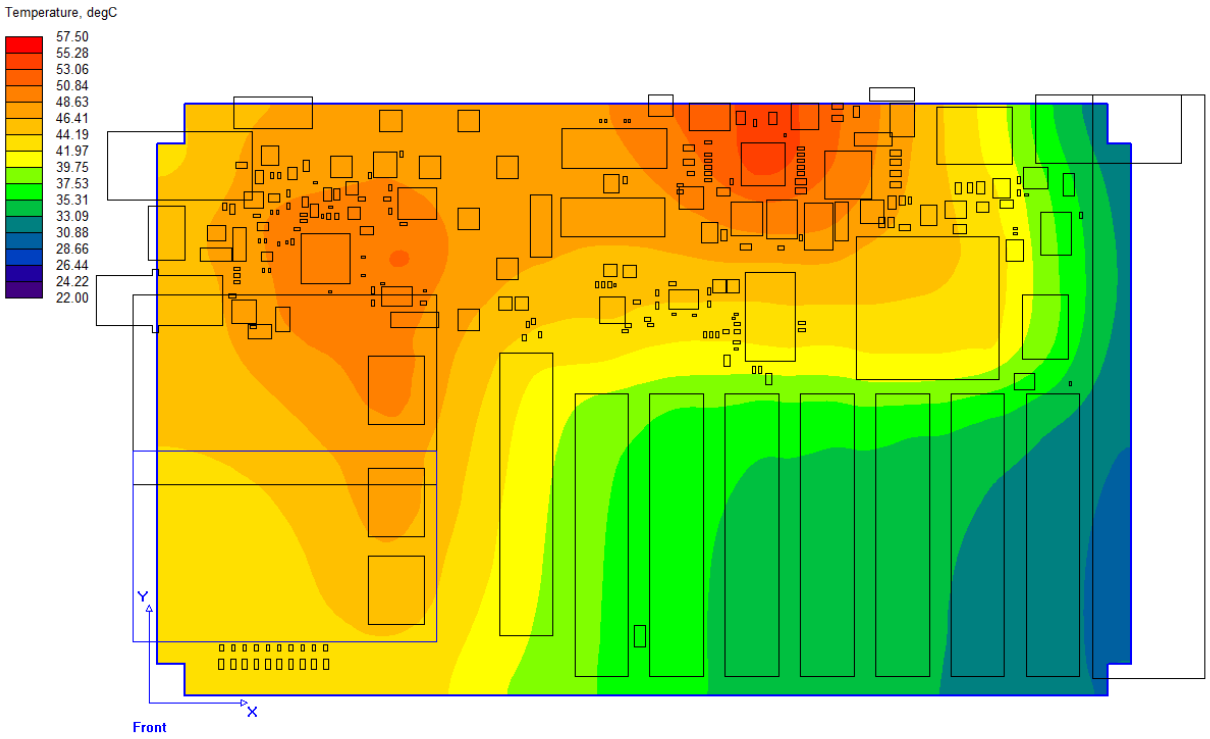


Figure 4.30: Thermal simulation – 50 cm/s forced air flow

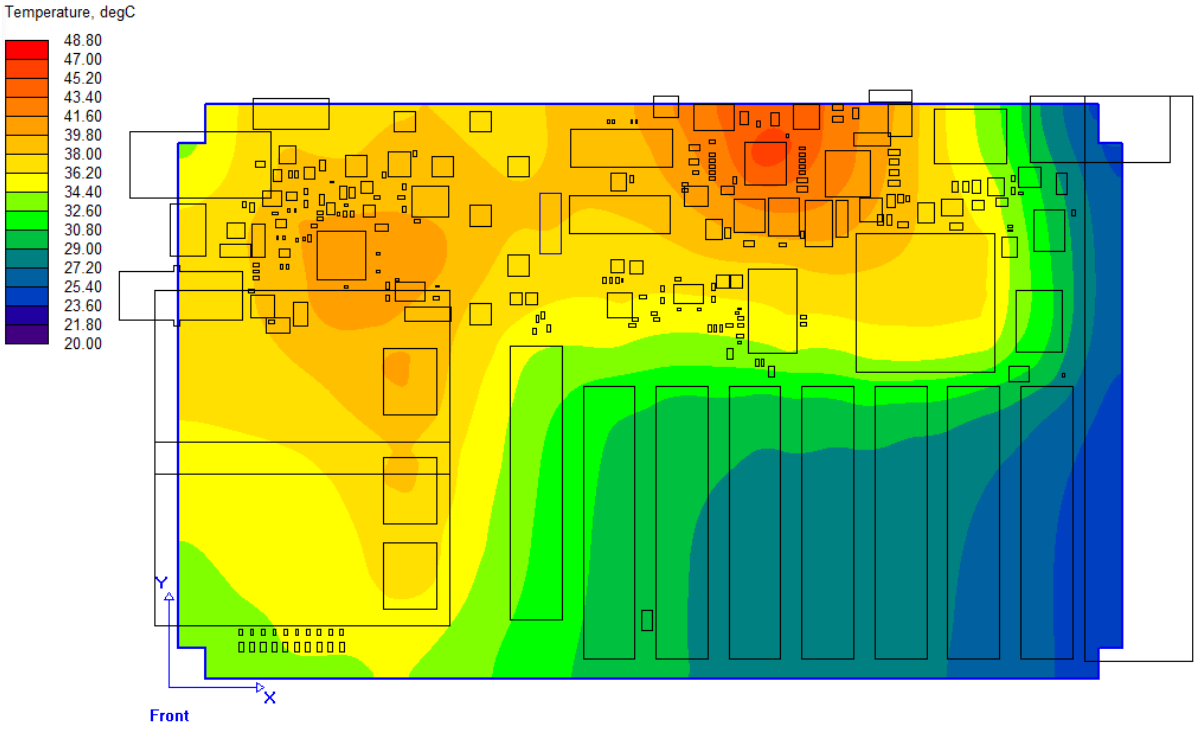


Figure 4.31: Thermal simulation – 130 cm/s forced air flow

5. Tests and measurements

After Kasli board (shown in figure 5.1) was manufactured all its functions had to be tested.

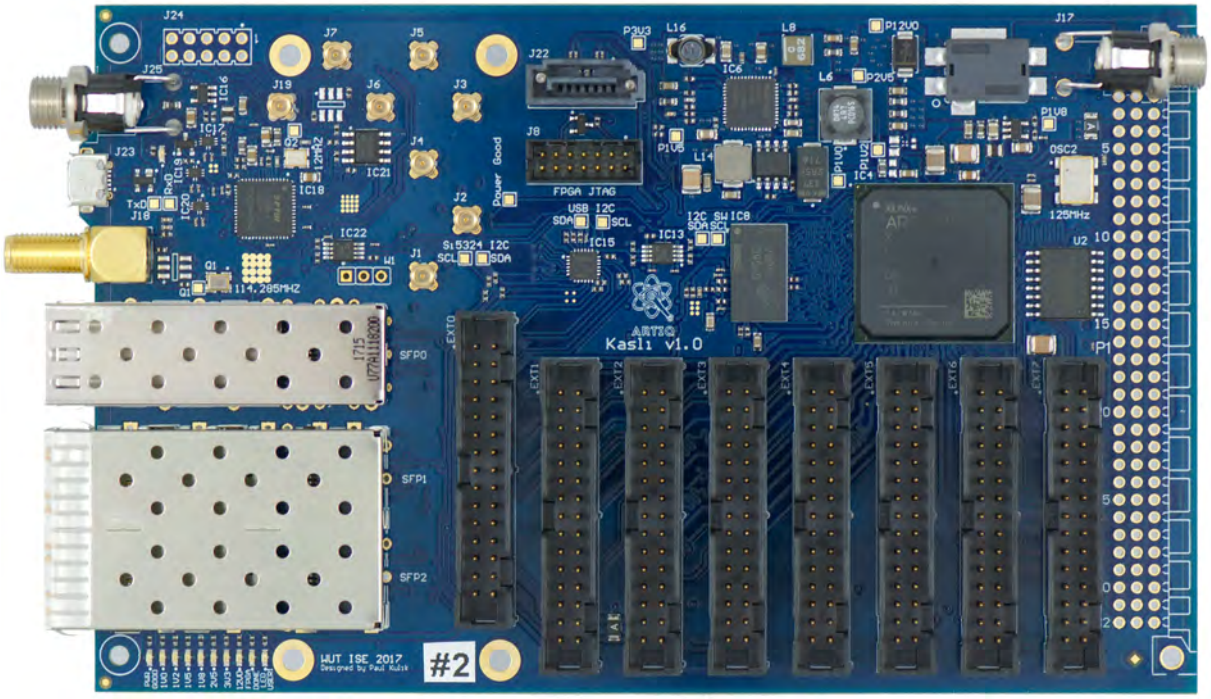


Figure 5.1: Manufactured board

5.1. I²C bus

The I²C bus was tested to check if each device on the bus accepts its own address and if bus switches work correctly. This was done with PyFtdi, an open-source Python library which provides a user-space driver for FT4232H [21]. However, the I²C mode could not be used because MPSSE, which is used to produce waveforms, is only available on bus A and B, while the I²C bus is connected to bus C. This required bit-banging I²C signals. Waveforms produced this way had random timing and frequency of ≈1 kHz due to pins being controlled by an interpreted script, using python library, using the USB connection to FTDI chip and then FTDI chip itself. Surprisingly all devices on bus accepted these transmissions, as none of them had timeout implemented and data was always stable at the rising edge of the clock. Communication with I²C switches and Si5324 was tested. I²C buses to SFP0 to 2 was checked using probe connected to oscilloscope and buses to all EEMs were tested using RJ45 EEM during testing of the EEM connectors.

In figure 5.2 a sample I²C transaction is shown. It is a transaction to the device with address 0x71. Last zero before acknowledge (ACK) means that this is a write transaction. This address was acknowledged by the slave. Then data (0x08) was sent and acknowledged. This transaction sets 3rd output as active in the I²C switch (IC15). The difference in voltage level between ACK and 0 came from TCA9517 bus repeater, which is able to automatically set translation direction since ACK should be propagated to the master side. Later I²C communication was also checked with the FPGA

acting as a master on the bus, using ARTIQ[11], which initializes Si5324 with the I^{2}C bus. Setting up ARTIQ development environment is described in appendix C. In all tests, addressed devices were acknowledging transmission correctly and I^{2}C switches connected appropriate buses to the main I^{2}C bus. Over 100 power cycles no errors were observed.

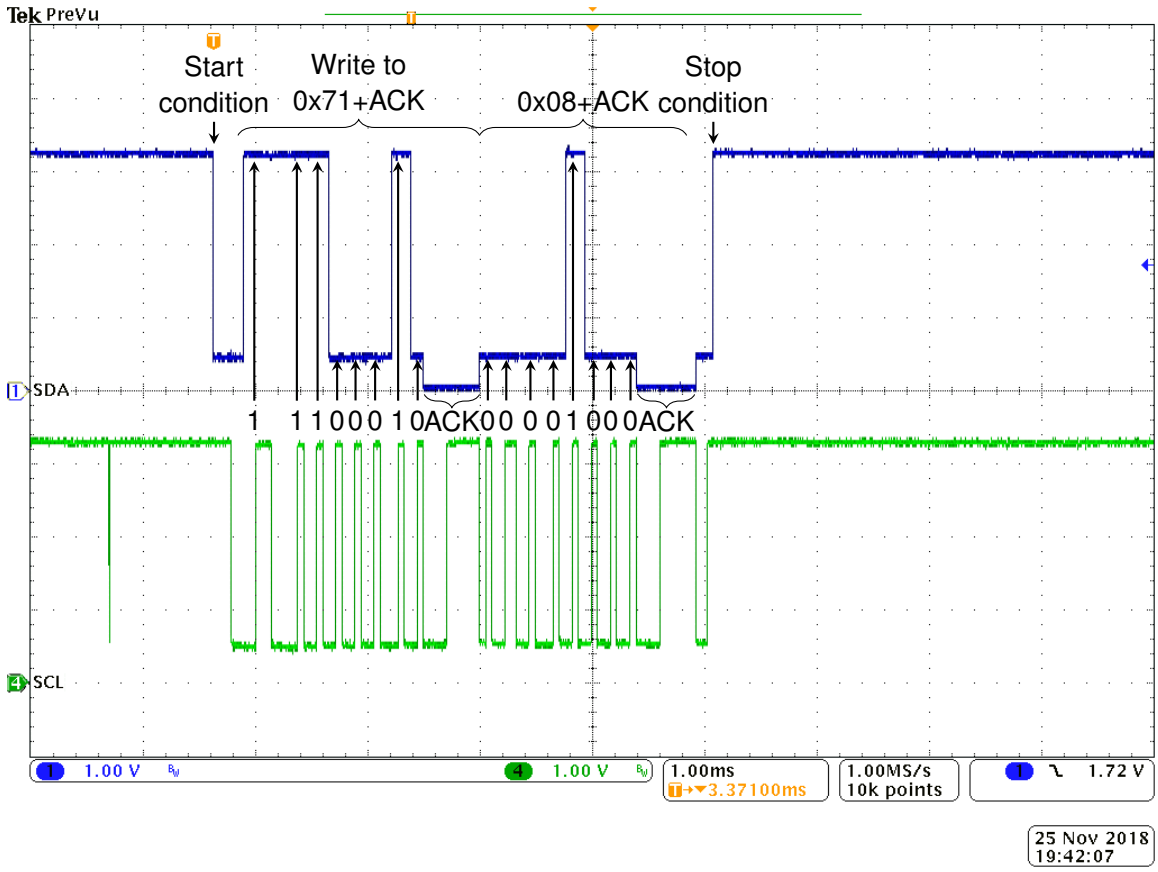


Figure 5.2: I^{2}C transaction

During tests by one of ARTIQ developers, one of the boards was damaged causing shorting of the I^{2}C bus to ground. This was suspected to be caused by ESD on I^{2}C lines, so Transient Voltage Suppressor (TVS) diodes were added in the next revision.

5.2. EEM connectors

For testing all EEM connectors a Vivado 2016.4 project with Virtual I/O (VIO) was used. All power and signal lines connected to EEM connectors were checked to make sure, that they are connected to appropriate source (FPGA, I^{2}C bus, 12 V or 3.3 V power supply). Virtual I/O allows a user to toggle specific signals connected to this IP core using Vivado IDE. User interface is shown in figure 5.3. From VIO menu there were options to toggle the direction of LVDS buffers on FPGA and to toggle output signal. Possible options for output signal were: logic 0, logic 1, high impedance, 50 MHz, 125 MHz and 200 MHz clock). The code used in this test is in appendix D.1.

Block schematic of test setup is shown in figure 5.4 and test setup is shown in figure 5.5. On the left side, an RJ45 EEM is shown. This is a simple module, which occupies two EEM connectors. It

[11] ARTIQ support for Kasli was developed by M-Labs.

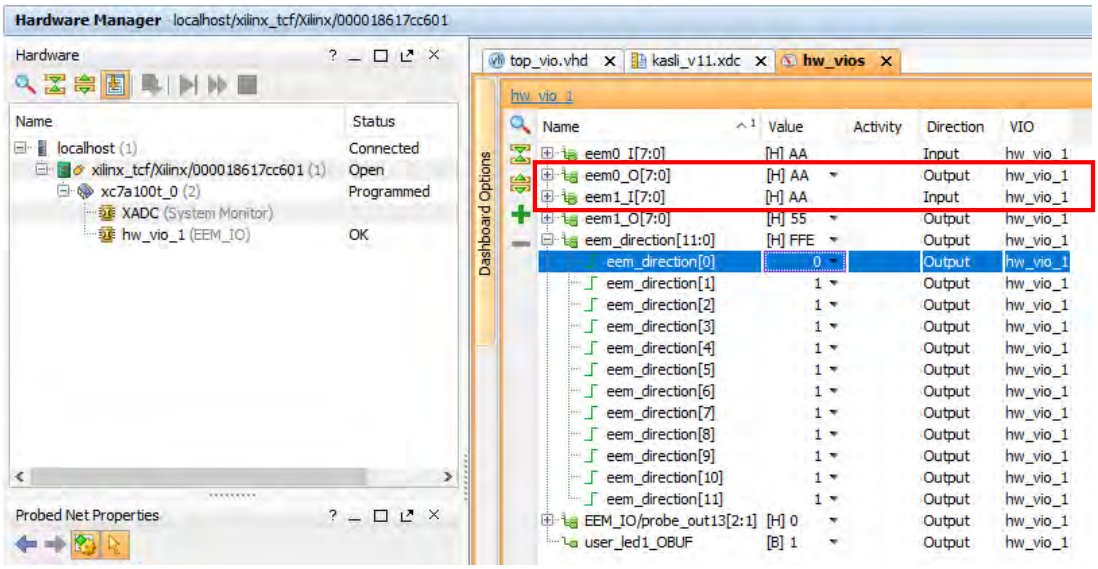


Figure 5.3: Virtual I/O user interface

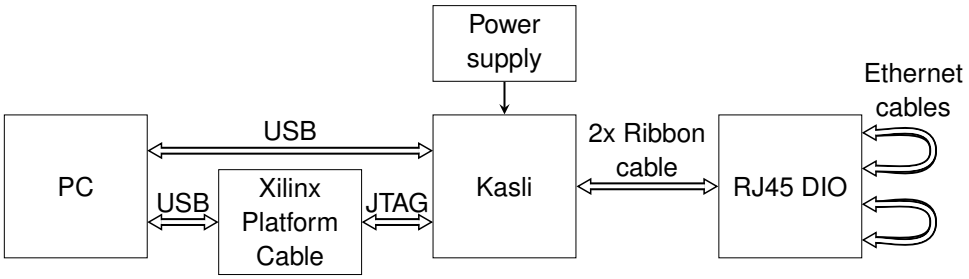


Figure 5.4: Block schematic of the test setup

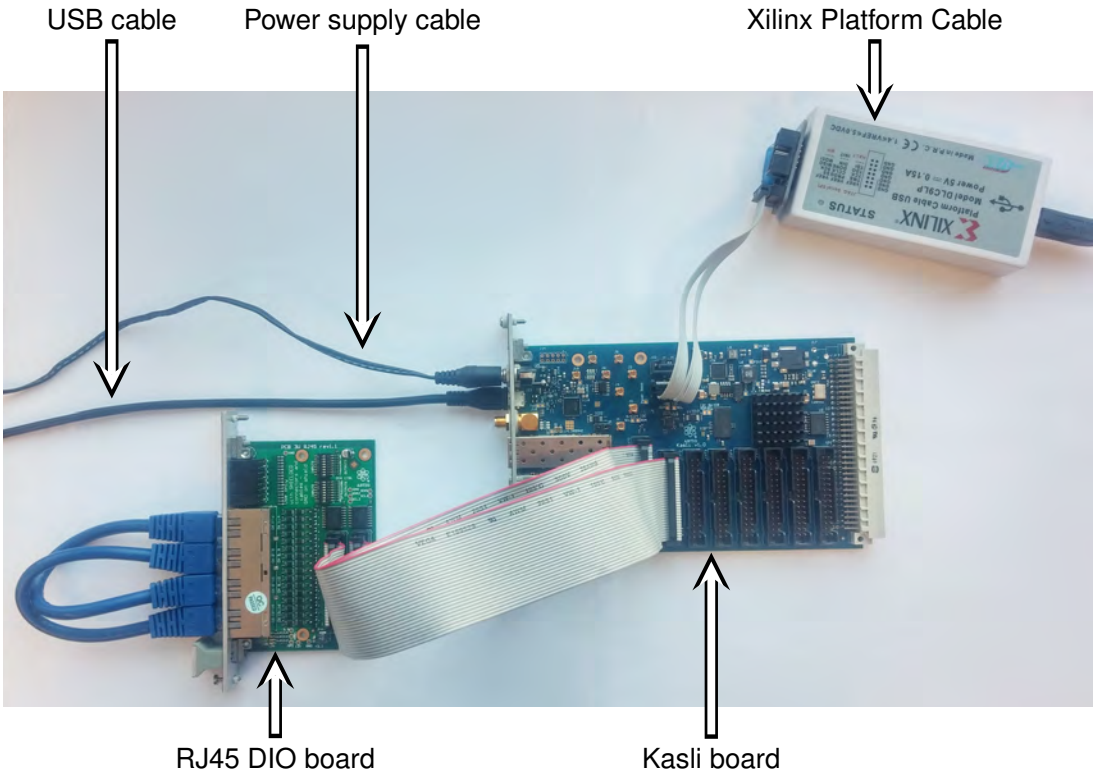


Figure 5.5: Test setup with RJ45 module

had signals looped back with Ethernet cables. EEM was connected to two extension connectors on Kasli. 8 signals on EXT0 were set as outputs, and 8 signals on EXT1 were set as inputs. PyFtdi I^2C script was used to set I/O direction on RJ45 EEM. Then the output of one extension connector was looped back to another connector and results were seen in the VIO interface (shown in red rectangle in figure 5.3). Alternating "0" and "1" signals were set on eem0 lines and were seen on eem1 lines (shown as "AA" in hexadecimal for all 8 lines). Then the RJ45 module was connected to the next two EEM connectors. Backplane adapter was used to check lines on backplane connector. If an LVDS pair was not connected properly, then it would not change in VIO. If I^2C interface did not work properly or 3.3 V power supply was not connected, then it would not be possible to set direction on RJ45 module. Finally, if 12 V power supply was not connected, then the RJ45 module would not work, and no signals could be seen on receiving lines. This test could not be done by simply connecting two EEM connectors together, because this way I^2C lines and power supplies on both connectors could not be tested.

Later an ARTIQ script was used (shown in appendix D.2), which automated testing process. In this script, EEM location was automatically detected via I^2C – based on the presence of acknowledge of the I^2C transaction from slave devices. Then the script tested lines, by setting all outputs high, then low, then alternating high and low to check for stuck lines.

5.3. Multi-gigabit lines

Multi-gigabit lines were tested using Integrated Bit Error Rate Tester (IBERT). This IP core from Xilinx automatically instantiates GTP block, inserts appropriate buffers and logic structures. Additionally, some status signals are wired to a debug hub and are visible when using programmer during an IBERT test. A user only needs to configure the IP core using a graphical wizard and instantiate a module in VHDL/Verilog, connecting clock signal and all GTP signals. Code doing this is shown in appendix D.1. Two SFP ports were connected together using SFP cable and SATA and SFP port were connected using SATA-SFP adapter [35] (see figures 5.6 and 5.7).

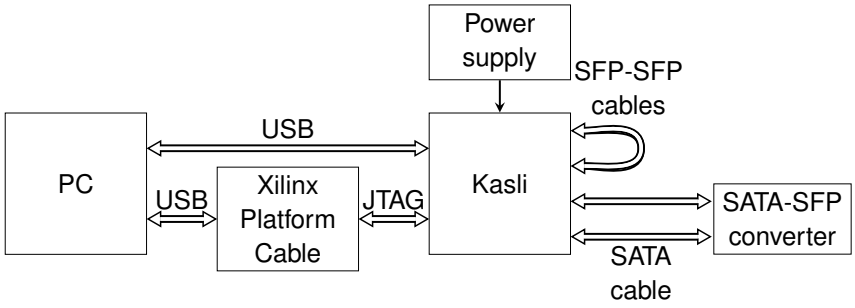


Figure 5.6: Block schematic of the test setup

Figure 5.8 shows GUI of Vivado when it is connected to device with working IBERT. Vivado can automatically detect, which ports are connected together. They are shown as Link 0..3. A user can select different testing patterns, pre- and post-emphasis and voltage swing used. IBERT sends a test pattern on one port and checks if the signal arriving at other port is correct. Then it calculates the bit error rate, which is visible in real-time to the user. Expected number of errors is as many as the user injected. Injecting errors allows checking if links are truly connected. One error to each pair was injected, as can be seen in figure 5.8.

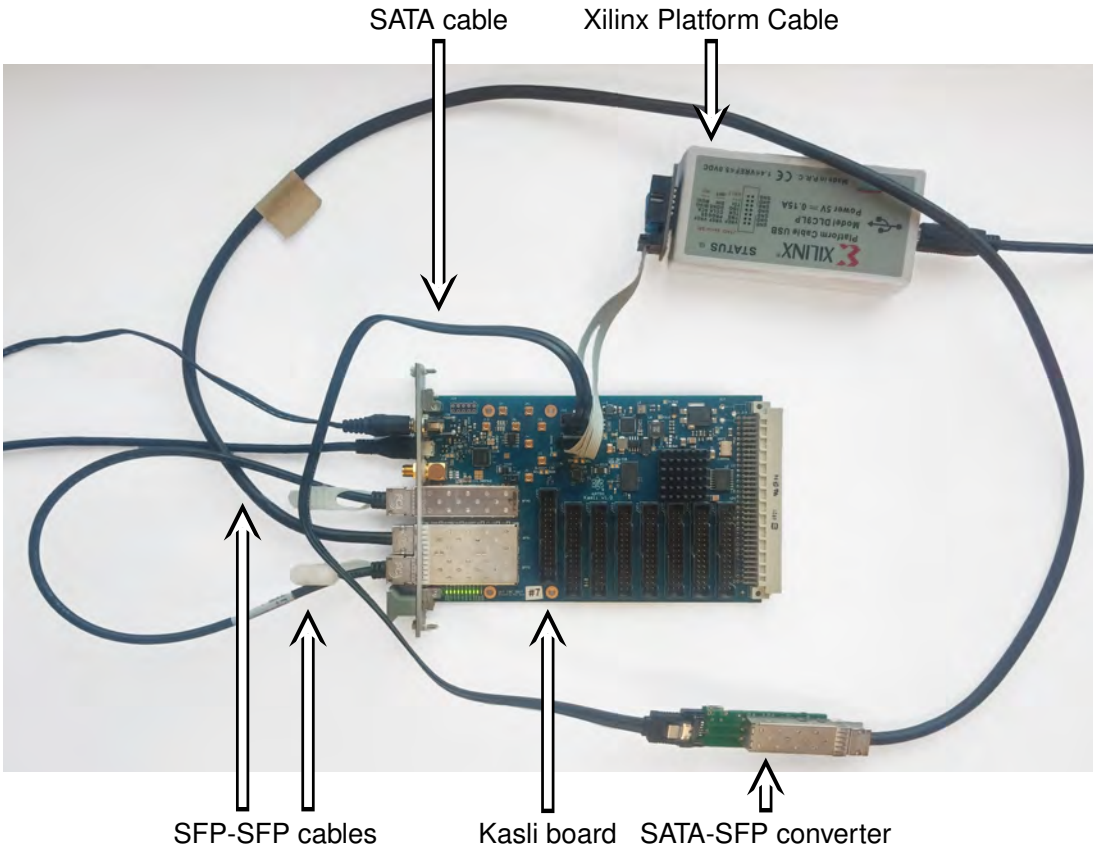


Figure 5.7: Test setup for IBERT

Name	TX	RX	Status	Bits	Errors	BER	BERT Reset	TX Pattern	RX Pattern	TX Pre-Cursor	TX Post-Cursor	TX Diff Swing
Ungrouped Links (2)												
Link 2	MGT_X0Y5/TX	MGT_X0Y6/RX	6.250 Gbps	1.104E12	1E0	9.061E-13	Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	959 mV (1100)
Link 3	MGT_X0Y6/TX	MGT_X0Y5/RX	6.250 Gbps	1.104E12	1E0	9.061E-13	Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	959 mV (1100)
Link Group 0 (2)							Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	959 mV (1100)
Link 0	MGT_X0Y4/TX	MGT_X0Y7/RX	6.250 Gbps	1.104E12	1E0	9.06E-13	Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	959 mV (1100)
Link 1	MGT_X0Y7/TX	MGT_X0Y4/RX	6.250 Gbps	1.104E12	1E0	9.062E-13	Reset	PRBS 7-bit	PRBS 7-bit	0.00 dB (00000)	0.00 dB (00000)	959 mV (1100)

Figure 5.8: IBERT tester

5.4. DDR3 memory

DDR3 memory was tested with a built-in ARTIQ memory test. Kasli board was programmed with ARTIQ firmware. Block schematic of test setup is shown in figure 5.9. It writes various patterns or random data and reads them back. ARTIQ then performs read leveling and optionally write leveling for each DQS group. Write leveling is not needed on Kasli. Memory tests only show "Memory test passed" on UART (console) output in this revision of ARTIQ if no errors were encountered. Listing 5.1 shows console output during SDRAM tests and leveling. No errors were encountered over 100 power cycles.

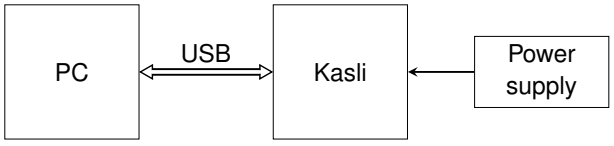


Figure 5.9: Block schematic of the test setup for SDRAM test

Listing 5.1: ARTIQ memory test

```
Initializing SDRAM...
Read leveling scan:
Module 1:
00000000001111111111100000000000
Module 0:
00000000000111111111110000000000
Read leveling: 15+-5 16+-5 done
SDRAM initialized
Memory test passed
```

5.5. Jitter attenuator circuit

Si5324 chip was programmed with scripts based on PyFtdi. A library was written for programming Si5324. It used the same code for the I^2C bus as one used in EEM connectors tests. This library handled writing appropriate values to registers in Si5324. After setting input clock frequency, it also calculated clock divider values given desired output frequency.

Clock fanouts were tested by setting various frequencies and checking each clock output (MMCX and ones going to FPGA) with an oscilloscope. The frequency set in the script was expected on all clock outputs. Measurement setup is shown in figure 5.10 and signal is shown in figure 5.11.

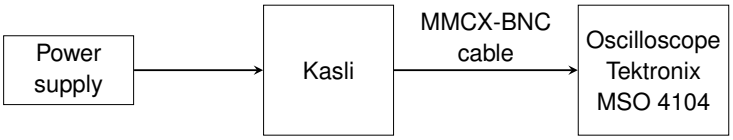


Figure 5.10: Block schematic of test setup with an oscilloscope

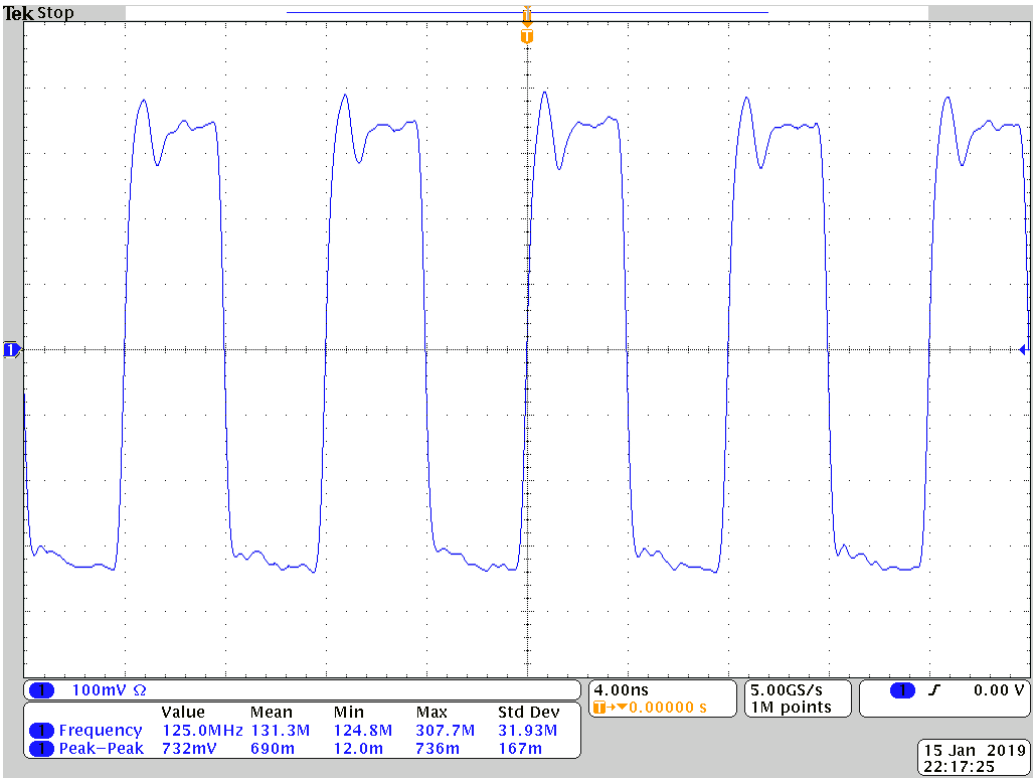


Figure 5.11: MMCX clock output signal

Additionally phase noise of clock outputs at 125 MHz was measured with a Signal Source Analyzer (SSA). The phase noise of clock output there had a peak at 450 kHz. This was traced to be noise from buck converter, as it works on the same frequency. In figures 5.12 and 5.13 a measurement setup for phase noise measurement is shown. Measurement from J1 connector (which showed the lowest RMS jitter measured) is shown in figure 5.14 and all RMS jitter values are in table 5.1. Jitter was calculated by the SSA. All measurements were taken from 10 Hz to 40 MHz from carrier frequency.

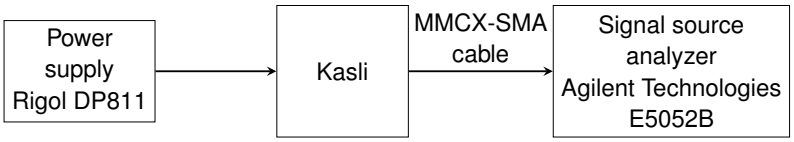


Figure 5.12: Block schematic of test setup

Figure 5.13: MMCX clock output phase noise measurement setup

Figure 5.14 also shows the same measurement from MMCX connector used on the backplane adapter. 450 kHz peak is very large there, and overall jitter is dramatically higher. This is caused by routing this signal in the neighborhood of noisy circuits, like power supply and the FPGA. Also, the passage of a differential signal through backplane connector and back to another board is less than ideal, to say at least.

5.6. JTAG and flash memory

JTAG and flash memory were tested to check if it was possible to program the FPGA and to write bitstream to flash memory using Xilinx Platform Cable and using the USB connection. After program-

Table 5.1: RMS jitter values of clock outputs measured on Kasli v1.0

Clock output	RMS Jitter
J1	0.63 ps
J2	0.64 ps
J3	0.65 ps
J4	0.64 ps
J5	0.68 ps
J6	0.64 ps
J7	0.65 ps
MMCX connector on the backplane adapter	2.91 ps

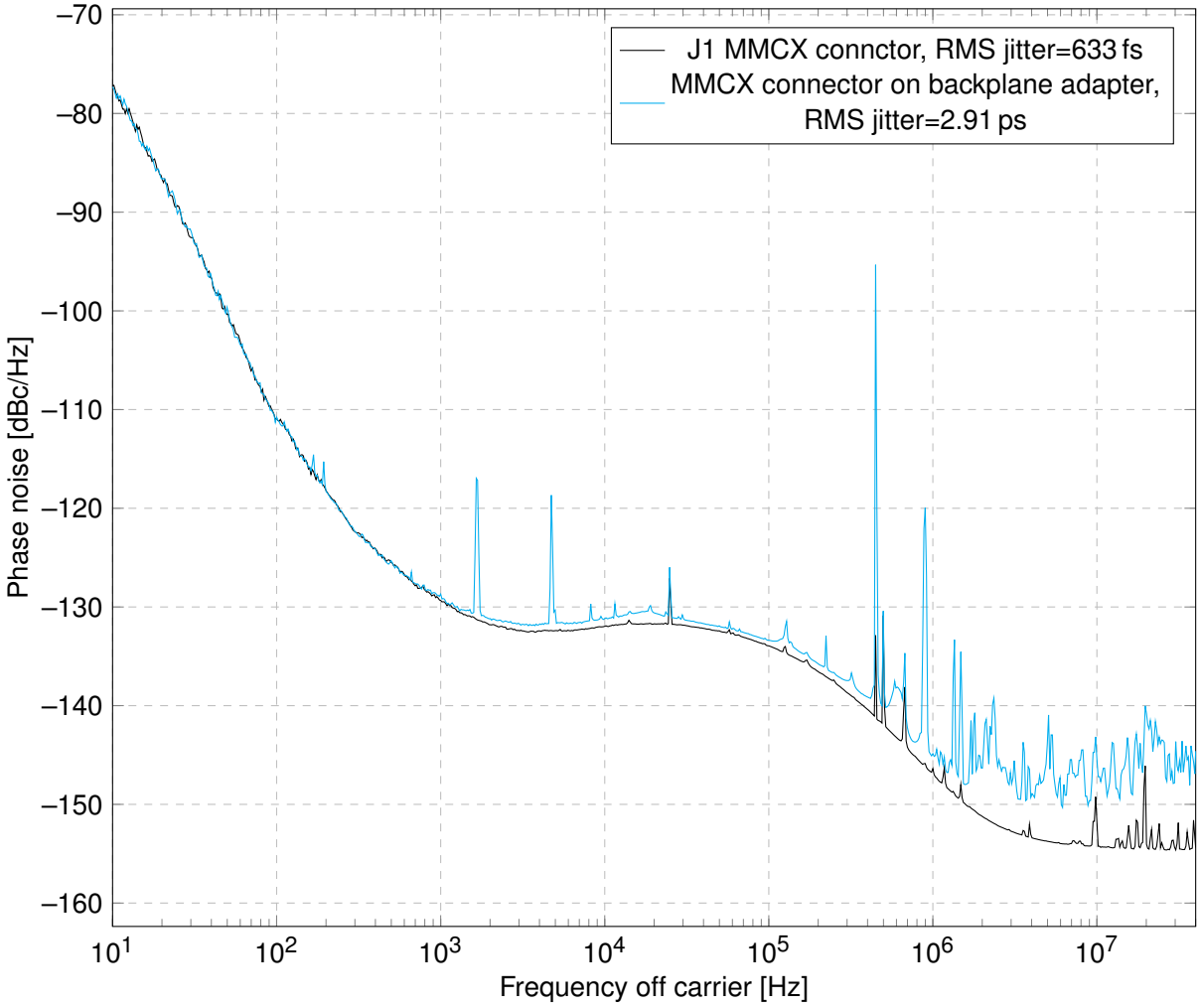


Figure 5.14: Phase noise comparison between MMCX connector on board and connector on backplane adapter board

ming the FPGA, the "DONE" pin didn't go high which caused Vivado to warn that the FPGA was not programmed. This was traced to too high pull-up resistor (R57) value. After R57 was changed to 300 Ω FPGA was programmed successfully. Next, from a synthesized bitstream, a flash memory configuration file was generated using Vivado IDE. Flash memory was then programmed (using x4 SPI). Then, after a power cycle, FPGA downloaded bitstream from flash memory and programmed itself, which showed that both JTAG and flash memory work as intended.

JTAG programming via USB was tested using OpenOCD. OpenOCD is an open-source program which provides on-chip programming and debugging. It supports the FT4232H JTAG interface. Two problems were found with JTAG programming via USB. The direction of TMS and TDO lines on level translating buffer (IC19) was swapped. This problem was scheduled for a fix in v1.1. Temporary fix for this buffer was needed, as direction pins were connected directly to ground and 3.3 V power supply. The buffer was replaced with 1 kΩ resistors bridging 3.3 V FTDI and 2.5 V FPGA interface. This ensured that there would not be an excessive current flow into the FPGA. All JTAG lines had pull-up resistors on FTDI side which ensured proper high-level voltage. After these fixes, programming the FPGA and flash memory worked. For tests an `artiq_flash` script was used. This script allows the user to download an ARTIQ image to flash memory of the device and boot FPGA from it. Console output from programming procedure is listed in appendix E.

5.7. XADC

Artix-7 FPGA contains an on-chip ADC (called XADC), which can measure external voltages, internal power supply rail voltages and die temperature. On this board, XADC provides information about voltage rails supplied to the FPGA and about its temperature. A user interface of the system monitor showing these parameters is shown in figure 5.15. In the window "Add Sensor" all available voltages are listed.

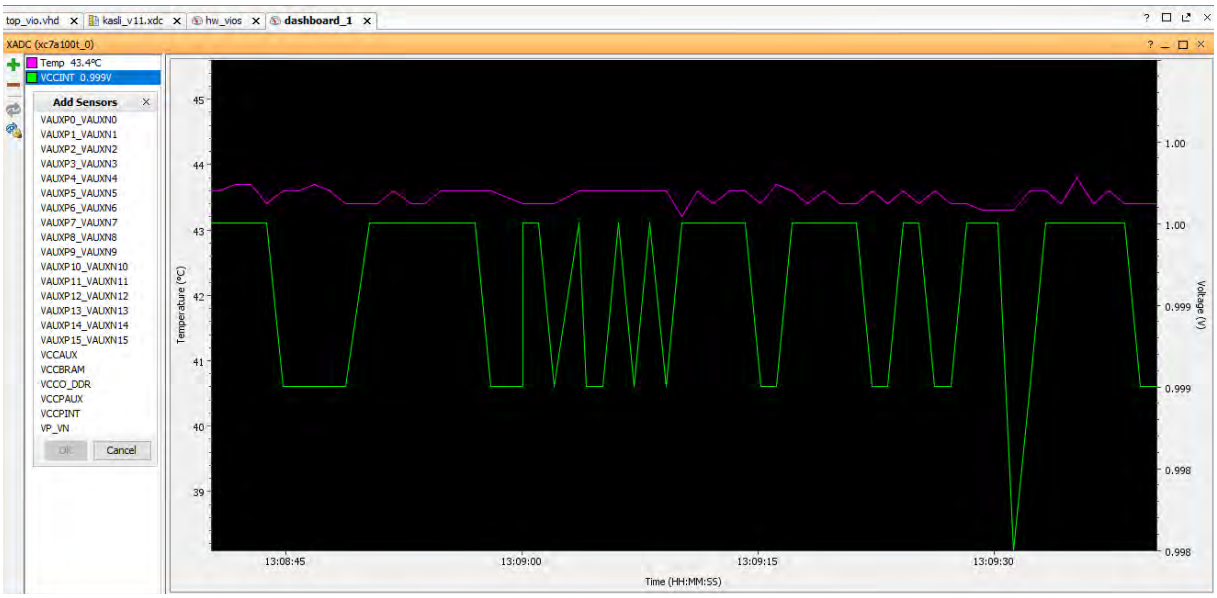


Figure 5.15: XADC user interface

When too many LVDS buffers were set to output, XADC measured high temperatures and additionally, FPGA was hot to touch. This meant that the FPGA needs a heatsink even with forced

cooling. During testing in the laboratory of one of ARTIQ developers, the FPGA overheated and was damaged. There was also a problem with XADC values, after some time of the FPGA power-on. Reading would show only zeroes (see figure 5.16). This was traced to VREFP and VREFN lines of FPGA being separated with a capacitor (C49)[12]. After desoldering it and putting a jumper instead XADC worked correctly.

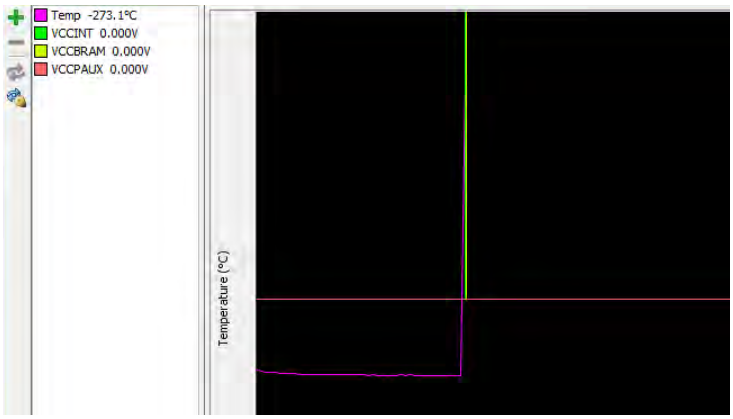


Figure 5.16: XADC interface with erroneous readings

5.8. UART

UART was tested first with a simple loopback in the gateware used for IBERT and EEM connector tests. A script was used which sent 100 000 characters and checked if the same characters were received. Additionally, ARTIQ sends debug info on UART, which can be seen during each boot. Example output is shown in appendix F. Over 100 power cycles with ARTIQ firmware were made with no visible malfunctions of UART interface.

5.9. Summary

Overall, there were 5 hardware bugs found during tests:

1. too high resistance of pull-up resistor on DONE pin,
2. I^2C on FT4232H channel, which doesn't support MPSSE,
3. wrong direction on TMS and TDO lines on the level translator,
4. 450 kHz spur on Si5324 output,
5. XADC VREFP and VREFN connection.

Additionally TVS on I^2C lines should be added to prevent I^2C EEM channels from being shorted to the ground due to transient voltage.

[12] "The XADC also has an on-chip reference option which is selected by connecting VREFP and VREFN to AD-CGND" [64]

6. Next revision of the PCB board

Development of Kasli did not stop with releasing first version of the board. Kasli v1.0 was meant to be a prototype run, where 10 boards were produced and tested in laboratories. Then, the next revision would be a product with fixed bugs of the previous version. Updated schematics are shown in appendix B.

6.1. v1.1 improvements

6.1.1. Jitter attenuator

After Kasli schematics were frozen a proposal came to make a clock distribution EEM (named Clocker) which could fan out at least 8 clocks. After this simple module was designed it became apparent that clock fanouts on Kasli are redundant, cause higher power consumption and hinder routing due to high signal density in the area of jitter attenuator. Clocker has a single purpose and is not a highly dense board which means that the phase noise of its outputs could be lower. This meant that ADCLK948 and some MMCX connectors could be safely removed from Kasli since Clocker fulfilled that role. Changes to simplify jitter attenuator schematic include:

- removing ADCLK948,
- removing an option of the external signal going to MMCX connectors,
- removing an option to have SMA connector as clock output,
- connecting backplane clock signal to the second output of Si5324,
- connecting four MMCX connectors to ADCLK944, marking those that have 180° phase,
- removing remaining MMCX connectors.

In figure 6.1 an updated jitter attenuator scheme is shown.

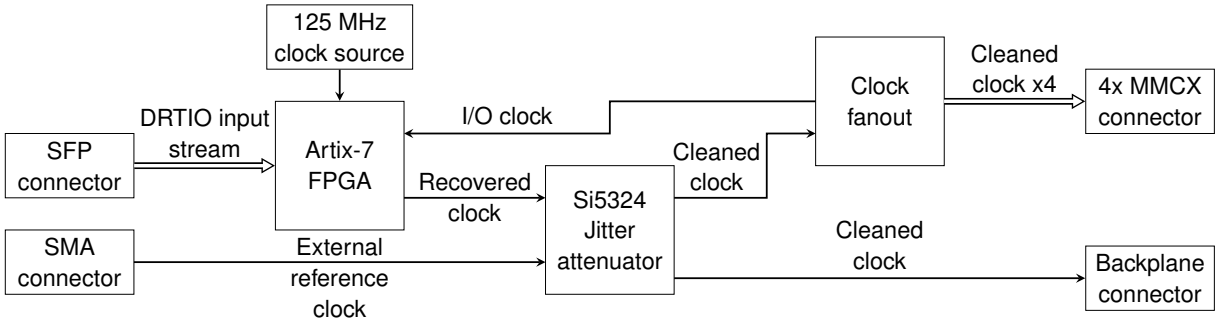


Figure 6.1: Block schematic of jitter attenuator scheme in Kasli v1.1

The 450 kHz noise, that was described in section 5.5, was caused by a buck converter. To suppress noise which propagated through 3.3 V power supply, a low-noise LDO was inserted – LT3045. Si5324 and ADCLK944 chips work with 2.5 V supply. LDO can be powered from 3.3 V, so no additional buck converter is required. LT3045 was chosen due to its high Power Suppression Rejection Ratio (PSRR), low dropout voltage and appropriate maximum output current. 0.8 V difference between input and output voltage isn't ideal as PSSR drops from 80 dB to 60 dB but it allows to make minimal changes in the existing layout of the board. The LDO has a precision current source on the

SET pin, which simplifies setting of an output voltage. 24 kΩ and 1 kΩ resistors were connected in series to SET pin to set 2.5 V output voltage. Additionally, a capacitor from SET to the ground was added to improve noise, PSRR, and transient response. Its value was taken from the datasheet as additional startup time is not critical in this project [7].

Another request was to float SMA and MMCX connectors from the ground by replacing jumpers with 100 kΩ resistors from PCB ground to connector ground. This was done to break ground loops. It wasn't considered for v1.0 since proper isolation would also require adding insulating washers to front panel SMA, which wasn't done in v1.0, but during the development of v1.1 a good quality washers were found for other boards and adding it to v1.1 was be a problem. The last request was to change transformer on SMA clock input to TC2-1TX+ [46] and include it in the default variant of the board to galvanically isolate clock input and to break the ground loop.

6.1.2. PCB layout

Changes made in the PCB layout include:

- Improving signal integrity for backplane LVDS lines. There are some cuts in reference plane for LVDS lines that were routed to the backplane, which could affect the signal integrity. This change is shown in figure 6.2.

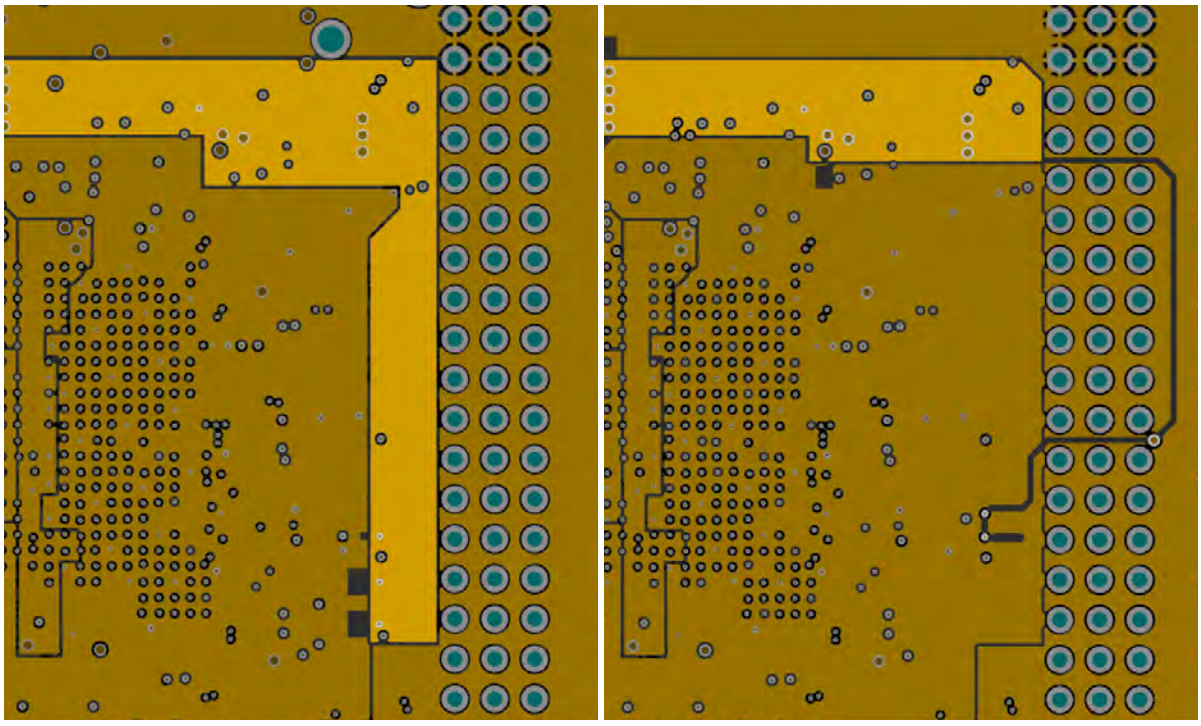


Figure 6.2: Difference between Kasli v1.0 (left) and Kasli v1.1 (right) in reference plane split. Layer 2 is shown, which is the reference plane for signals on layer 1 and 3.

- Adding missing teardrops – previously they were not generated for all pads.
- Setting up consistent rules for polygon pours so that they have the same clearance from various objects. Making these rules consistent makes it easier to spot troublesome parts of the board. Additionally, polygon pours needed a cleanup. This was just an aesthetic change. Both changes are presented in figure 6.3.

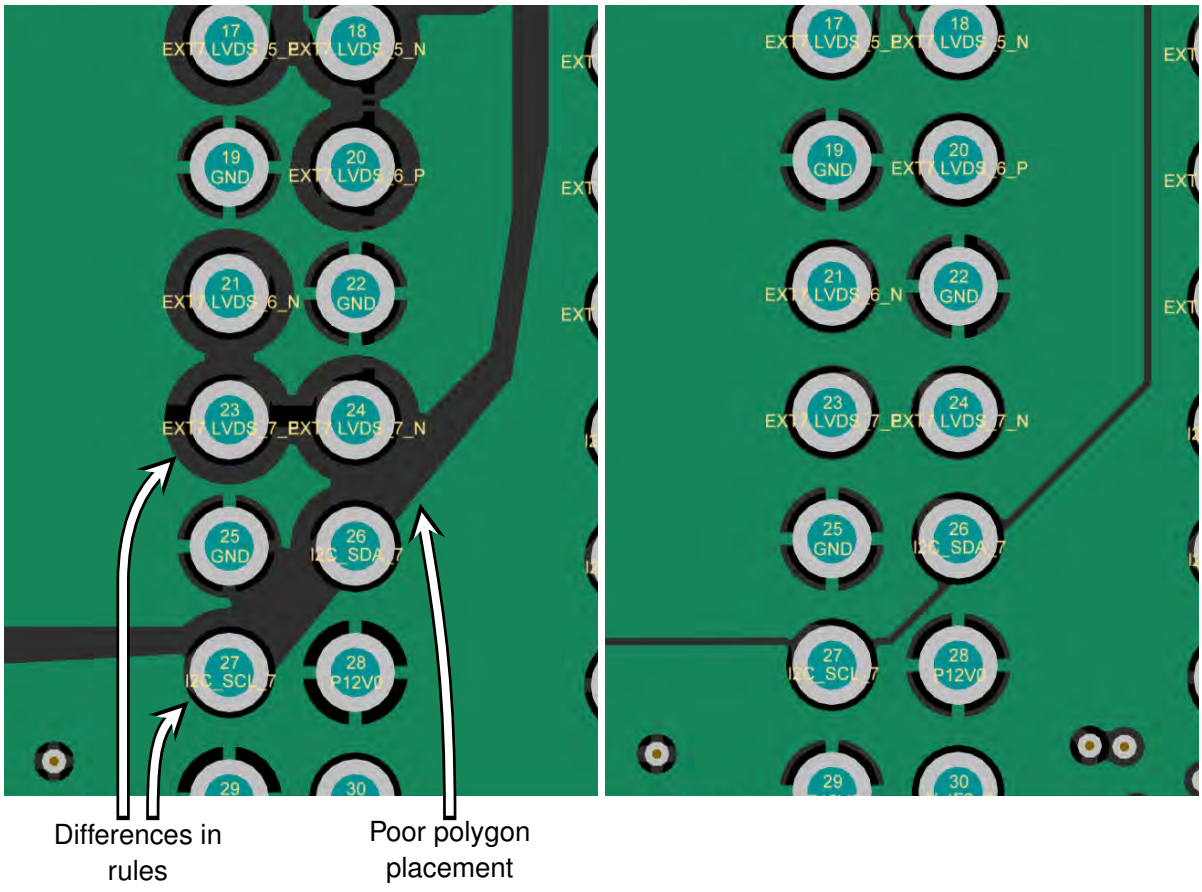


Figure 6.3: Example of polygon clearance rules and polygon placement in Kasli v1.0 (left) and Kasli v1.1 (right), green colour is copper

- Adding stitching vias to the edge of the board to minimize EMI. Stitching vias connect to ground on all layers and act like a Faraday cage for EMI that might be generated in inner layers.
- Removing the power connector which was facing the backplane and populating the backplane connector. This meant that unfiltered 12 V line did not have to pass through the whole board from front to back which greatly simplified routing and freed up some board space for other power supplies.

6.1.3. Other notable changes

Hardware bugs found during tests of Kasli v1.0, listed in 5.9, had to be resolved:

- Change resistance of pull-up resistor on the DONE pin to 300 Ω.
- Move I^2C bus on FT4232H to channel B, which supports MPSSE and UART to channel C.
- Change direction of TMS and TDO translation in IC19.
- Connect XADC lines VREFP and VREFN.

Additional changes that were requested:

- Add two LEDs connected to the FPGA.
- Add an ID chip (with 48-bit unique device ID) which will enable users to distinguish boards and may be used to set MAC address.

- Connect reset line of I^2C switches to one of GPIO lines of FT4232H.
- Move SFP LEDs closer to the panel, so that they are visible through the cutouts in the panel.
- Remove direct connection of MOD_DEF1 and MOD_DEF2 signals to the FPGA (instead only route them through I^2C lines).
- Add a heatsink for the FPGA.

After these changes were made, same simulations as in Kasli v1.0 were conducted with no significant differences in results.

6.2. Tests and measurements

Same tests that were made for v1.0 were used in this version. In figure 6.4 a manufactured board is shown.

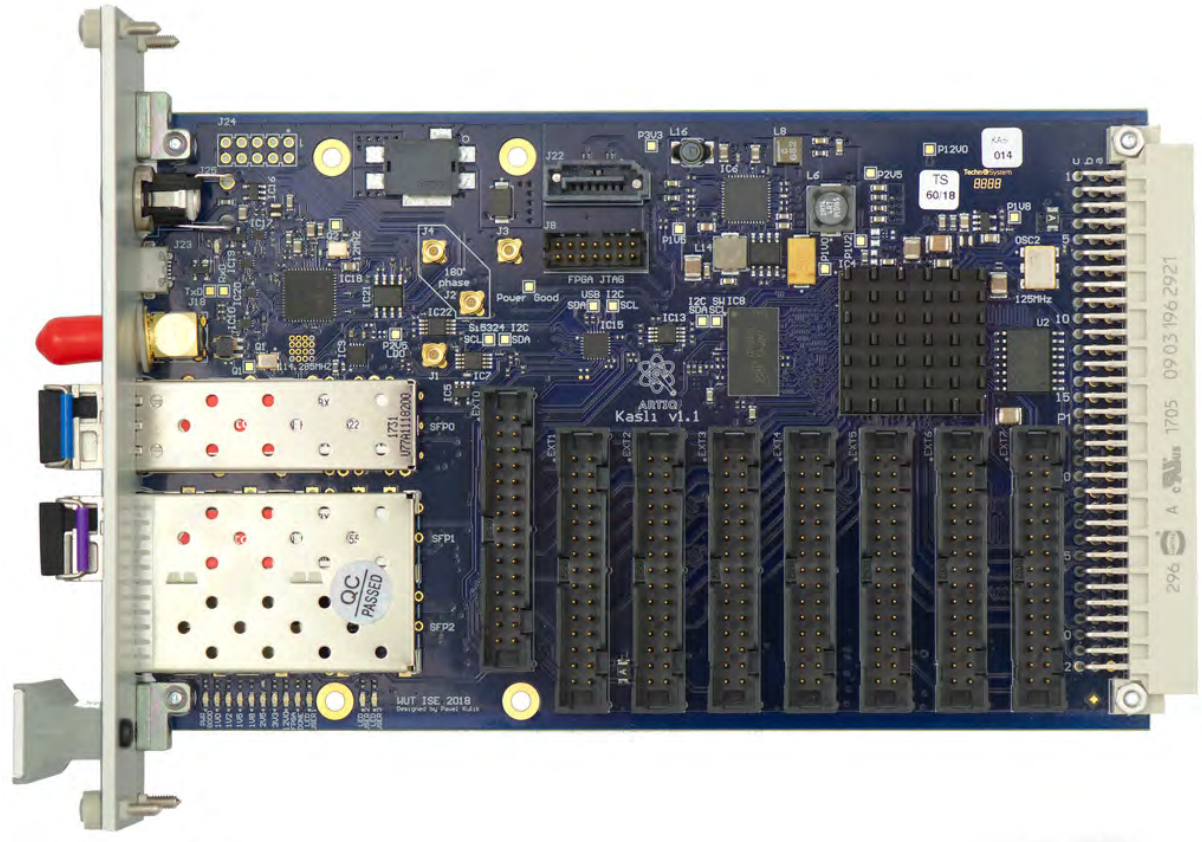


Figure 6.4: Manufactured board, version v1.1

Jitter attenuator circuit was one area of the board where significant changes were made, due to an additional separation of power supplies and removal of ADCLK948 fanout. Clock outputs were measured using oscilloscope and SSA, as in section 5.5, to check performance. Figure 6.5 shows clock signal generated by Kasli on one of MMCX connectors. All clock outputs measured in this section have 125 MHz signal, unless noted otherwise. In figure 6.6 a phase noise plot of this output is shown, compared with the output of Kasli v1.0. It's visible, that while peaks at 450 kHz are indeed lower in v1.1, overall phase noise is higher than in Kasli v1.0, particularly in the lower end of

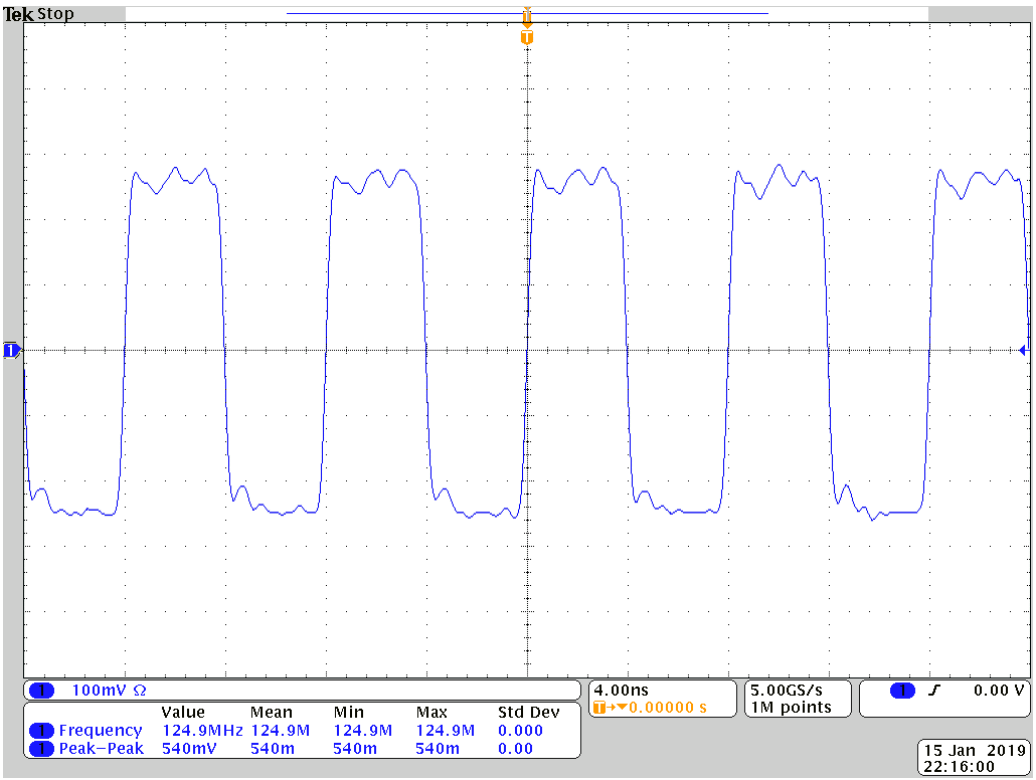


Figure 6.5: MMCX clock output signal

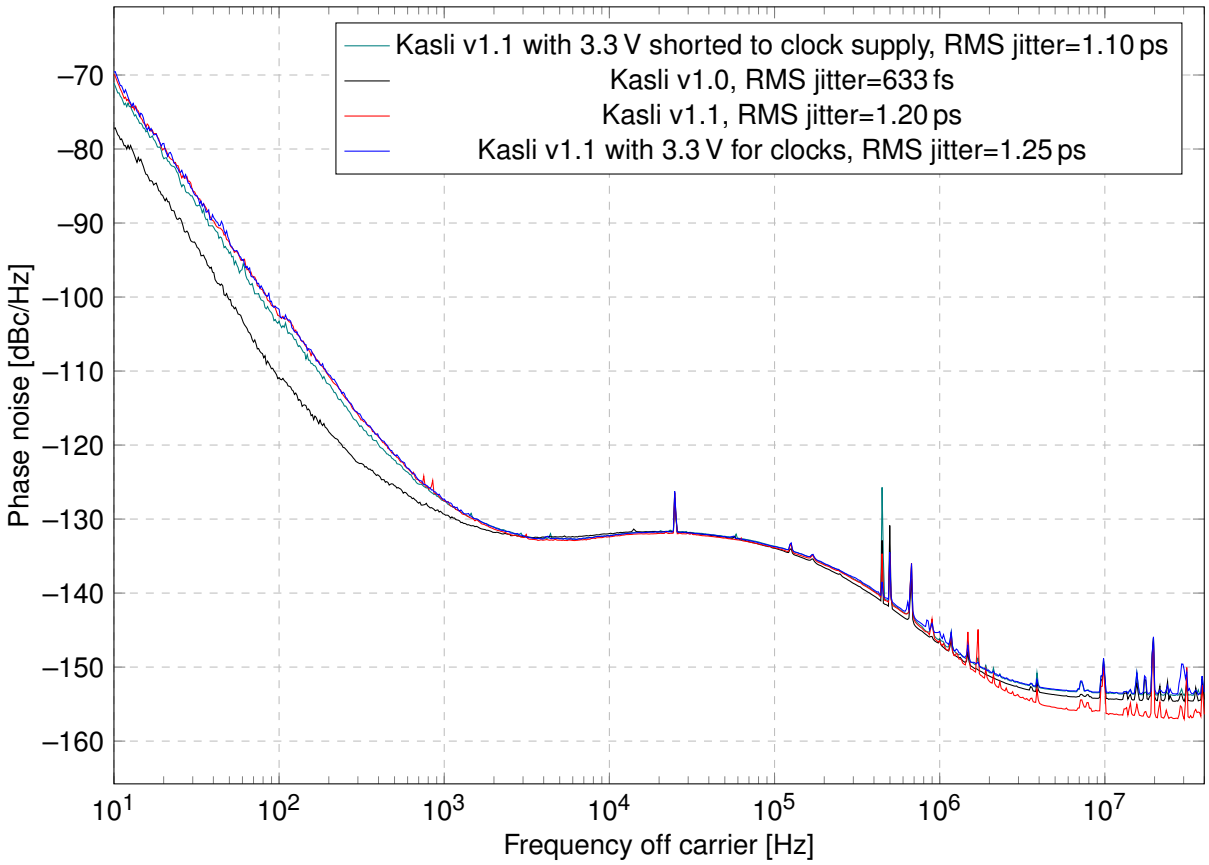


Figure 6.6: Phase noise comparison between Kasli v1.0 and v1.1

frequencies. Table 6.1 compares RMS jitter values[13] between Kasli v1.0 and v1.1 and also shows, that clock outputs are noisier in a newer revision. Both measurements were taken using a built-in oscillator as a reference.

Table 6.1: Comparison of jitter values of clock outputs measured on Kasli v1.0 and v1.1

	RMS Jitter	
Clock output	**Kasli v1.0**	**Kasli v1.1**
J1	0.63 ps	1.20 ps
J2	0.64 ps	1.17 ps
J3	0.65 ps	1.16 ps
J4	0.64 ps	1.17 ps
J5	0.68 ps	–
J6	0.64 ps	–
J7	0.65 ps	–
MMCX connector on the backplane adapter	2.91 ps	1.76 ps

The initial theory was that this could be caused by a lower voltage supplied to Si5324 and ADCLK944. Si5324 datasheet states, that rise/fall times for CMOS output are higher with V_{DD} = 1.71 V than with V_{DD} = 2.97 V (table 3 in [58]). No data is given for LVPECL or LVDS outputs, but one could assume that this relation holds true also for other output standards and is monotonic. This increases the slew rate of clock input of ADCLK944 and increases random jitter of clock outputs (figure 11 in [1]). The weak point of this theory is that only 1/f noise was higher and ADCLK944 datasheet states that random jitter is increased.

This theory was checked by increasing voltage supplied to both chips. Voltage was supplied to the LDO from an external power supply and SET resistors were changed so that 3.3 V is supplied to Si5324 and ADCLK944. A different, two-channel power supply was used for this measurement however it did not affect measurement in a significant way[14]. Switching power supply to 3.3 V didn't change performance significantly. Measured jitter was similar at 1.25 ps and it is still far worse than v1.0 performance. Higher drop on the LDO yields a better PSRR (visible in a lower value of 450 kHz peak) [7]. Noise floor at frequencies >1 MHz is higher, similar to v1.0. However, the noise in the 1/f range is still higher than in v1.0. Additional measurements with 22 µF tantalum capacitor on the SET pin didn't show any significant improvement. Shorting the LDO input and output pads (after desoldering the chip) also didn't change the performance, only the 450 kHz peak returned. This means that power supplies were not the issue.

Maciej Grzegrzółka[15] suggested, that higher 1/f noise could be caused by differences in oscillators. One on Kasli v1.1 could be exposed to too high temperature during the manufacturing process. To reduce the influence of oscillator, other measurements were made with high-quality external frequency source connected to SMA input. Measurements with this source are presented in figure 6.7. Block schematic of measurement setup is shown in figure 6.8. It's visible that peaks at 50 Hz and its harmonics are coming from this clock source. Kasli v1.0 and v1.1 performance is similar here.

[13] Computed by Signal Source Analyser.

[14] This was verified by measuring phase noise of unmodified Kasli v1.0 twice, once with DP811, the second time with DP832 power supply. RMS jitter was measured to be 646 fs and 655 fs, respectively, so no significant change came with different power supply.

[15] Maciej Grzegrzółka is from Microwave Circuits and Instrumentation Division (*Zakład Układów i Aparatury Mikrofalowej*), Institute of Electronic Systems, Warsaw University of Technology.

It's also worth noting, that small peak at around 25 kHz, as well as several other above 500 kHz disappeared when Si5324 had an external frequency source.

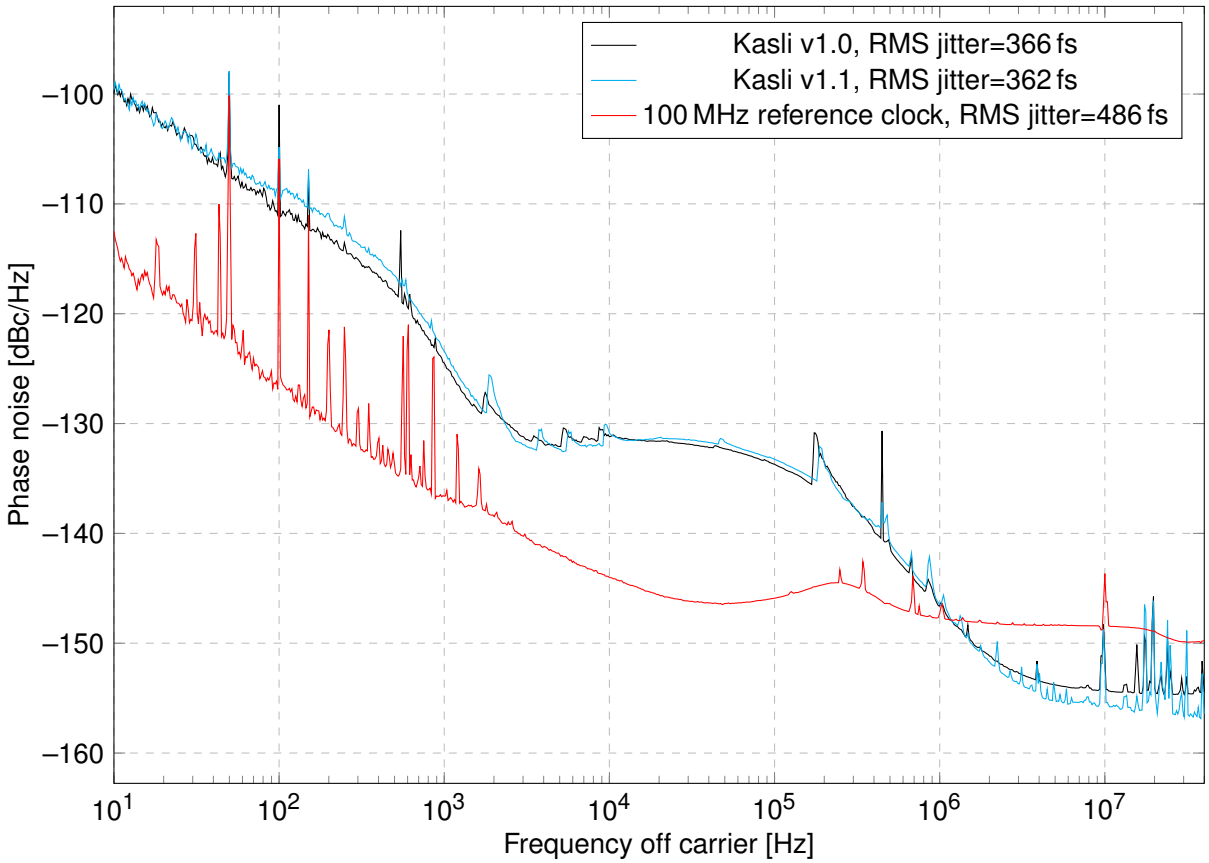


Figure 6.7: Phase noise comparison between Kasli v1.0 and v1.1 – external clock source

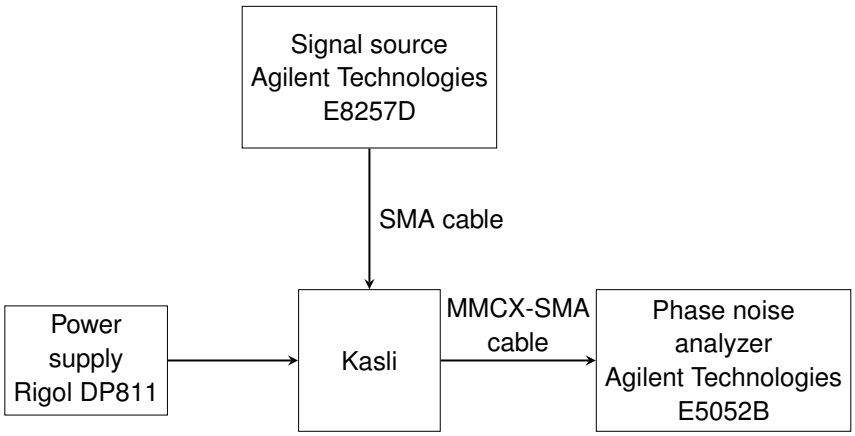


Figure 6.8: Block schematic of test setup

After these measurements were taken, another two v1.1 boards were measured. One of them exhibits similar performance to v1.0 board and the other similar to first v1.1 board measured. Figure 6.9 present these measurements. Boards #2 and #3 were measured in the exactly same conditions, which means that only part variation or possible overheating during the manufacturing process remain as possible explanations to this range of encountered performances. Changing Si5324 loop bandwidth didn't significantly affect the measurements. Crystal oscillators were not swapped between boards to check if they are the source of variation in phase noise. Heat required to desolder

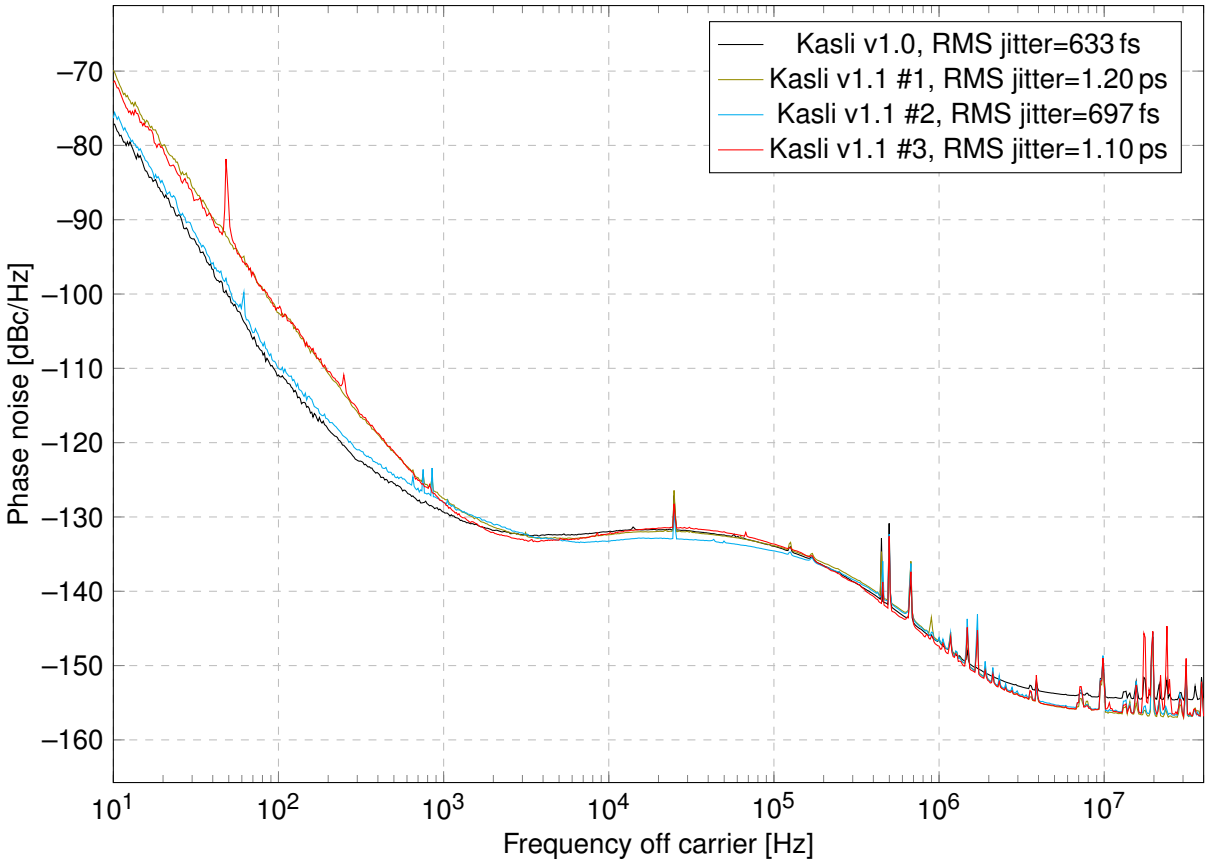


Figure 6.9: Phase noise comparison between Kasli v1.0 and v1.1 – additional boards

them from the boards could change their parameters, thus making the measurement pointless. The key takeaway from this is that Si5324 performance without external signal is heavily dependent on oscillator reference[16]. Board to board variations are quite high, so if high-quality clock signal from Kasli is needed in the experiment, then an external signal should be supplied, as it yields the best and most consistent results. Overall it seems that the only way to decrease this variance is to use a higher quality reference crystal or oscillator. Phase noise plots of all outputs are in appendix H.

Otherwise, all other tests were the same for this board and for v1.0. Without any hardware changes board could boot up after flashing ARTIQ, which means that DONE pin has appropriate resistance and JTAG lines have a proper direction set on level translators. XADC now doesn't show zeroes after some time of device usage.

After switching ports on FT4232H, it is able to drive I^2C lines with MPSSE however, this could not be tested with the current version of PyFtdi due to enable pin of level translating IC being connected to the same channel. PyFtdi currently doesn't support I^2C and GPIO simultaneously instead it drives all other outputs to a low level during I^2C transactions, disabling level translator in the process. Note that this is a limitation of PyFtdi not of the chip itself as it supports GPIO while using MPSSE engine [25]. Instead, an open source library with support for MPSSE was used. Code and steps to install it are in appendix D.3. Also resetting of I^2C switches (IC14 and IC15) was tested successfully, along with reading ID of EEPROM chip added in v1.1. Listing 6.1 shows output of the test code.

[16]Which is also mentioned in Si5324 datasheet: "Due to the low bandwidth capabilities of this part, any low-frequency wander or instability on the external reference will transfer to the output clocks. To address this issue, a stable external reference, TXCO, OCXO, or thermally-isolated crystal is recommended." [58]

Listing 6.1: Output of test I^2C testing code

```
Description: Quad RS232-HS
Serial: Kasli-v1.1-019
Writing 1 bytes to I2C slave 0x70
Writing 1 bytes to I2C slave 0x71
Reading 1 bytes from I2C slave 0x70
Reading 1 bytes from I2C slave 0x71
Before reset: Switch 0: 80  Switch 1: 08
Reset
Reading 1 bytes from I2C slave 0x70
Reading 1 bytes from I2C slave 0x71
After reset:  Switch 0: 00  Switch 1: 00
Writing 1 bytes to I2C slave 0x71
Reading register : 0xfa with 6 bytes
Writing 1 bytes to I2C slave 0x50
Reading 6 bytes from I2C slave 0x50
EEPROM EUI: 54-10-ec-aa-64-b2
```

7. Conclusion and outlook

During this project, a new real-time controller module was developed for laboratories doing quantum physics experiments which, together with extension modules, provides a modular design, adaptable to the needs of many laboratories. It is a unique solution as both hardware and software used for Sinara is completely open source. Kasli schematics and board files are published on GitHub[17].

Technical requirements listed in section 2.3 are fulfilled: Kasli is a standalone EEM controller compatible with ARTIQ which allows to connect up to 12 EEMs and has available timing resolution of 1 ns. Kasli is able to provide a clock signal with jitter values around single picosecond without external reference signal and below 400 fs when used with low-jitter external clock source as measurements described in section 6.2 has shown. RMS jitter variations between boards were investigated with a conclusion that standalone clock generation is highly dependent on crystal oscillator and that the best results can be obtained when using external reference signal. The clock generated by jitter cleaner circuit can be distributed up to 4 EEMs that require it using MMCX cables. If more modules require this clock reference signal then an additional clock fanout module (Clocker) has to be used. Clock recovery from input data stream can be used to synchronize multiple Kasli modules.

This module is in production in two companies and is currently used in at least 14 laboratories around the world according to the poll conducted by Joseph Britton, one of ARTIQ founders from the University of Maryland. The number of experiments using ARTIQ steadily rises and Kasli is now the most used controller as can be seen in figure 7.2. This number is only expected to rise, as the number of users planning to use ARTIQ/Sinara in 2019 is also rising (figure 7.1) and additional EEMs become available (figure 7.3) [14]. This shows that the goal outlined in section 2.2 was

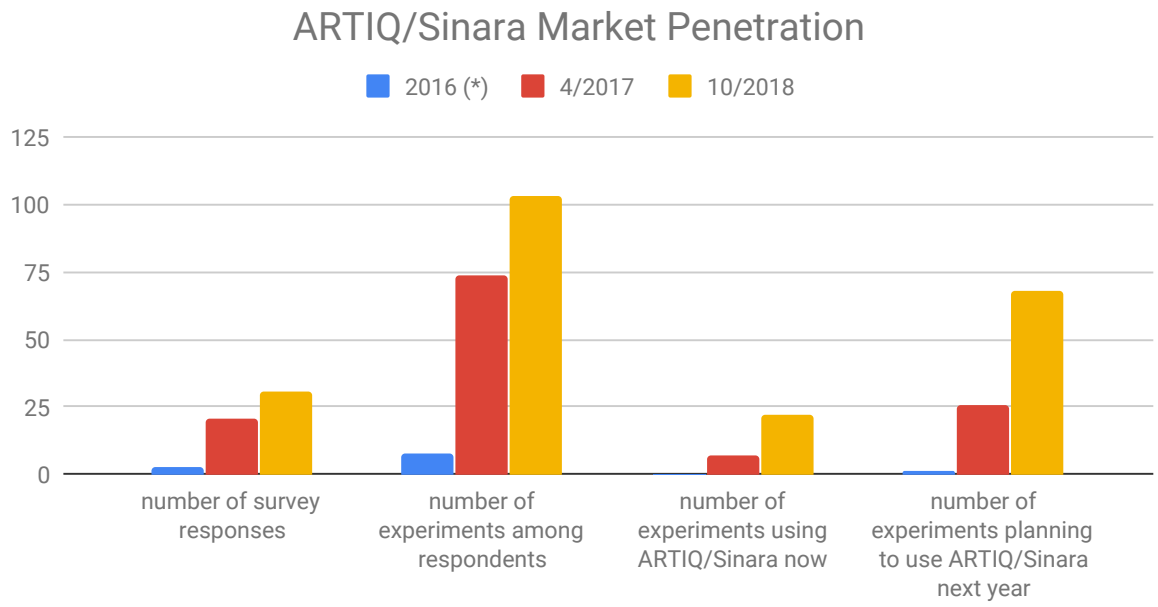


Figure 7.1: ARTIQ/Sinara Market Penetration[18]

[17] Repository address: `https://github.com/sinara-hw/kasli`

[18] Values in the first year are based on NIST usage in 2016.

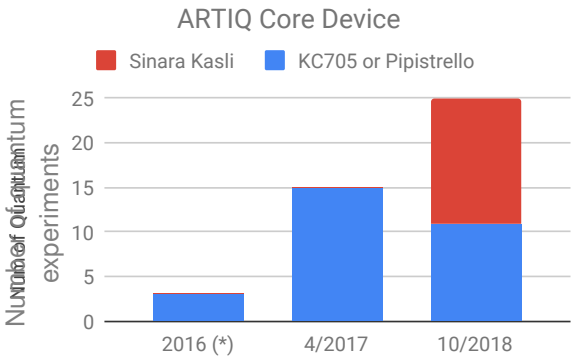


Figure 7.2: ARTIQ Core Device choice among respondents[19]

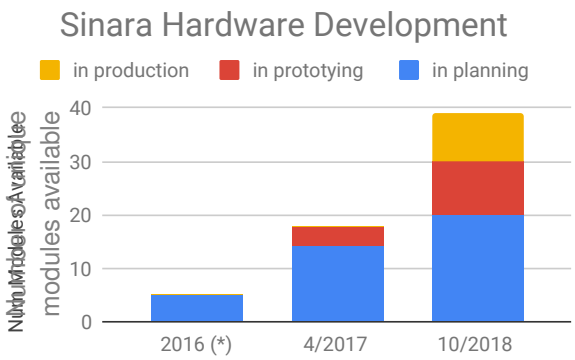


Figure 7.3: Sinara Hardware Development[19]

7.1. Sinara state at the end of the project

Sinara project grew during the development of this board, both in hardware and in software support. Now, on hardware side Sinara ecosystem offers:

- MicroTCA form factor:
  - Sayma – arbitrary waveform generator,
- 3U form factor:
  - Kasli – this controller,
  - Sampler – 1 MS/s 16-bit, 8 channel ADC,
  - Zotino – 32 channel 1 MS/s (shared across all channels) 16-bit DAC,
  - Urukul – 4 channel 1 GS/s DDS,
  - 3 digital I/O modules with BNC, SMA and RJ45 connectors,
  - Grabber – camera frame grabber input module,
  - Clocker – clock buffer and fanout module.

Additional modules are planned, including microwave synthesizer, temperature controller, feedback control module with 2 ADCs and 2 DACs and more. Software now supports all of the listed boards. Additionally, support for Sampler-Urukul servo (SU-Servo) was developed.

7.2. Kasli usage in experiment

Ion Trap Quantum Computing Group at University of Oxford described their usage of Kasli in experiments[20]:

- All but one experiments are now using ARTIQ with Kasli as the FPGA controller.
- Most of the experiments use more than one Kasli, connected together with fiber as master and slave and soon this will be extended to several daisy-chained Kasli boards in each experiment.
- Urukul board and Booster (RF amplifier, shown in figure 7.5) are used to drive AOMs, which modulate power, frequency, and phase of laser light.

[19] Values in the first year are based on NIST usage in 2016.

[20] Source: private communication with Thomas Harty, a research fellow from Ion Trap Quantum Computing Group at University of Oxford, 2018 and 2019

- Sampler-Urukul servos (also shown in figure 7.5) are used to stabilize the intensity of the laser.
- Urukul is used to generate the RF voltages that trap ions and to generate microwaves (by an external IQ mixer) which control qubits directly.
- Zotino is used to produce the DC trapping voltages (figure 7.6).
- Ion state is measured using PMTs (Photo-Multiplier Tubes) – photons emitted by the ions are counted using BNC DIO board connected to the PMT.
- DIO boards are also used for a number of other tasks e.g. controlling optical shutters and RF switches or monitoring trigger signals.

Setup of one of their experiment is shown in figures 7.4-7.6.

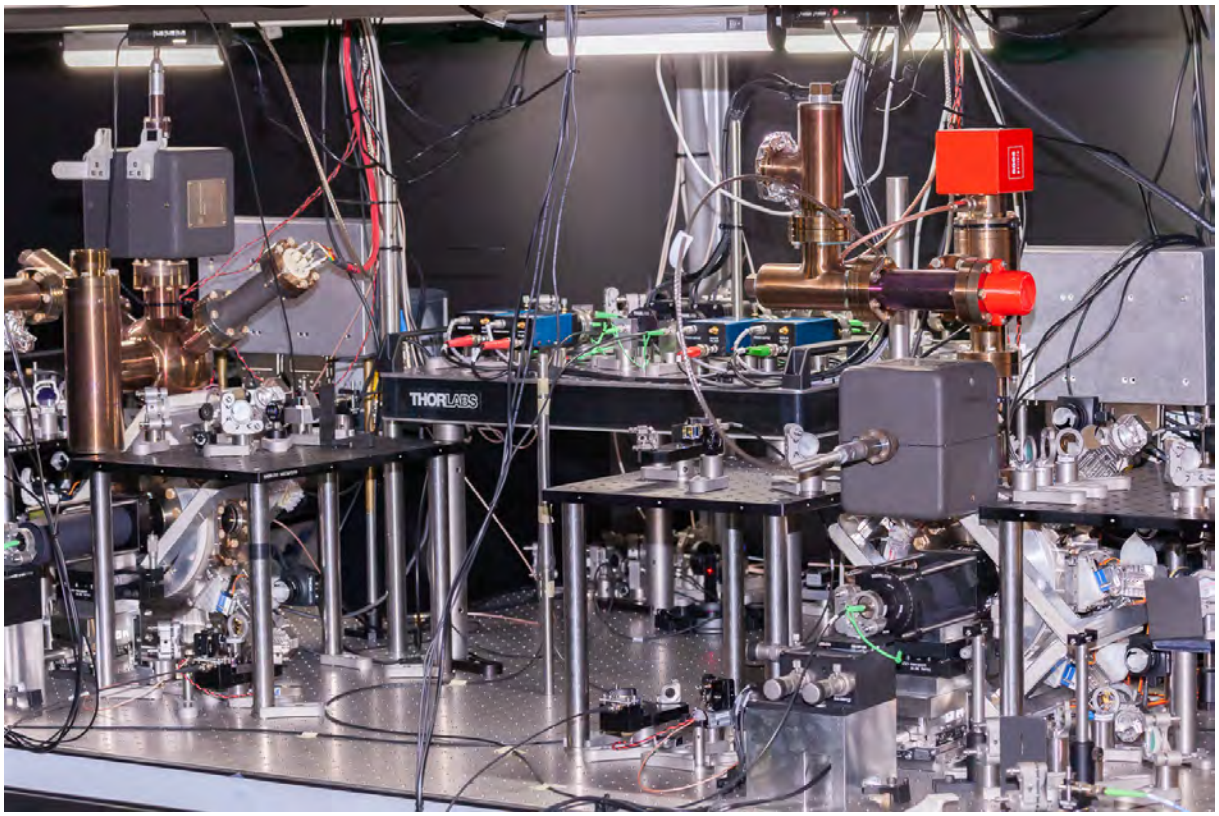


Figure 7.4: Setup overview

7.3. Further work

Planned next stages of this project include a new version of Kasli board with all 12 EEM connectors on the main board, without the current backplane connector. Most 3U racks support longer modules and this removes mechanical issues caused by this connector and backplane adapter. Additionally, there is a version with Zynq SoC (System on Chip) planned. Kasli and its modules could possibly use Compact-PCI backplane instead of ribbon cables. This could help with airflow around the Kasli module since ribbon cables obstruct it. Then, the community's work should concentrate on expanding hardware designs compatible with Sinara and Kasli.

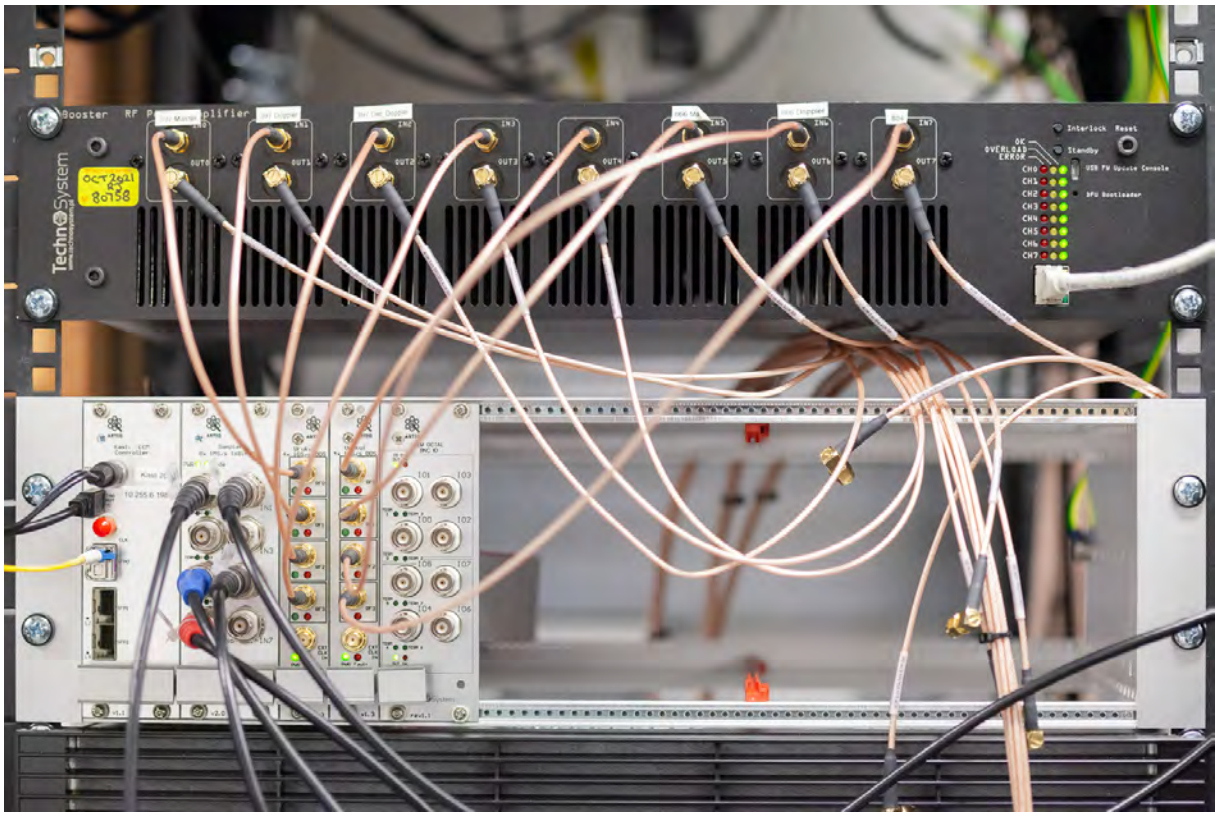


Figure 7.5: SU-Servo (lower, the leftmost board is Kasli, then Sampler ADC, 2 Urukul DDSs and the rightmost board is DIO) and Booster (higher)

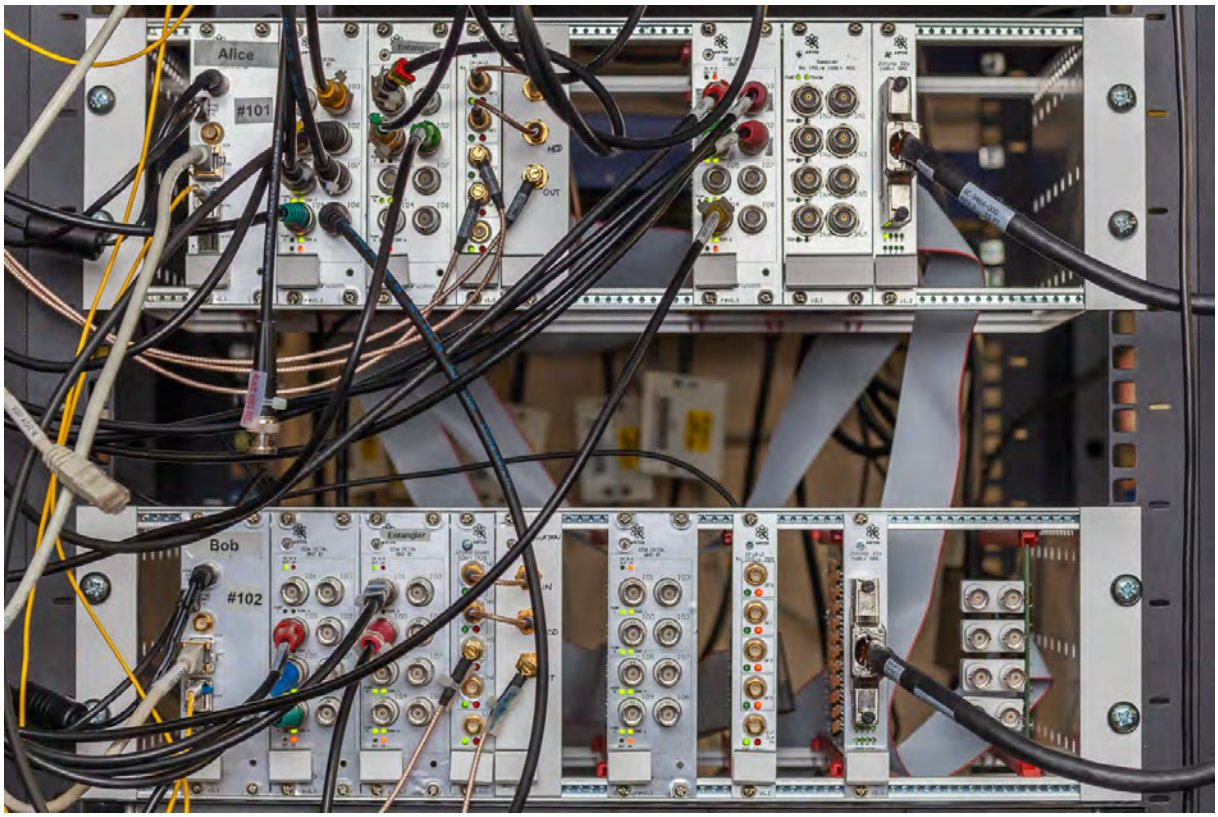


Figure 7.6: Urukul + Zotino

References

[1] Analog Devices, *ADCLK944 2.5 V/3.3 V, Four LVPECL Outputs, SiGe Clock Fanout Buffer datasheet*, Mar. 2010, URL: http://www.analog.com/media/en/technical-documentation/data-sheets/ADCLK944.pdf (visited on 29/08/2018).

[2] Analog Devices, *ADCLK944 IBIS model*, version 3.1, 16th July 2012, URL: https://www.analog.com/media/en/simulation-models/ibis-models/adclk944.ibs (visited on 11/11/2018).

[3] Analog Devices, *ADCLK948 IBIS model*, version 3.0, 23rd Mar. 2010, URL: https://www.analog.com/media/en/simulation-models/ibis-models/adclk948.ibs (visited on 11/11/2018).

[4] Analog Devices, *ADCLK948 Two Selectable Inputs, 8 LVPECL Outputs, SiGe Clock Fanout Buffer datasheet*, Aug. 2016, URL: http://www.analog.com/media/en/technical-documentation/data-sheets/ADCLK948.pdf (visited on 29/08/2018).

[5] Analog Devices, *ADP5052 5-Channel Integrated Power Solution with Quad Buck Regulators and 200 mA LDO Regulator datasheet*, July 2017, URL: http://www.analog.com/media/en/technical-documentation/data-sheets/ADP5052.PDF (visited on 30/08/2018).

[6] Analog Devices, *ADP505x Buck Regulator Design Tool*, version 1.134, 28th Sept. 2017, URL: http://download.analog.com/PMP/ADP505x_BuckDesigner.zip (visited on 30/08/2018).

[7] Analog Devices, *LT3045 20V, 500mA, Ultralow Noise, Ultrahigh PSRR Linear Regulator datasheet*, Oct. 2017, URL: https://www.analog.com/media/en/technical-documentation/data-sheets/3045fa.pdf (visited on 02/12/2018).

[8] R. Blatt and C. F. Roos, *Quantum simulations with trapped ions*, "Nature Physics" Apr. 2012 8, pages 277–284, DOI: 10.1038/nphys2252, URL: http://dx.doi.org/10.1038/nphys2252 (visited on 15/08/2018).

[9] Sergio Boixo et al., *Characterizing quantum supremacy in near-term devices*, "Nature Physics" 2018 14.nr. 6, page 595.

[10] Joppe Bos et al., *Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE*, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, ACM, Vienna, Austria 2016, pages 1006–1018, DOI: 10.1145/2976749.2978425, URL: http://doi.acm.org/10.1145/2976749.2978425 (visited on 15/08/2018).

[11] Sébastien Bourdeauducq et al., *ARTIQ documentation*, Jan. 2019, URL: https://m-labs.hk/artiq/manual-master/index.html (visited on 14/01/2019).

[12] Sébastien Bourdeauducq et al., *ARTIQ overview*, M-Labs, 2016, URL: https://m-labs.hk/artiq/index.html (visited on 23/08/2018).

[13] Sébastien Bourdeauducq et al., *m-labs/artiq: 3.6*, Mar. 2018, DOI: 10.5281/zenodo.591804, URL: https://doi.org/10.5281/zenodo.591804.

[14] Joseph Britton, *ARTIQ/Sinara Survey results*, 21st Sept. 2018, URL: https://ssl.serverraum.org/lists-archive/artiq/2019-January/001248.html (visited on 31/01/2019).

[15] Joseph W. Britton et al., *Engineered two-dimensional Ising interactions in a trapped-ion quantum simulator with hundreds of spins*, "Nature" Apr. 2012 484, pages 489–492, DOI: 10.1038/nature10981, URL: http://dx.doi.org/10.1038/nature10981 (visited on 15/08/2018).

[16] *CERN Open Hardware Licence*, version 1.2, CERN, 6th Sept. 2013, URL: https://www.ohwr.org/licenses/cern-ohl/license_versions/v1.2 (visited on 23/08/2018).

[17] Ryan Danell, *Creating an Integrated and Flexible Control Solution for Mass Spectrometer Systems Using PXI Hardware and LabVIEW*, Danell Consulting, URL: http://sine.ni.com/cs/app/doc/p/id/cs-13003# (visited on 20/08/2018).

[18] David P DiVincenzo, *The physical implementation of quantum computation*, "Fortschritte der Physik: Progress of Physics" 2000 48.nr. 9-11, pages 771–783.

[19] Collin Draughon, *Timing and Synchronization for the CompactRIO*, National Instruments, URL: ftp://ftp.ni.com/pub/branches/us/Dev%20Days/compactrio_timing_and_synchronization.pdf (visited on 20/08/2018).

[20] D-Wave Systems, *The D-Wave 2000Q^{TM} Quantum Computer Technology Overview*, 29th Oct. 2018, URL: https://www.dwavesys.com/sites/default/files/D-Wave%202000Q%20Tech%20Collateral_1029F.pdf (visited on 16/01/2019).

[21] eblot, *Repository: PyFtdi – FTDI device driver written in pure Python*, version 0.28.3, 26th Jan. 2018, URL: https://github.com/eblot/pyftdi (visited on 12/11/2018).

[22] Richard P. Feynman, *Simulating physics with computers*, "International Journal of Theoretical Physics" 6-7 June 1982 21, pages 467–488, DOI: 10.1007/BF02650179, URL: https://doi.org/10.1007/BF02650179 (visited on 15/08/2018).

[23] Future Technology Devices International Ltd, *FT4232H Quad High Speed USB to Multipurpose UART/MPSSE IC datasheet*, version 2.5, 7th Nov. 2017, URL: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT4232H.pdf (visited on 29/08/2018).

[24] Future Technology Devices International Ltd, *Interfacing FT2232H Hi-Speed Devices To I2C Bus*, 25th Feb. 2011, URL: http://www.ftdichip.com/Support/Documents/AppNotes/AN_113_FTDI_Hi_Speed_USB_To_I2C_Example.pdf (visited on 29/08/2018).

[25] Future Technology Devices International Ltd, *Programming Guide for High Speed FTCI2C DLL*, version 1.2, 2nd July 2009, URL: https://www.ftdichip.com/Support/Documents/AppNotes/AN_109_Programming_Guide_for_High_Speed_FTCI2C_DLL.pdf (visited on 06/01/2019).

[26] Ravindra Gali, *DDR2/DDR3 Low-Cost PCB Design Guidelines for Artix-7 and Spartan-7 FPGAs*, technical report, version 1.0, Sept. 2016, URL: https://www.xilinx.com/support/documentation/white_papers/wp484-a7-s7-ddr2-3-pcb.pdf (visited on 28/08/2018).

[27] Google, *A Preview of Bristlecone, Google's New Quantum Processor*, 5th Mar. 2018, URL: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (visited on 16/01/2019).

[28] Daniel Gottesman and Hoi-Kwong Lo, *From quantum cheating to quantum security*, "arXiv preprint quant-ph/0111100" 2001.

[29] Lov K Grover, *A fast quantum mechanical algorithm for database search*, *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, 1996, pages 212–219.

[30] Olivier Guinnard et al., *Easy-phi*, URL: http://easy-phi.unige.ch (visited on 20/08/2018).

[31] Olivier Guinnard et al., *Easy-phi repository*, URL: https://github.com/easy-phi/main (visited on 20/08/2018).

[32] Nick Holland, *Interfacing Between LVPECL, VML, CML, and LVDS Levels*, Xilinx Inc., Dec. 2002, URL: http://www.ti.com/lit/an/slla120/slla120.pdf (visited on 29/08/2018).

[33] IBM, *IBM Unveils World's First Integrated Quantum Computing System for Commercial Use*, 8th Jan. 2019, URL: https://newsroom.ibm.com/2019-01-08-IBM-Unveils-Worlds-First-Integrated-Quantum-Computing-System-for-Commercial-Use (visited on 16/01/2019).

[34] Marty Johnson et al., *Latch-Up*, technical report, Apr. 2015, URL: http://www.ti.com/lit/wp/scaa124/scaa124.pdf (visited on 29/08/2018).

[35] Grzegorz Kasprowicz, *Repository: SATA to SFP+ converter*, version v1.1, 23rd Aug. 2018, URL: https://github.com/sinara-hw/SATA_SFP (visited on 25/11/2018).

[36] Grzegorz Kasprowicz et al., *Sinara: open source modular hardware for quantum physics*, *1st North American Conference on Trapped Ions*, Boulder, Colorado Aug. 2017, URL: https://www.nist.gov/sites/default/files/documents/2018/01/29/nacti_booklet_180125.pdf (visited on 18/02/2019).

[37] Keysight Technologies, *Quantum Researchers Toolkit + Labber*, 4th Aug. 2017, URL: http://literature.cdn.keysight.com/litweb/pdf/5992-2500EN.pdf (visited on 29/12/2018).

[38] P. Kómár et al., *A quantum network of clocks*, "Nature Physics" June 2014 10, pages 582–587, DOI: 10.1038/nphys3000, URL: http://dx.doi.org/10.1038/nphys3000 (visited on 15/08/2018).

[39] David Kramer, *DARPA looks beyond GPS for positioning, navigating, and timing*, "Physics Today" 2014 67.nr. 10.

[40] Lab Control Software Scandinavia AB, *Labber Quantum*, URL: http://quantum.labber.org/products (visited on 29/12/2018).

[41] Benjamin P Lanyon et al., *Towards quantum chemistry on a quantum computer*, "Nature chemistry" 2010 2.nr. 2, page 106.

[42] D. R. Leibrandt and J. Heidecker, *An open source digital servo for atomic, molecular, and optical physics experiments*, "The Review of scientific instruments" Dec. 2015 86, DOI: 10.1063/1.4938282.

[43] Micron Technology, *DDR3 SDRAM MT41K256M16 datasheet*, 14th Nov. 2006, URL: https://www.micron.com/~/media/documents/products/data-sheet/dram/ddr3/1gb_ddr3_sdram.pdf (visited on 28/08/2018).

[44] Micron Technology, *DDR3L SDRAM MT41K256M16 datasheet*, 17th Dec. 2017, URL: https://www.micron.com/~/media/documents/products/data-sheet/dram/ddr3/4gb_ddr3l.pdf (visited on 28/08/2018).

[45] Micron Technology, *Ibis models for MT41K256M16TW-107 IT*, 27th Aug. 2015, URL: https://www.micron.com/~/media/documents/products/sim-model/dram/ddr3/v00h_1p35_ibis.zip (visited on 28/08/2018).

[46] Mini-Circuits, *TC2-1TX+ – Surface mount RF transformer datasheet*, 22nd Dec. 2015, URL: https://ww2.minicircuits.com/pdfs/TC2-1TX+.pdf (visited on 03/12/2018).

[47] Murata Manufacturing Co., Ltd., *BLM18SG121TN1# product datasheet*, 31st Jan. 2019, URL: https://www.murata.com/en-eu/api/pdfdownloadapi?cate=luNoiseSupprFilteChipFerriBead&partno=BLM18SG121TN1%23 (visited on 31/01/2019).

[48] Yunseong Nam et al., *Ground-state energy estimation of the water molecule on a trapped ion quantum computer*, "arXiv e-prints" Feb. 2019, arXiv:1902.10171, arXiv:1902.10171, arXiv: 1902.10171 [quant-ph].

[49] Vlad Negnevitsky, *Feedback-stabilised quantum states in a mixed-species ion system*, PhD thesis, ETH Zurich, 2018, URL: https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/295923/Thesis_VNegnevitsky.pdf (visited on 24/02/2019).

[50] NI Instruments, *Choosing a CompactRIO Synchronization Technology*, technical report, June 2018, URL: http://www.ni.com/white-paper/52962/en/ (visited on 20/08/2018).

[51] Michael A. Nielsen and Isaac L. Chuang, *Distance measures for quantum information*, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2010, pages 399–424, DOI: 10.1017/CBO9780511976667.013.

[52] Michael A. Nielsen and Isaac L. Chuang, *Introduction and overview*, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2010, pages 1–59, DOI: 10.1017/CBO9780511976667.005.

[53] Elise M. Novitski, *Apparatus and Methods for a New Measurement of the Electron and Positron Magnetic Moments*, PhD thesis, Harvard University, Dec. 2017, URL: http://gabrielse.physics.harvard.edu/gabrielse/papers/2017/elise_thesis.pdf (visited on 03/01/2019).

[54] John Preskill, *Quantum Computing in the NISQ era and beyond*, "Quantum" 2018 2, page 79.

[55] T Rosenband et al., *Alpha-dot or not: comparison of two single atom optical clocks*, *Frequency Standards And Metrology*, World Scientific, 2009, pages 20–33.

[56] Ch. Schneider, Diego Porras and Tobias Schaetz, *Experimental quantum simulations of many-body physics with trapped ions*, "Reports on Progress in Physics" Jan. 2012 75, page 024401, DOI: 10.1088/0034-4885/75/2/024401, URL: https://doi.org/10.1088/0034-4885/75/2/024401 (visited on 15/08/2018).

[57] Peter W Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, Ieee, 1994, pages 124–134.

[58] Silicon Laboratories, *Si5324 Any-frequency precision clock multiplier/jitter attenuator datasheet*, version 1.1, Jan. 2014, URL: https://www.silabs.com/documents/public/data-sheets/Si5324.pdf (visited on 28/08/2018).

[59] Silicon Laboratories, *Si5324 LVPECL IBIS model*, version 0.1, 12th Nov. 2007, URL: https://www.silabs.com/documents/public/software/Si531x_2x_Any-rate_Clocks3v3IBIS.zip (visited on 11/11/2018).

[60] *Sinara Open Hardware Project*, 20th Aug. 2018, URL: https://github.com/sinara-hw (visited on 23/08/2018).

[61] Switchcraft, *Jack & Plugs Product Bulletin*, Jan. 2012, URL: http://www.switchcraft.com/Documents/529.pdf (visited on 30/08/2018).

[62] Texas Instruments Inc., *TPS51200 Sink and Source DDR Termination Regulator*, Nov. 2016, URL: http://www.ti.com/lit/ds/symlink/tps51200.pdf (visited on 28/08/2018).

[63] Umesh Vazirani and Thomas Vidick, *Fully device-independent quantum key distribution*, "Physical review letters" 2014 113.nr. 14, page 140501.

[64] Xilinx Inc., *7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide*, version 1.10.1, 23rd July 2018, URL: https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf (visited on 30/01/2019).

[65] Xilinx Inc., *7 Series FPGAs GTP Transceivers*, version 1.9, 19th Dec. 2016, URL: https://www.xilinx.com/support/documentation/user_guides/ug482_7Series_GTP_Transceivers.pdf (visited on 29/08/2018).

[66] Xilinx Inc., *7 Series FPGAs Packaging and Pinout Product Specification*, version 1.17, 17th Aug. 2018, URL: https://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf (visited on 29/08/2018).

[67] Xilinx Inc., *7 Series FPGAs PCB Design Guide*, version 1.13, 18th Aug. 2018, URL: https://www.xilinx.com/support/documentation/user_guides/ug483_7Series_PCB.pdf (visited on 30/08/2018).

[68] Xilinx Inc., *7 Series FPGAs SelectIO Resources User Guide*, version 1.8, 8th May 2018, URL: https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf (visited on 28/08/2018).

[69] Xilinx Inc., *7 Series FPGAs Configuration User Guide*, version 1.13.1, 20th Aug. 2018, URL: https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf (visited on 29/08/2018).

[70] Xilinx Inc., *7-Series Transceiver IBIS-AMI models*, version 1.0, 25th Feb. 2013, URL: http://www.xilinx.com/member/ibis_ami/ (visited on 13/09/2018).

[71] Xilinx Inc., *All Programmable 7 Series Product Selection Guide*, version 1.7, 2018, URL: https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf (visited on 26/08/2018).

[72] Xilinx Inc., *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics*, version 1.25, 18th June 2018, URL: https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf (visited on 29/08/2018).

[73] Xilinx Inc., *Platform Cable USB II datasheet*, version 1.5.1, 6th Aug. 2018, URL: https://www.xilinx.com/support/documentation/data_sheets/ds593.pdf (visited on 29/08/2018).

[74] Xilinx Inc., *Recommended Design Rules and Strategies for BGA Devices*, version 1.0, 1st Mar. 2016, URL: https://www.xilinx.com/support/documentation/user_guides/ug1099-bga-device-design-rules.pdf (visited on 30/08/2018).

[75] Xilinx Inc., *Vivado Design Suite User Guide Programming and Debugging*, version 2018.2, 6th June 2018, URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug908-vivado-programming-debugging.pdf (visited on 29/08/2018).

[76] Xilinx Inc., *Xilinx Power Estimator for 7 Series and Zynq®-7000*, version 2018.2, 18th June 2018, URL: https://www.xilinx.com/products/technology/power/xpe.html (visited on 30/08/2018).

[77] Xilinx Inc., *Zynq-7000 AP SoC and 7 Series Devices Memory Interface Solutions User Guide*, version 4.2, 4th Oct. 2017, URL: https://www.xilinx.com/support/documentation/ip_documentation/mig_7series/v4_2/ug586_7Series_MIS.pdf (visited on 29/08/2018).

[78] Zurich Instruments, *Quantum Computing Control System (QCCS)*, URL: https://www.zhinst.com/products/quantum (visited on 29/12/2018).

List of Abbreviations

Acousto-Optic Modulator (AOM) Modulator which uses the acousto-optic effect to shift the frequency of light using sound waves (made by piezo-electric transducer). 12, 73

Advanced Mezzanine Card (AMC) A PCB that is inserted into a MicroTCA crate, various cards are offered by many companies depending on requirements in the user's system. 17, 82

Advanced Real-Time Infrastructure for Quantum physics (ARTIQ) Software and gateware real-time framework used for interfacing with devices, computation and fast feedback control. 16, 17, 20, 21, 23, 25, 27, 37, 54, 56–58, 61, 62, 70, 72, 73, 108–110, 118

Analog-to-Digital Converter (ADC) A system (module or single IC) that converts an analog signal (e.g. voltage level) into a digital number representing given input. 17, 19, 61, 73, 75

Application Programming Interface (API) A set of methods used for communication between programs or its components. 15

Ball Grid Array (BGA) A type of surface-mount packaging used for integrated circuits, with a grid array of solder pins. 23, 28, 35

Bill of Materials (BoM) A list of the components and parts with quantities of each needed to manufacture an end product. 23

CERN Open Hardware License (CERN OHL) License used for open-source hardware. It describes usage and modification terms, in particular it forces publication of modifications to the original design. 13, 17

Current Mode Logic (CML) Differential, digital, point-to-point logic standard. 25

Digital Input/Output module (DIO) An EEM which supplies 8 digital inputs/outputs via various connectors. 19, 74, 75

Digital-to-Analog Converter (DAC) A system (module or single IC) that converts a digital input into an analog signal (e.g. voltage level). 17, 19, 73, 81

Direct Digital Synthesizer (DDS) A device that generates signals using Direct Digital Synthesis, a method in which a series of digital signals representing the desired waveform is generated and sent to a DAC. 19, 73, 75

Distributed Real-Time Input/Output (DRTIO) Time and data transfer system that allows ARTIQ real-time I/O channels to be distributed among several satellite devices synchronized and controlled by a central core device. 17, 21, 23

Double Data Rate 3 (DDR3) Memory operating at double data rate – transferring data on both rising and falling edges of the clock signal. 30, 33, 39, 40, 45, 48, 49, 57

Electrically Erasable Programmable Read-Only Memory (EEPROM) Type of non-volatile memory. 22, 70

Electromagnetic interference (EMI) A disturbance generated by an external source that affects an electrical circuit by electromagnetic induction, electrostatic coupling, or conduction. 31, 34, 65

Electrostatic Discharge (ESD) A sudden flow of current between two electrically charged objects. 26, 54

Eurocard Extension Module (EEM) Extension modules in 3U form factor designed for performing a single function like ADC or DAC modules. 19–22, 25, 27–29, 31, 33–35, 39, 41, 43, 45, 48, 53, 54, 56, 58, 62, 63, 72, 74, 81, 109, 113, 118

Field-Programmable Gate Array (FPGA) IC that has various programmable logic blocks and connections that can be programmed by the user. 16, 19, 21, 23–30, 33–37, 39, 40, 45, 46, 48, 53, 54, 58, 59, 61, 62, 65, 66, 73, 82, 83, 108, 110

FPGA Mezzanine Card (FMC) Standard that defines connectors and I/O mezzanine modules with connection to a device with configurable inputs/outputs. 19

General Purpose Input Output (GPIO) Digital signal pin on an IC or electronic circuit. Its behaviour and purpose is defined by user. 42, 45, 66, 70

Global Positioning System (GPS) Satellite-based system providing time and location information to GPS receivers. 11

Graphical User Interface (GUI) Interface in which the user interacts with a device through graphical icons, menus etc. 16, 56

Input/output Buffer Information Specification (IBIS) A Specification of a method to provide information about the input/output buffers of their product. 45, 46, 48

Insulation-Displacement Connector (IDC) Type of electrical connector. 21, 23, 29, 34

Integrated Bit Error Rate Tester (IBERT) IP core from Xilinx used for testing of gigabit transceivers. 56, 57, 62

Integrated Circuit (IC) A set of electronic circuits on one piece of semiconductor. 23–27, 29, 30, 40, 45, 48, 70, 81–83

Integrated Development Environment (IDE) A software application providing integrated environment for development, modifying and testing software or hardware. 14, 27, 54, 61, 108

Integrated Logic Analyzer (ILA) Xilinx' IP core that can be used to capture and show signals inside the FPGA. 27

Inter-Integrated Circuit (I^2C) Synchronous serial bus used for attaching lower-speed ICs to processors or microcontrollers over short distances (typically in are of one board). 9, 10, 21, 22, 25, 27–30, 32, 41, 42, 45, 46, 53, 54, 56, 58, 62, 65, 66, 70, 71, 82, 121

Joint Test Action Group (JTAG) A standard protocol and interface used for testing of integrated circuits and PCBs, named after Joint Test Action Group which defined it. 21, 27, 29, 42, 59, 61, 70, 82

Light-Emitting Diode (LED) A semiconductor light source. 28, 30, 46, 65, 66, 109, 110

Low-Dropout regulator (LDO) Linear voltage regulator that can regulate output voltage even when input supply is close to the output voltage. 29, 30, 63, 68

Low-Voltage Differential Signalling (LVDS) A technical standard specifying electrical characteristics of a differential, low-power communication protocol. 21, 25, 27–29, 33–35, 41, 45, 54, 56, 61, 64, 68

Low-Voltage Positive Emitter-Coupled Logic (LVPECL) High-speed low-voltage integrated circuit bipolar transistor logic. 25, 33, 34, 45, 68

Memory Interface Generator (MIG) Xilinx' IP core which generates memory controller and pin assignments for the FPGA. 24, 39

Micro-Miniature Coaxial (MMCX) Type of coaxial RF connectors with lock-snap mechanism. 25, 33, 37, 38, 46, 47, 58–60, 63, 64, 66–68, 72, 129–133

MicroTCA Carrier Hub (MCH) A card that manages AMC cards inserted into a MicroTCA crate and its power supplies and fan speed. 17

Multi-Protocol Synchronous Serial Engine (MPSSE) Serial engine used by FTDI ICs for generating I^2C, JTAG, SPI and other signals for synchronous interfaces. 26, 53, 62, 65, 70, 121

National Institute of Standards and Technology (NIST) 15–17, 72, 73

On-Die Termination (ODT) Technology where termination resistors are located on IC die, reducing space required for termination. 24, 39, 48

Power Suppression Rejection Ratio (PSRR) Ratio describing capability of an electronic circuit to suppress any power supply variations to its output signal. 63, 64, 68

Precision Time Protocol (PTP) A protocol used to synchronize clocks through a computer network. 14

Printed Circuit Board (PCB) Board with electrical connections and solder pads for electronic components. 16, 28, 33, 38, 41, 45, 64, 81, 83, 110

Pseudo-Random Bit Sequence (PRBS) 46

Quick-locking SMA (QMA) A type of coaxial RF connector designed to replace SMA connectors with quick-locking ones. 14

Radio frequency (RF) Signal frequencies spanning from as low as 20 kHz up to around 300 GHz. In this thesis signals are considered to have RF when their frequency exceeds 100 MHz. 12, 73, 74, 82, 83

Random Access Memory (RAM) Fast, volatile memory used for keeping data and code currently being used that allows write and read access to any location in the memory. 21, 28, 30

Serial AT Attachment (SATA) A computer bus interface and connectors, usually used in computer storage devices. 31, 36, 56, 57

Serial Peripheral Interface (SPI) Synchronous serial communication interface. 27, 28, 61, 82

Signal Source Analyzer (SSA) An instrument that measures magnitude and phase of the input signal at a single frequency. In this thesis it was used to measure phase noise, by sweeping frequency from the carrier. 59, 66

Small Form-factor Pluggable (SFP) Hot-pluggable transceiver. 21, 23, 28–31, 34, 36, 42, 56, 57, 66, 109

SubMiniature version A (SMA) Coaxial RF connectors. 25, 31, 33, 34, 38, 63, 64, 68, 73, 83

Surface Mount Device (SMD) A method of producing electronic circuits in which the components are placed directly onto the surface of PCB. 35

Synchronous Dynamic Random-Access Memory (SDRAM) Random access memory, in which its operation is coordinated by an externally supplied clock signal. 23, 24, 27, 39, 40, 45, 48, 57

Transient Voltage Suppressor (TVS) Type of devices designed to react to a sudden or momentary overvoltage conditions. 54, 62

Universal Asynchronous Receiver-Transmitter (UART) A device used for communications over a serial port. 21, 27–29, 41, 42, 46, 57, 62, 65

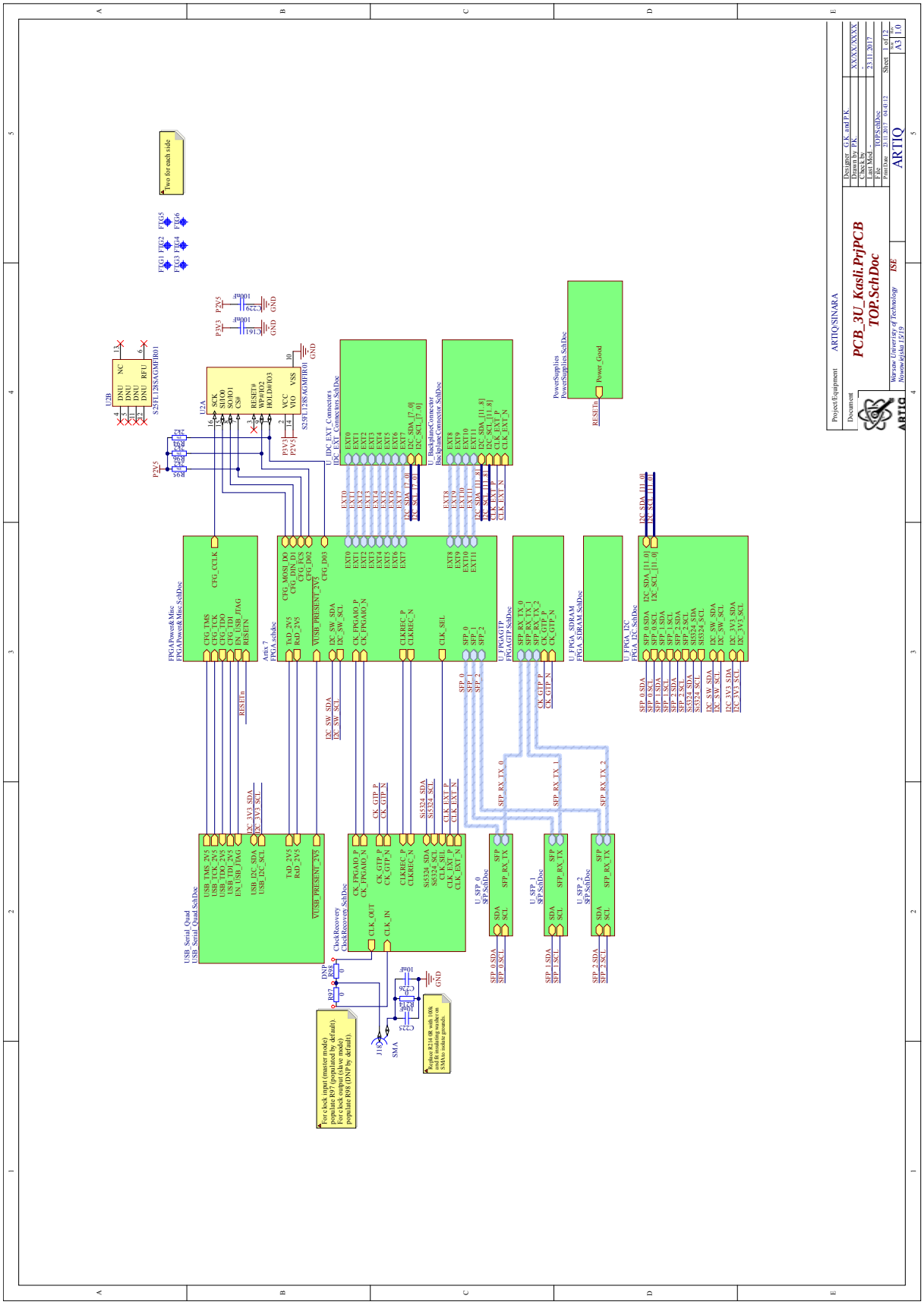
Universal Serial Bus (USB) A communication port and interface widely used for connecting peripherals to a computer. 14, 21, 26–28, 31, 33, 34, 42, 45, 53, 55, 59, 61, 109

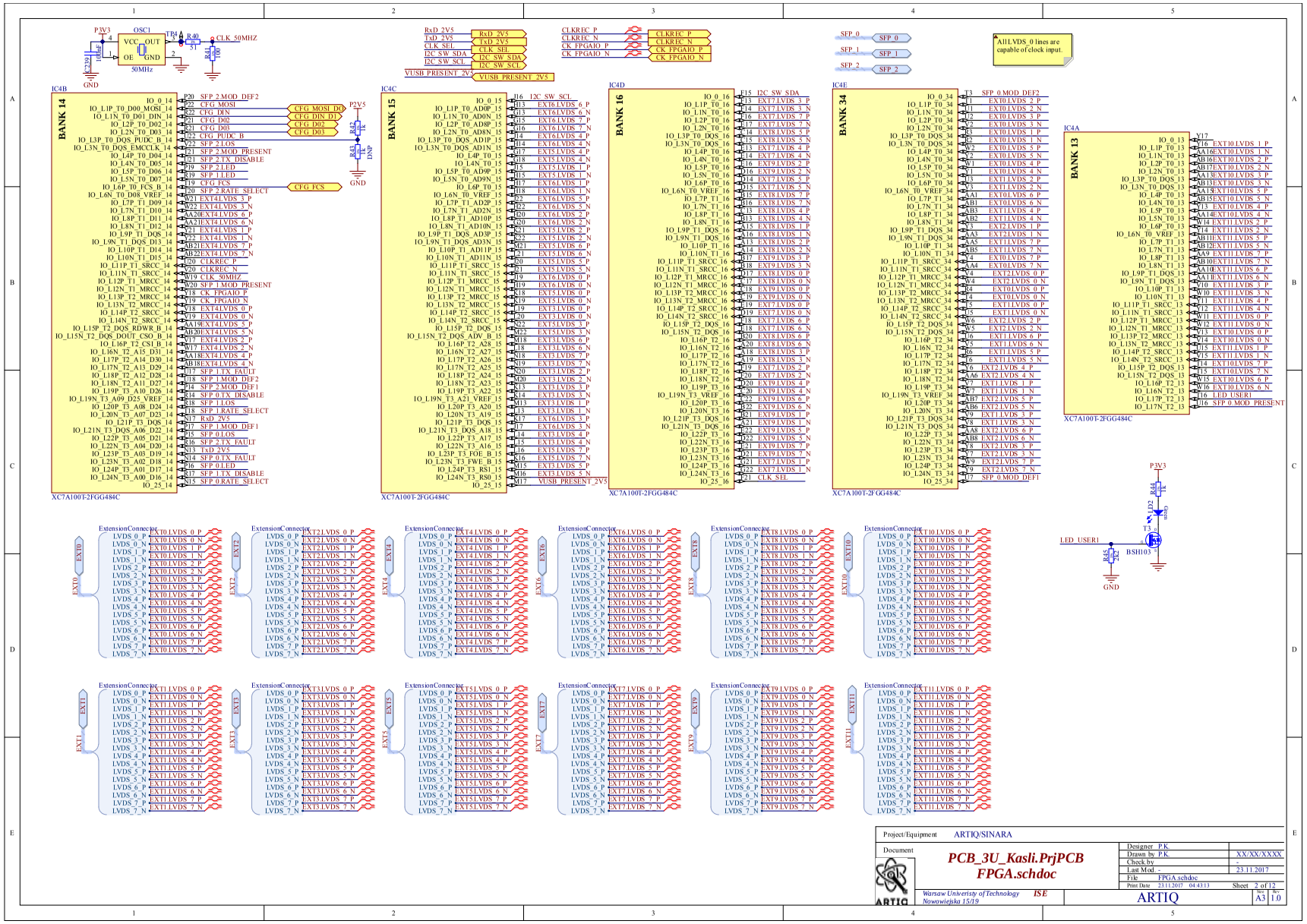
Very-High-Density Cable Interconnect (VHDCI) 68-pin compact connector. 19–21, 29, 31

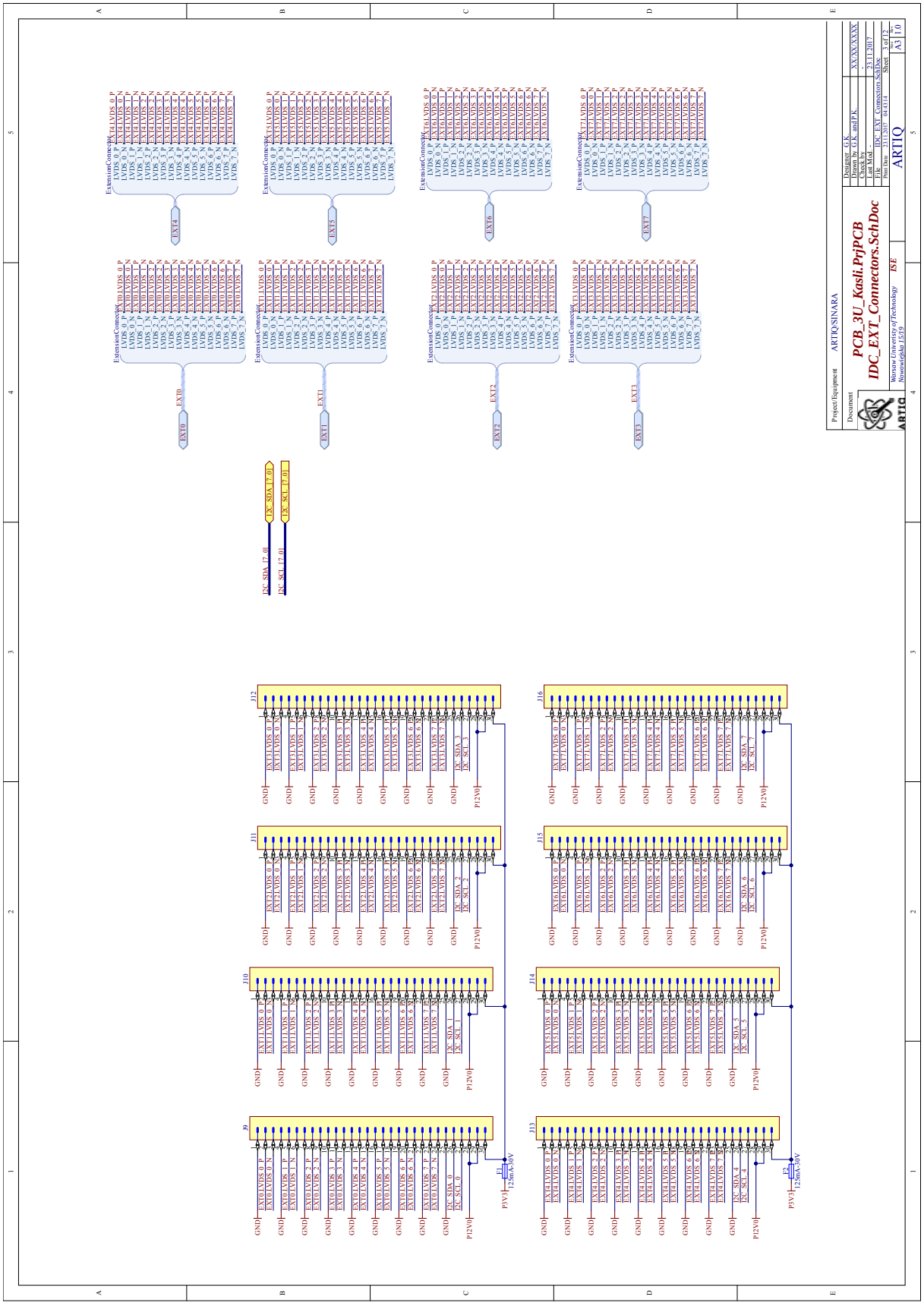
VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (VHDL) Hardware description language used in electronic design. 27, 56

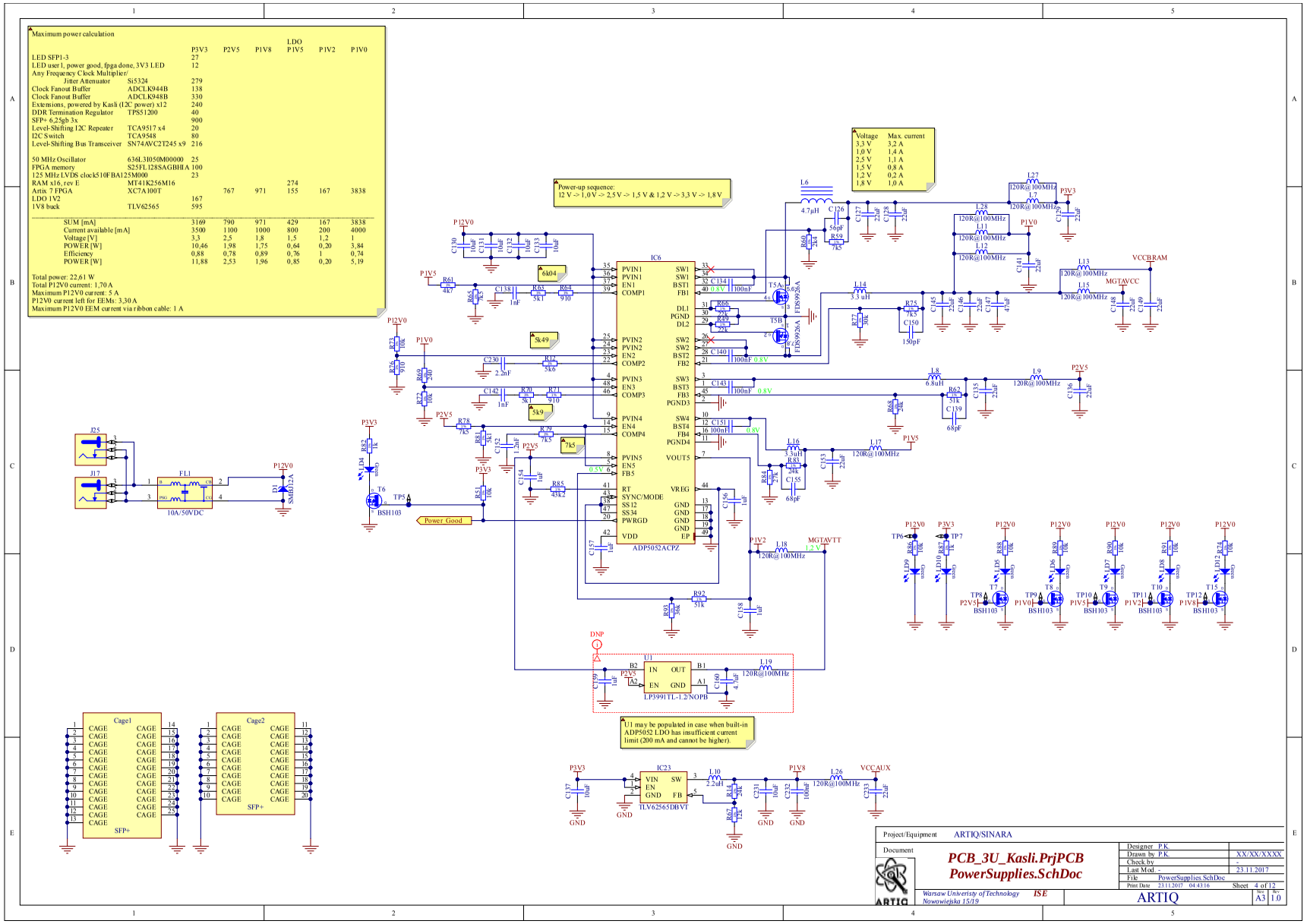
Virtual I/O (VIO) Xilinx' IP core that can be used to manually set or read signals inside the FPGA. 54, 56

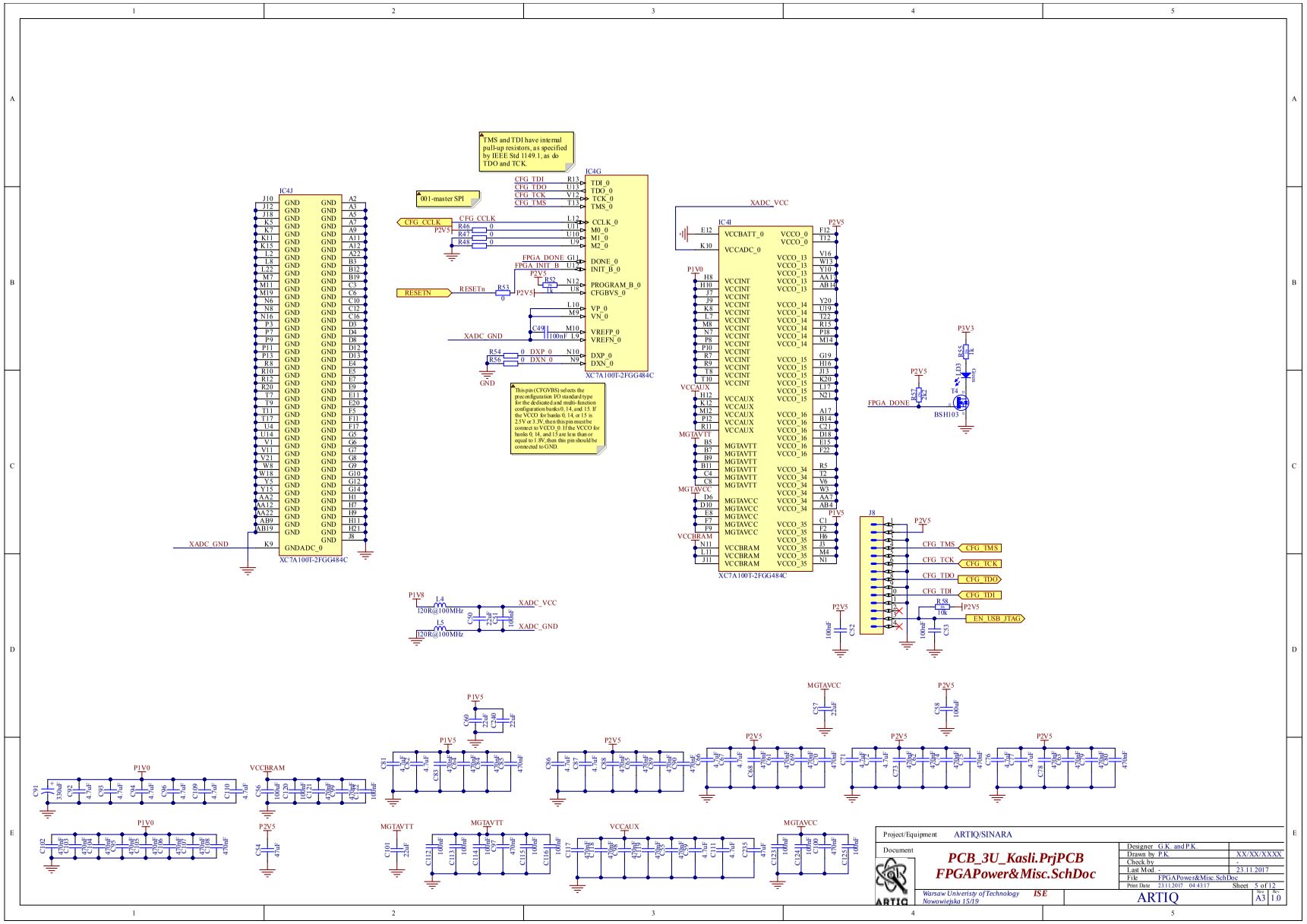
A. Electrical schematics – Kasli v1.0

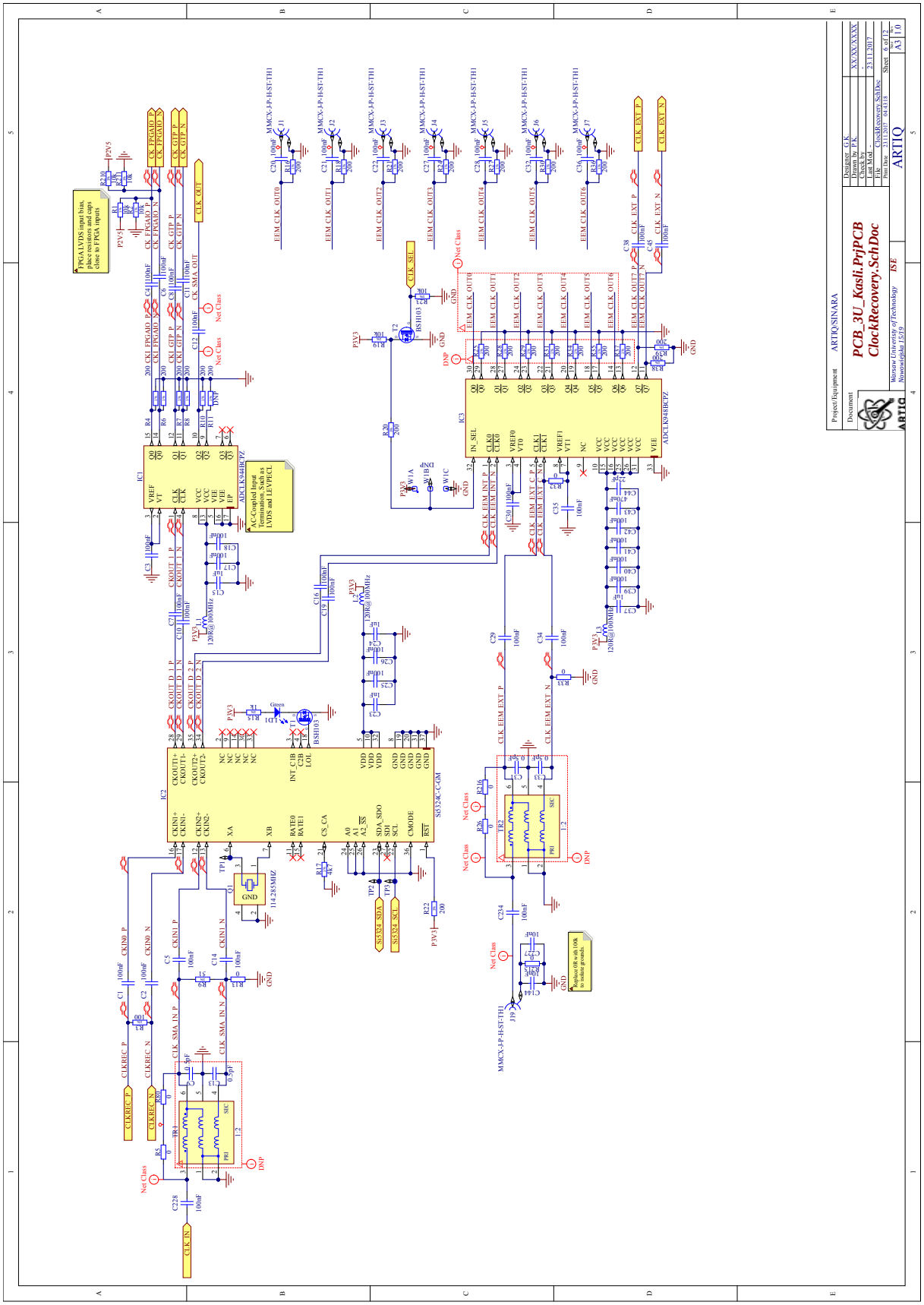


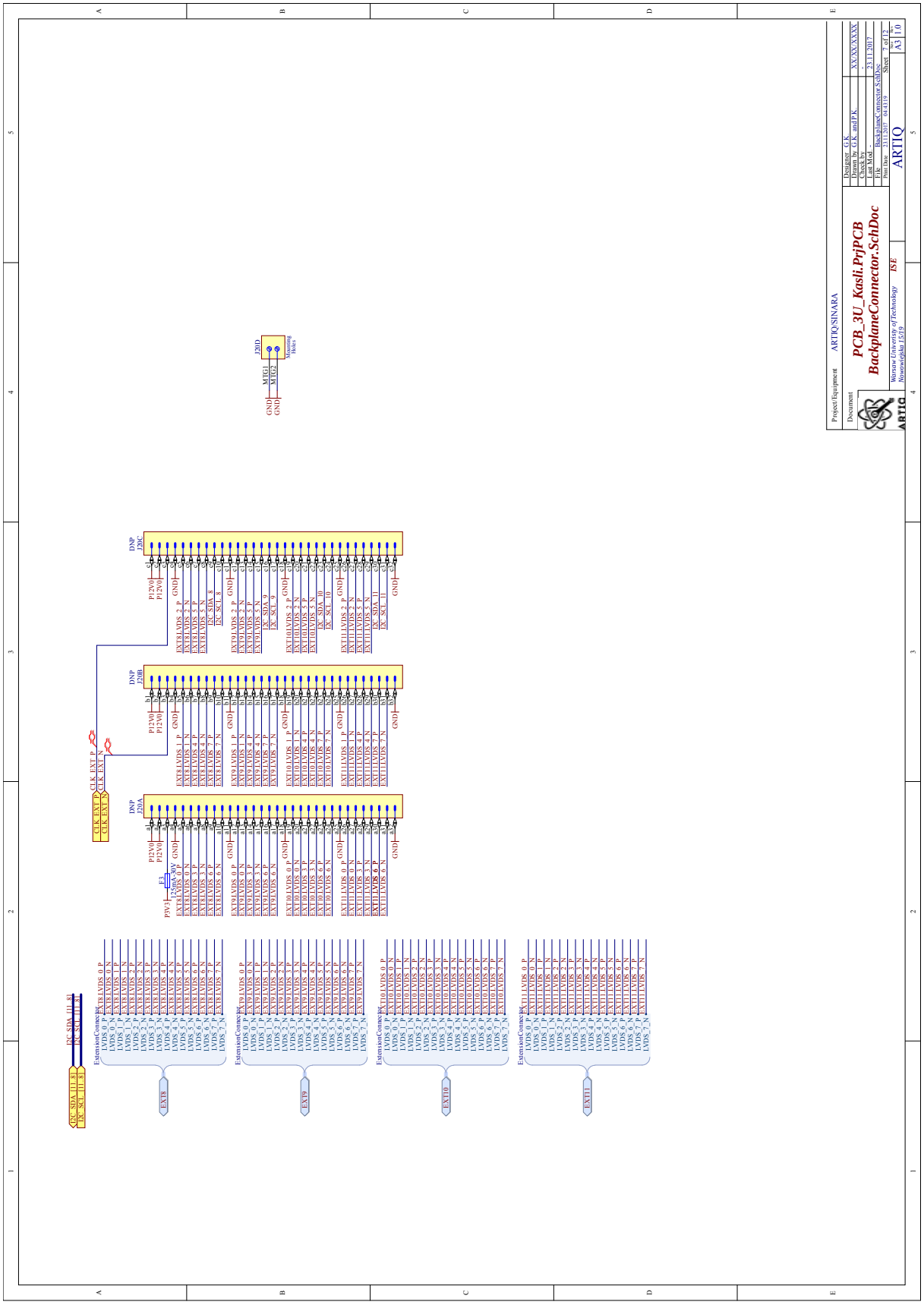

OSC1
50MHz
CLK_50MHZ
GND
P3V3
P2V5
IC4B
BANK 14
IC4C
BANK 15
IC4D
BANK 16
IC4E
BANK 34
IC4A
BANK 13
XC7A100T-2FGG484C
RxD_2V5
TxD_2V5
CLK_SEL
I2C_SW_SDA
I2C_SW_SCL
VUSB_PRESENT_2V5
CLKREC_P
CLKREC_N
CK_FPGAIO_P
CK_FPGAIO_N
SFP_0
SFP_1
SFP_2
All LVDS_0 lines are capable of clock input.
CFG_MOSI_D0
CFG_DIN_D1
CFG_D02
CFG_D03
CFG_FCS
LED_USER1
BSH103
ExtensionConnector
Project/Equipment
ARTIQ/SINARA
Document
PCB_3U_Kasli.PrjPCB
FPGA.schdoc
Designer P.K.
Drawn by P.K.
XX/XX/XXXX
Check by
Last Mod. 23.11.2017
File FPGA.schdoc
Print Date 23.11.2017 04:43:13
Sheet 2 of 12
Warsaw University of Technology
Nowowiejska 15/19
ISE
ARTIQ
Size A3
Rev 1.0

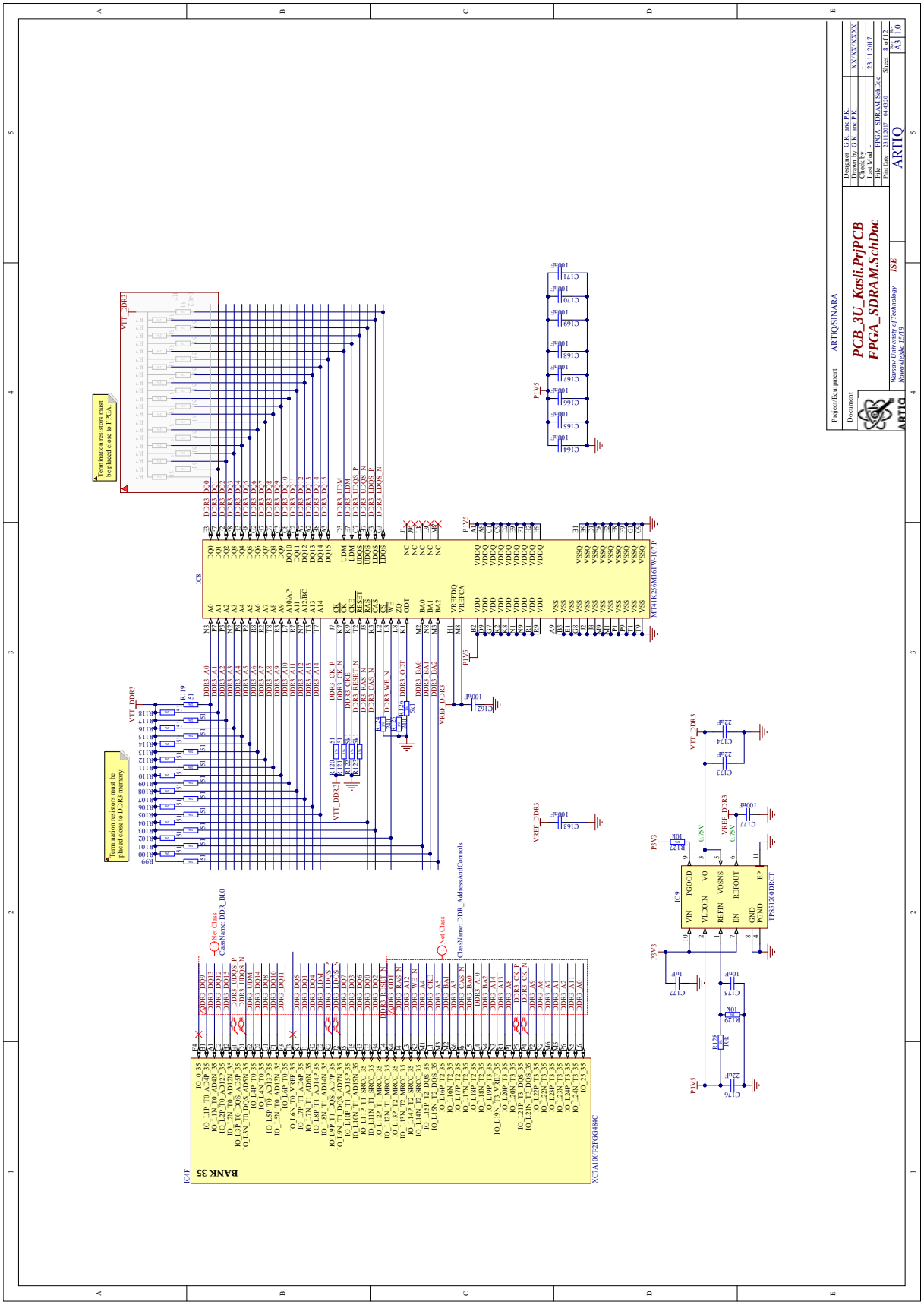

PCB_3U_Kasli.PrjPCB
IDC_EXT_Connectors.SchDoc
ARTIQ/SINARA
ARTIQ
Warsaw University of Technology
Instytut Systemów Elektronicznych
Designer G.K.
Drawn by G.K. and PK
Check by
Engr Mgr
Date 21/11/2017
Sheet 3 of 12
File IDC_EXT_Connectors.SchDoc
Print Date 21/11/2017
A3 1.0
EXT0
EXT1
EXT2
EXT3
EXT4
EXT5
EXT6
EXT7
J9
J10
J11
J12
J13
J14
J15
J16
I2C_SDA_[7..0]
I2C_SCL_[7..0]
P12V0
P3V3
GND

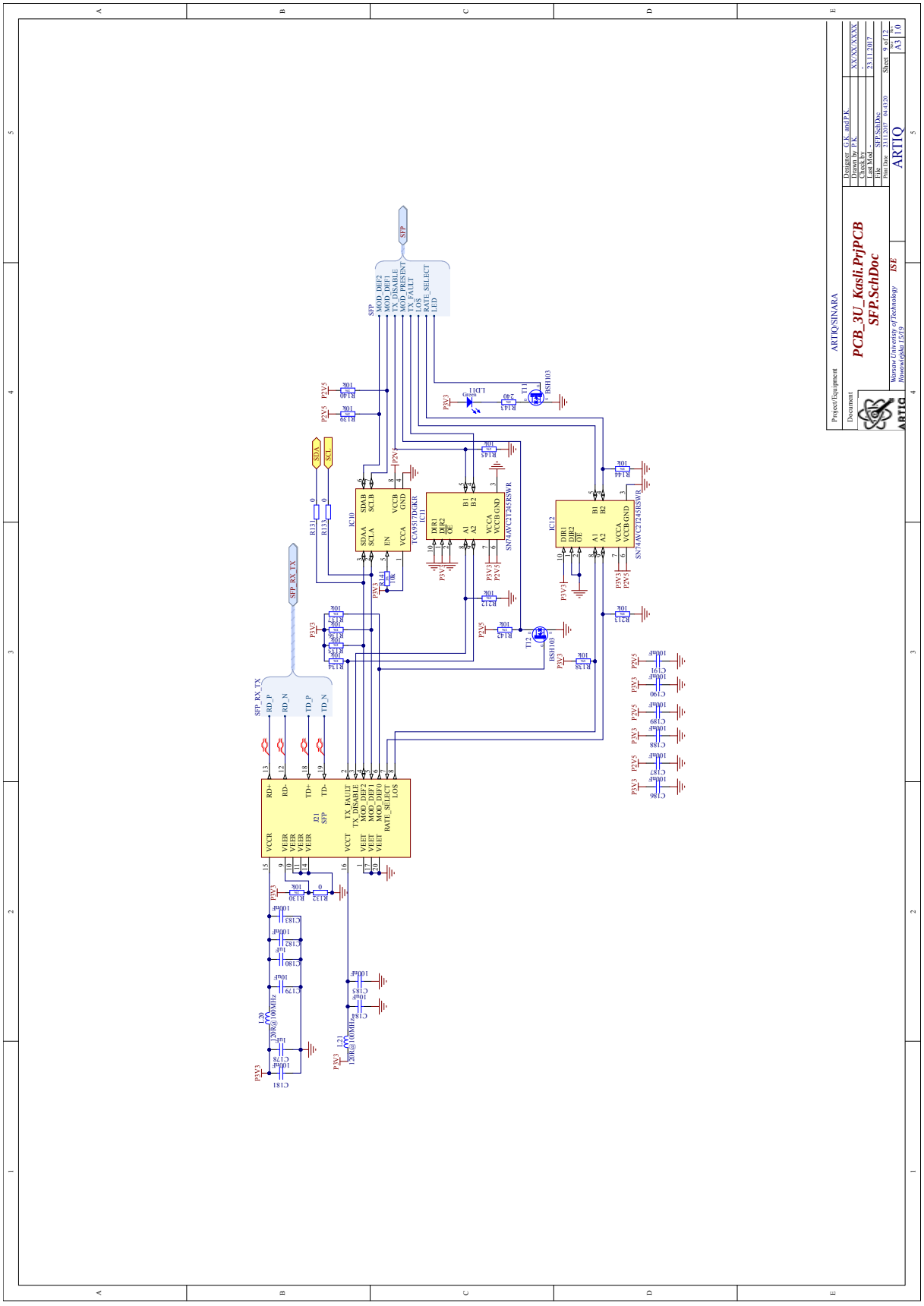

Maximum power calculation
P3V3 P2V5 P1V8 LDO P1V5 P1V2 P1V0
LED SFP1-3 27
LED user1, power good, fpga done, 3V3 LED 12
Any Frequency Clock Multiplier/ Jitter Attenuator Si5324 279
Clock Fanout Buffer ADCLK944B 138
Clock Fanout Buffer ADCLK948B 330
Extensions, powered by Kasli (I2C power) x12 240
DDR Termination Regulator TPS51200 40
SFP+ 6,25gb 3x 900
Level-Shifting I2C Repeater TCA9517 x4 20
I2C Switch TCA9548 80
Level-Shifting Bus Transceiver SN74AVC2T245 x9 216
50 MHz Oscillator 636L3I050M00000 25
FPGA memory S25FL128SAGBHIA 100
125 MHz LVDS clock510FBA125M000 23
RAM x16, rev E MT41K256M16 274
Artix 7 FPGA XC7A100T 767 971 155 167 3838
LDO 1V2 167
1V8 buck TLV62565 595
SUM [mA] 3169 790 971 429 167 3838
Current available [mA] 3500 1100 1000 800 200 4000
Voltage [V] 3,3 2,5 1,8 1,5 1,2 1
POWER [W] 10,46 1,98 1,75 0,64 0,20 3,84
Efficiency 0,88 0,78 0,89 0,76 1 0,74
POWER [W] 11,88 2,53 1,96 0,85 0,20 5,19
Total power: 22,61 W
Total P12V0 current: 1,70 A
Maximum P12V0 current: 5 A
P12V0 current left for EEMs: 3,30 A
Maximum P12V0 EEM current via ribbon cable: 1 A
Voltage Max. current
3,3 V 3,2 A
1,0 V 1,4 A
2,5 V 1,1 A
1,5 V 0,8 A
1,2 V 0,2 A
1,8 V 1,0 A
Power-up sequence:
12 V -> 1,0 V -> 2,5 V -> 1,5 V & 1,2 V -> 3,3 V -> 1,8 V
IC6
ADP5052ACPZ
PVIN1 PVIN2 PVIN3 PVIN4 PVIN5
SW1 SW2 SW3 SW4 VOUT5
EN1 EN2 EN3 EN4 EN5
BST1 BST2 BST3 BST4
COMP1 COMP2 COMP3 COMP4
FB1 FB2 FB3 FB4 FB5
DL1 DL2 PGND PGND3 PGND4
RT SYNC/MODE SS12 SS34 PWRGD VDD VREG GND EP
Power Good
IC23
TLV62565DBVT
VIN SW EN GND FB
U1
LP3991TL-1.2/NOPB
IN OUT EN GND
DNP
U1 may be populated in case when built-in ADP5052 LDO has insufficient current limit (200 mA and cannot be higher).
J25
J17
FL1
10A/50VDC
D1
SMBJ12A
T6
BSH103
Cage1
Cage2
SFP+
Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB
PowerSupplies.SchDoc
Designer P.K.
Drawn by P.K. XX/XX/XXXX
Check by -
Last Mod. 23.11.2017
File PowerSupplies.SchDoc
Print Date 23.11.2017 04:43:16
Sheet 4 of 12
Warsaw University of Technology ISE
Nowowiejska 15/19
ARTIQ
Size A3 Rev 1.0

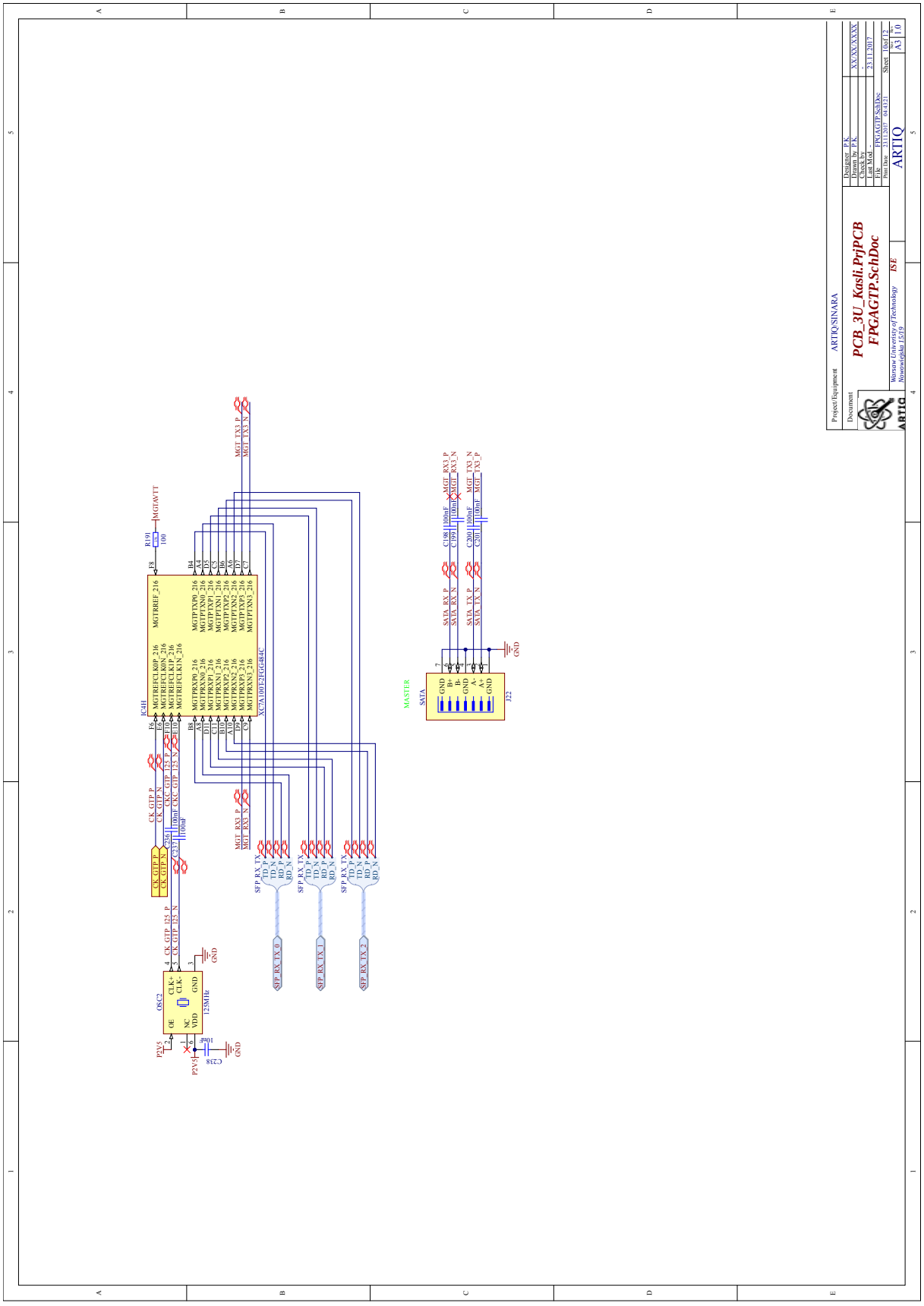

TMS and TDI have internal pull-up resistors, as specified by IEEE Std 1149.1, as do TDO and TCK.
001-master SPI
This pin (CFGVBS) selects the pre configuration I/O standard type for the dedicated and multi-function configuration banks 0, 14, and 15. If the VCCO for banks 0, 14, or 15 is 2.5V or 1.3V, then this pin must be connect to VCCO_0. If the VCCO for banks 0, 14, and 15 are less than or equal to 1.8V, then this pin should be connected to GND.
IC4J
IC4G
IC4I
XC7A100T-2FGG484C
XADC_VCC
XADC_GND
P1V0
P1V5
P2V5
P1V8
P2V3
VCCBRAM
MGTAVTT
MGTAVCC
VCCAUX
CFG_CCLK
RESETn
FPGA_DONE
FPGA_INIT_B
CFG_TDI
CFG_TDO
CFG_TCK
CFG_TMS
EN_USB_JTAG
BSH103
Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB
FPGAPower&Misc.SchDoc
Designer G.K. and P.K.
Drawn by P.K. XX/XX/XXXX
Check by -
Last Mod. - 23.11.2017
File FPGAPower&Misc.SchDoc
Print Date 23.11.2017 04:43:17 Sheet 5 of 12
Warsaw University of Technology Nowowiejska 15/19
ISE
ARTIQ
Size A3 Rev 1.0

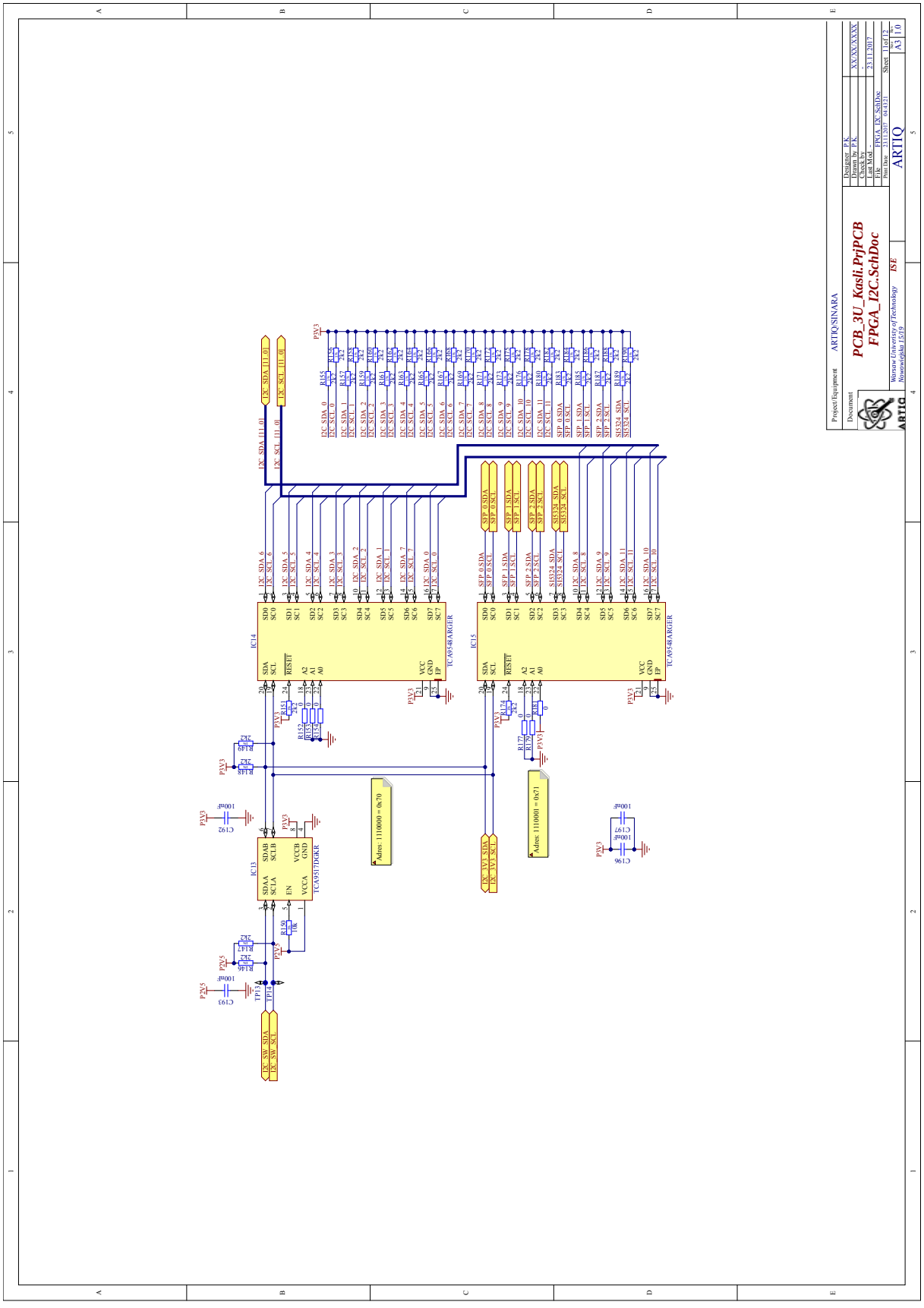

PCB_3U_Kasli.PrjPCB
ClockRecovery.SchDoc
ARTIQ/SINARA
Designer G.K.
Drawn by P.K.
Check by
Eng. Mod
Date 25.11.2017
Sheet 6 of 12
File ClockRecovery.SchDoc
Print Date 21/11/2017 04:43:18
Size A3 1.0
Warsaw University of Technology
Nowowiejska 15/19
ISE
ARTIQ
Project Equipment
Document
PFGA LVDS input bias, place resistors and caps close to PFGA inputs
AC-coupled input terminated with a LVDS and LEVPECL
Replace 0R with 100k to isolate ground
CLK_EXT_IN
CLK_EXT_OUT
CLK_SMA_OUT
CLK_SEL
CLK_IN
CLKREC_IN
CLK_SMA_IN
CK_FPGA0_P
CK_FPGA0_N
CK_GTP_P
CK_GTP_N
ADCLK948BCPZ
SI5324_SDA
SI5324_SCL
MMCX-JP-H-ST-TH1
EEM_CLK_OUT0
EEM_CLK_OUT1
EEM_CLK_OUT2
EEM_CLK_OUT3
EEM_CLK_OUT4
EEM_CLK_OUT5
EEM_CLK_OUT6
EEM_CLK_OUT7_P
EEM_CLK_OUT7_N
CLK_EEM_EXT_P
CLK_EEM_EXT_N
CLK_EEM_INT_P
CLK_EEM_INT_N
DNP

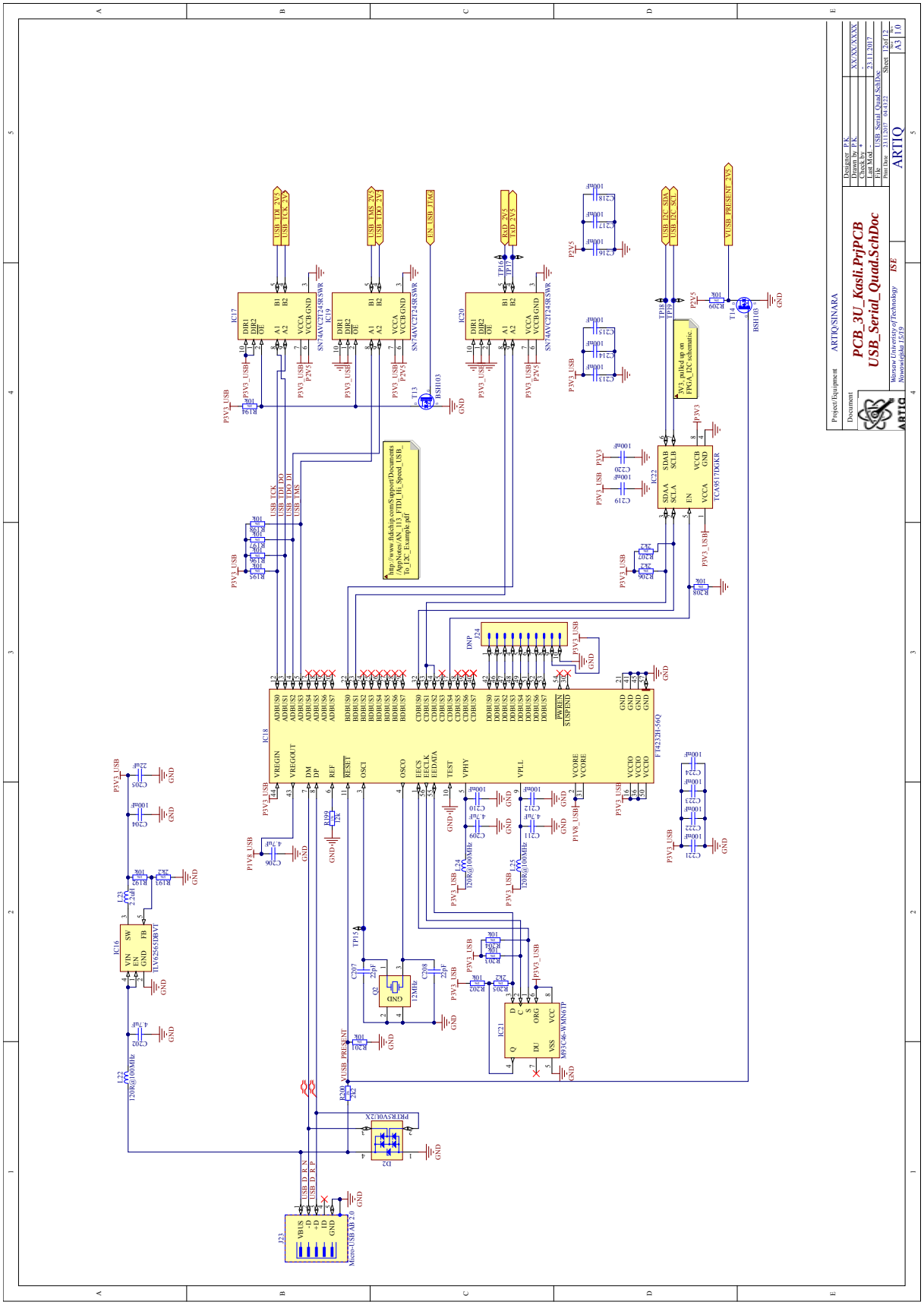

Project/Equipment ARTIQ/SINARA
Document
PCB_3U_Kasli.PrjPCB
BackplaneConnector.SchDoc
Designer G.K.
Drawn by G.K. and P.K. XX/XX/XXXX
Check by
File Mod 23/11/2017
File BackplaneConnector.SchDoc
Print Date 23/11/2017 04:43:19 Sheet 7 of 12
Warsaw University of Technology ISE
Wydział Fizyki
ARTIQ
A3 1.0
J20D
MTG1
MTG2
Mounting Hole
GND
J20C
J20B
J20A
P12V0
P3V3
CLK_EXT_P
CLK_EXT_N
I2C_SDA_9
I2C_SCL_9
I2C_SDA_10
I2C_SCL_10
I2C_SDA_11
I2C_SCL_11
I2C_SDA_8
I2C_SCL_8
I2C_SDA_[11..8]
I2C_SCL_[11..8]
EXT8
EXT9
EXT10
EXT11


PCB_3U_Kasli.PrjPCB
FPGA_SDRAM.SchDoc
ARTIQ/SINARA
Designer G.K. and P.K.
Drawn by G.K. and P.K.
Check by
File Mod
Print Date 23/11/2017
Sheet 8 of 12
ARTIQ
termination resistors must be placed close to FPGA
termination resistors must be placed close to DDR3 memory
VTT_DDR3
VREF_DDR3
IC8
IC4F
IC9
BANK 35
XC7A100F2FGG484C
TPS51200DRCT
MT41K256M16TW-107:P
DDR_AddressAndControls
DDR_BL0

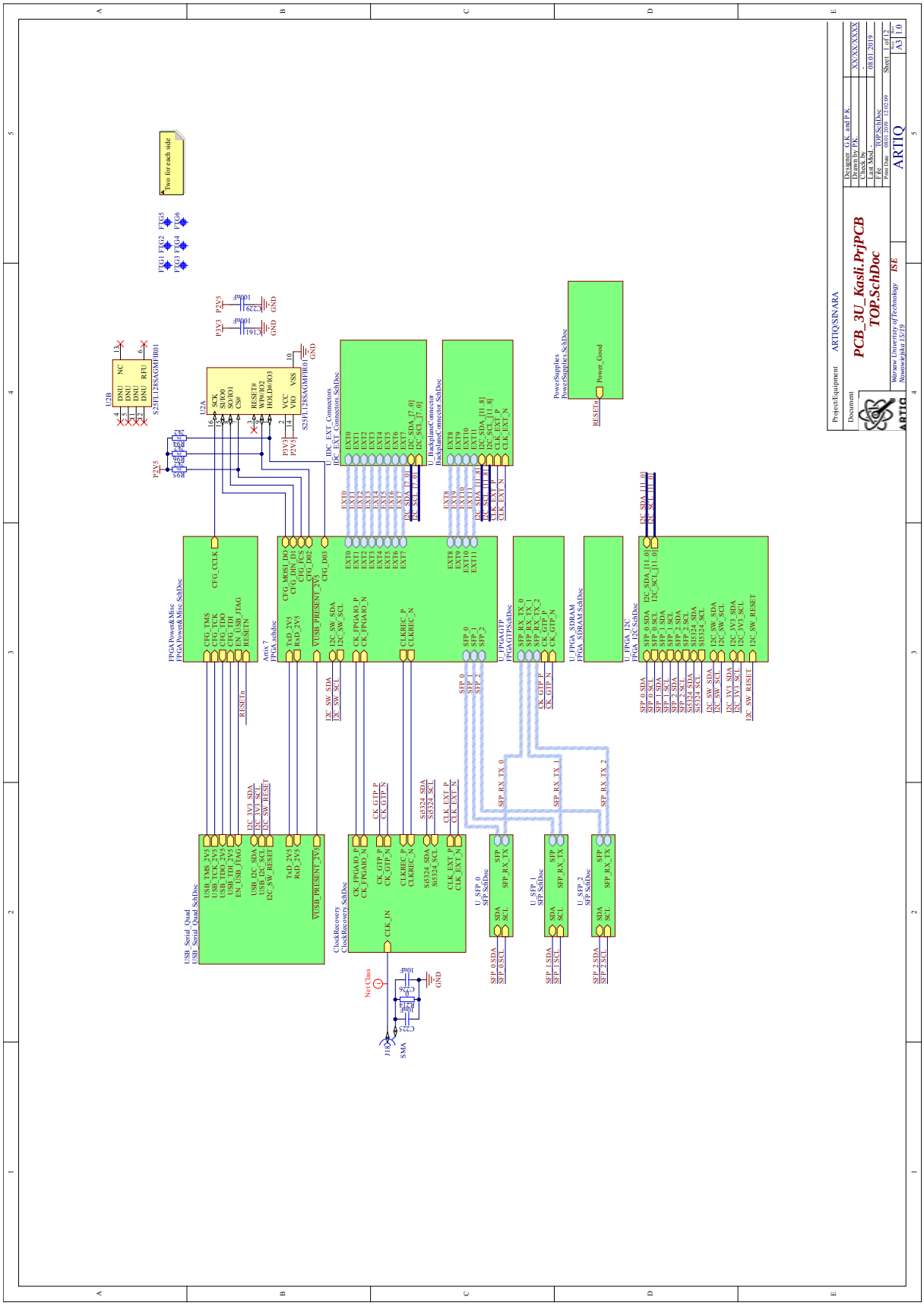

PCB_3U_Kasli.PrjPCB
SFP.SchDoc
ARTIQ/SINARA
Warsaw University of Technology
Wydział Fizyki
ISE
ARTIQ
SFP
SFP_RX_TX
SDA
SCL
IC10
TCA9517DGKR
IC11
SN74AVC2T245RSWR
IC12
SN74AVC2T245RSWR
J21
SFP
T11
BSH103
T12
BSH103
LD11
L20
L21
120R@100MHz

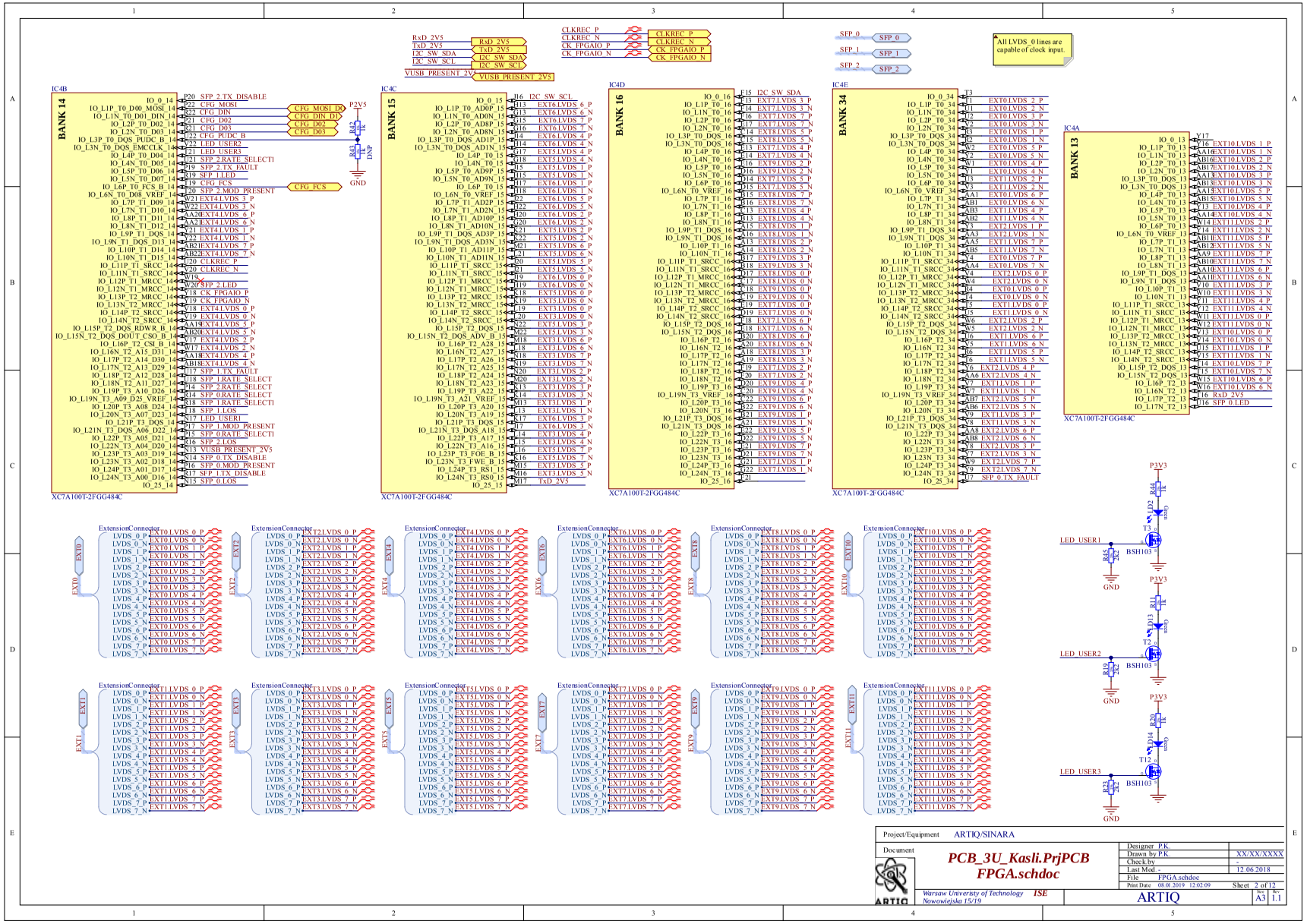

Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB FPGAGTP.SchDoc
Designer P.K.
Drawn by P.K.
Check by:
File Mod 25.11.2017
File PKGAGTP.SchDoc
Print Date 21/11/2017 04:43:21 Sheet 10of 12
Warsaw University of Technology ISE Nowowiejska 15/19
ARTIQ
A3 1.0
IC4H
XC7A100T-2CSG484C
MGTREFCLK0P_216
MGTREFCLK0N_216
MGTREFCLK1P_216
MGTREFCLK1N_216
MGTREFP_216
MGTPRXP0_216
MGTPRXN0_216
MGTPRXP1_216
MGTPRXN1_216
MGTPRXP2_216
MGTPRXN2_216
MGTPRXP3_216
MGTPRXN3_216
MGTPTXP0_216
MGTPTXN0_216
MGTPTXP1_216
MGTPTXN1_216
MGTPTXP2_216
MGTPTXN2_216
MGTPTXP3_216
MGTPTXN3_216
CK_GTP_P
CK_GTP_N
CK_GTP_125_P
CK_GTP_125_N
MGTAVTT
R193
100
C237
100nF
OSC2
CLK+
CLK-
OE
NC
VDD
GND
125MHz
P2V5
C238
10nF
SFP_RX_TX
TD_P
TD_N
RD_P
RD_N
SFP_RX_TX_0
SFP_RX_TX_1
SFP_RX_TX_2
MGT_RX3_P
MGT_RX3_N
MGT_TX3_P
MGT_TX3_N
MASTER
SATA
SATA_RX_P
SATA_RX_N
SATA_TX_P
SATA_TX_N
C198
C199
C200
C201
100nF
722

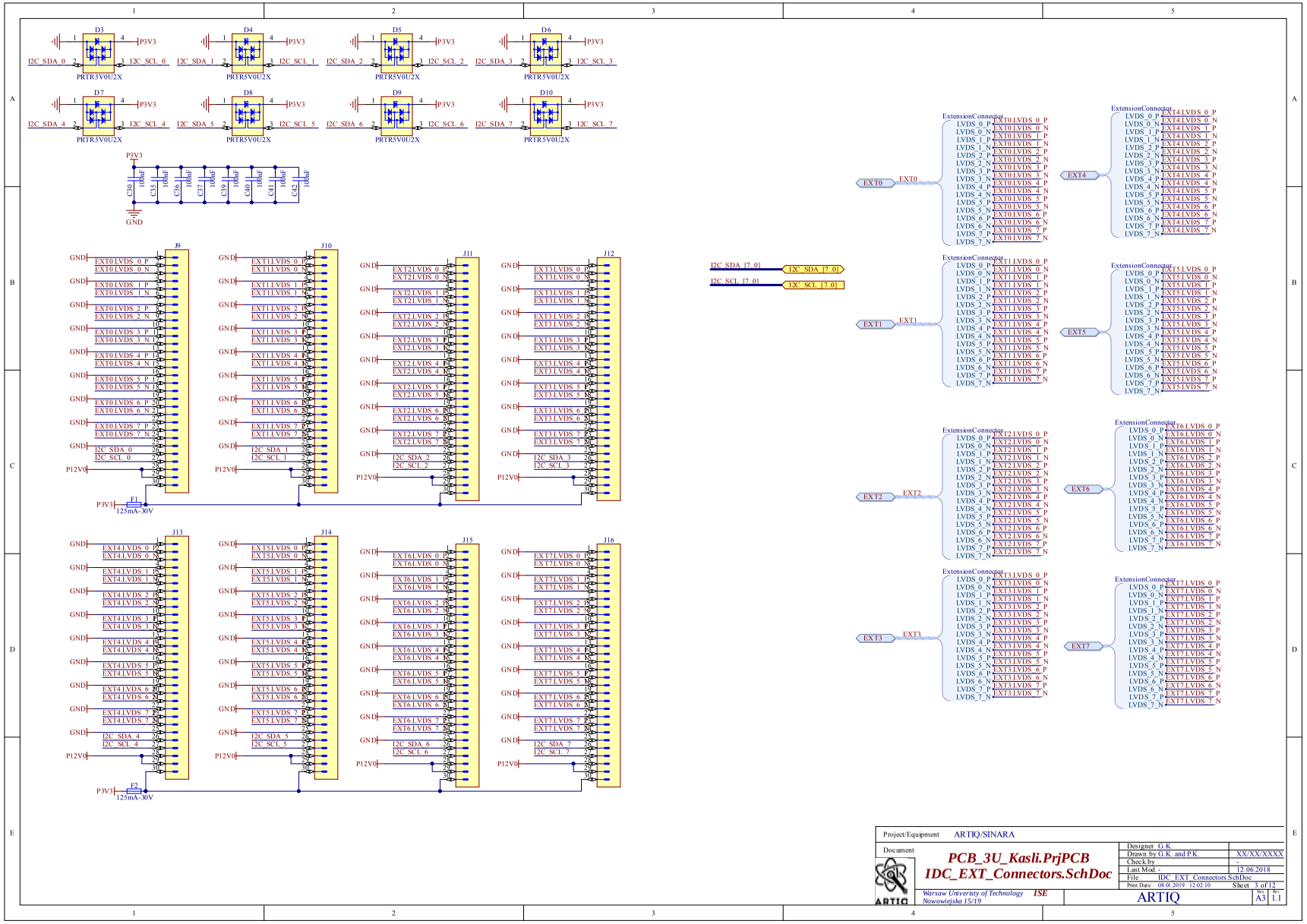

PCB_3U_Kasli.PrjPCB
FPGA_I2C.SchDoc
ARTIQ/SINARA
ARTIQ
Warsaw University of Technology
Instytut Systemów Elektronicznych
IC14
IC15
IC13
TCA9548ARGER
TCA9517DGKR
Adres: 1110000 = 0x70
Adres: 1110001 = 0x71

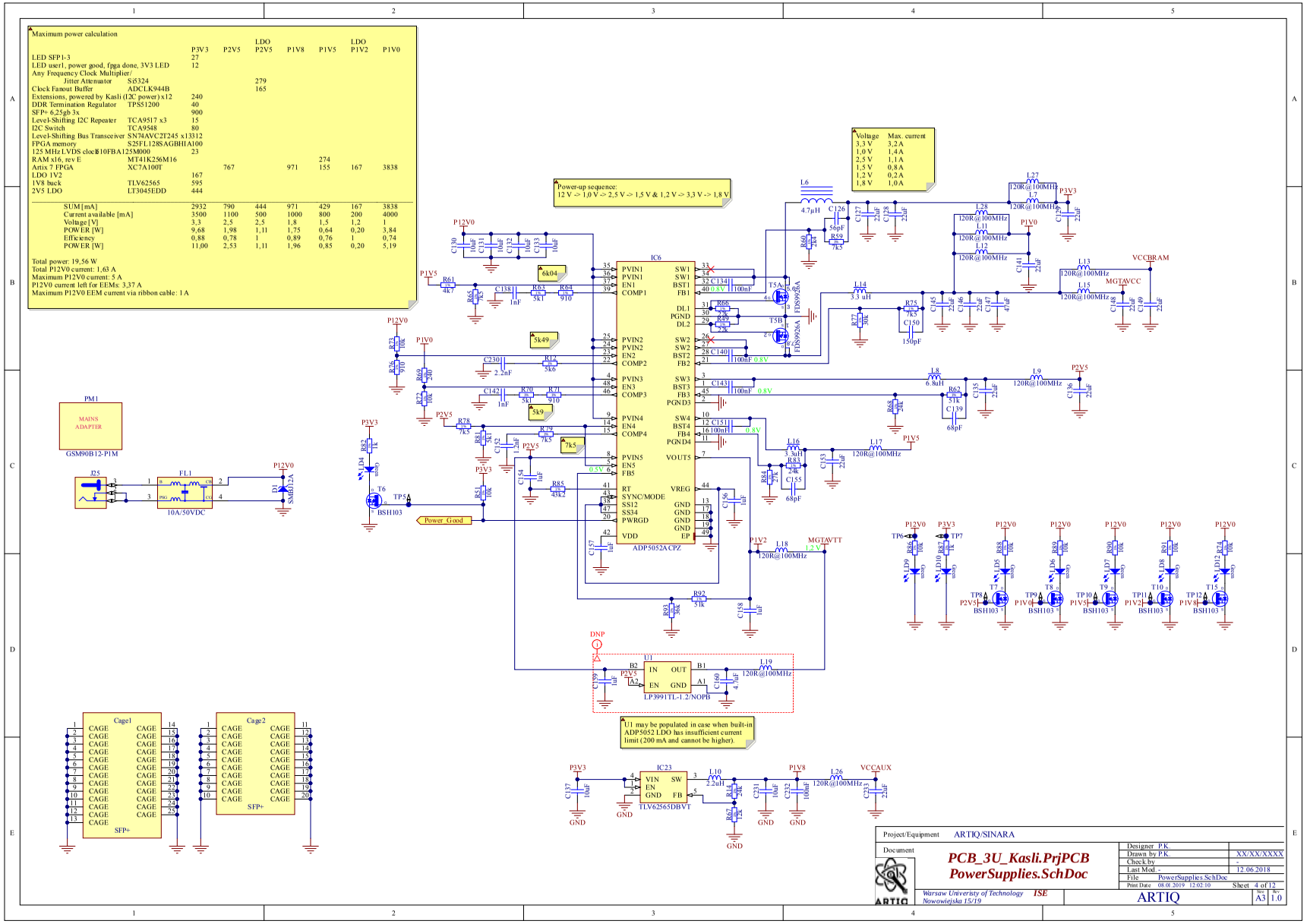

PCB_3U_Kasli.PrjPCB
USB_Serial_Quad.SchDoc
ARTIQ/SINARA
Designer P.K.
Drawn by P.K.
Check by *
File Mod 25/11/2017
Print Date 25/11/2017 04:43:23
Sheet 12of 12
A3 1.0
ARTIQ
Warsaw University of Technology
Instytut Fizyki
ISE

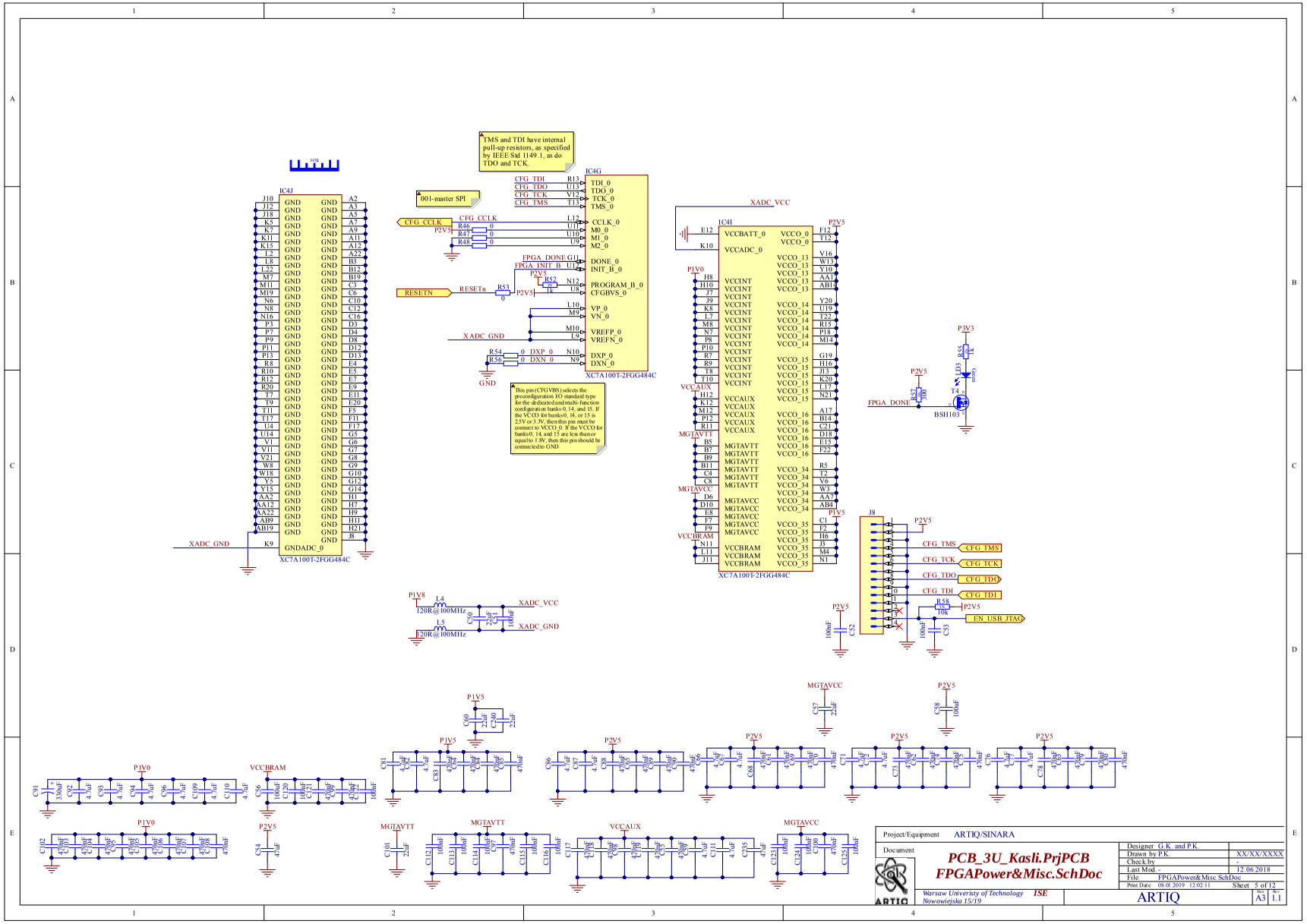
B. Electrical schematics – Kasli v1.1

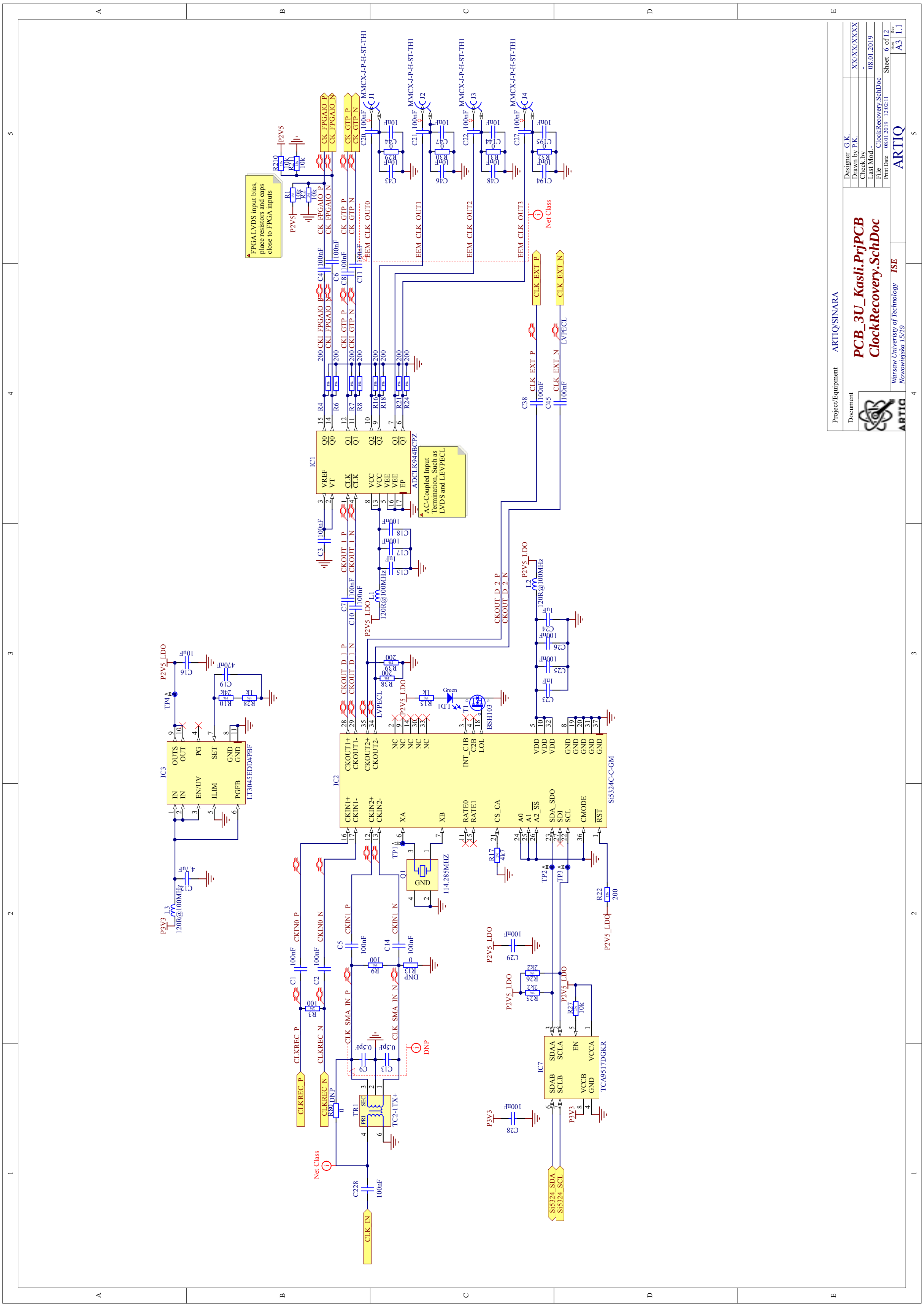



IC4B
BANK 14
XC7A100T-2FGG484C
IC4C
BANK 15
XC7A100T-2FGG484C
IC4D
BANK 16
XC7A100T-2FGG484C
IC4E
BANK 34
XC7A100T-2FGG484C
IC4A
BANK 13
XC7A100T-2FGG484C
All LVDS_0 lines are capable of clock input.
RxD_2V5
TxD_2V5
I2C_SW_SDA
I2C_SW_SCL
VUSB_PRESENT_2V5
CLKREC_P
CLKREC_N
CK_FPGAIO_P
CK_FPGAIO_N
SFP_0
SFP_1
SFP_2
CFG_MOSI_D0
CFG_DIN_D1
CFG_D02
CFG_D03
CFG_FCS
P2V5
R42
1k
R43
DNP
GND
P3V3
LED_USER1
LED_USER2
LED_USER3
BSH103
R44
R45
R11
R12
R20
R21
R22
R23
1k
100R
D12
D13
D14
T1
T2
T3
Project/Equipment
ARTIQ/SINARA
Document
PCB_3U_Kasli.PrjPCB
FPGA.schdoc
Designer P.K.
Drawn by P.K.
XX/XX/XXXX
Check.by
Last Mod. -
12.06.2018
File
FPGA.schdoc
Print Date 08.01.2019 12.02.09
Sheet 2 of 12
Warsaw Univeristy of Technology
Nowowiejska 15/19
ISE
ARTIQ
Size A3
Rev 1.1

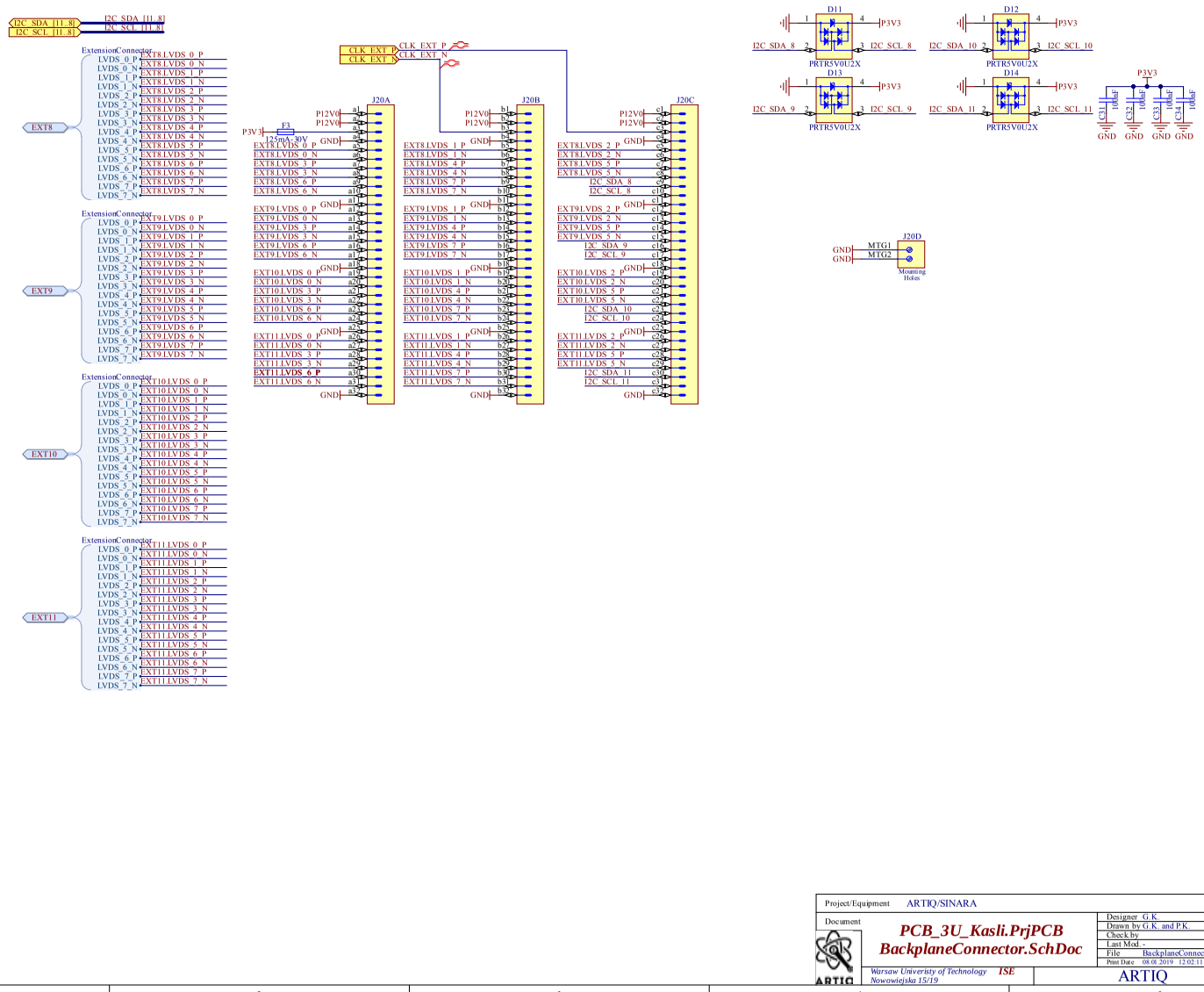

D3
D4
D5
D6
D7
D8
D9
D10
PRTR5V0U2X
P3V3
I2C_SDA_0
I2C_SCL_0
I2C_SDA_1
I2C_SCL_1
I2C_SDA_2
I2C_SCL_2
I2C_SDA_3
I2C_SCL_3
I2C_SDA_4
I2C_SCL_4
I2C_SDA_5
I2C_SCL_5
I2C_SDA_6
I2C_SCL_6
I2C_SDA_7
I2C_SCL_7
C30 100nF
C35 100nF
C36 100nF
C37 100nF
C39 100nF
C40 100nF
C41 100nF
C42 100nF
GND
J9
J10
J11
J12
J13
J14
J15
J16
P12V0
F1
125mA-30V
F2
125mA-30V
I2C_SDA_[7..0]
I2C_SCL_[7..0]
EXT0
EXT1
EXT2
EXT3
EXT4
EXT5
EXT6
EXT7
ExtensionConnector
Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB
IDC_EXT_Connectors.SchDoc
Designer G.K.
Drawn by G.K. and P.K. XX/XX/XXXX
Check.by
Last Mod. 12.06.2018
File IDC_EXT_Connectors.SchDoc
Print Date 08.08.2019 12:02:10
Sheet 3 of 12
Warsaw Univeristy of Technology ISE
Nowowiejska 15/19
ARTIQ
Size A3
Rev 1.1

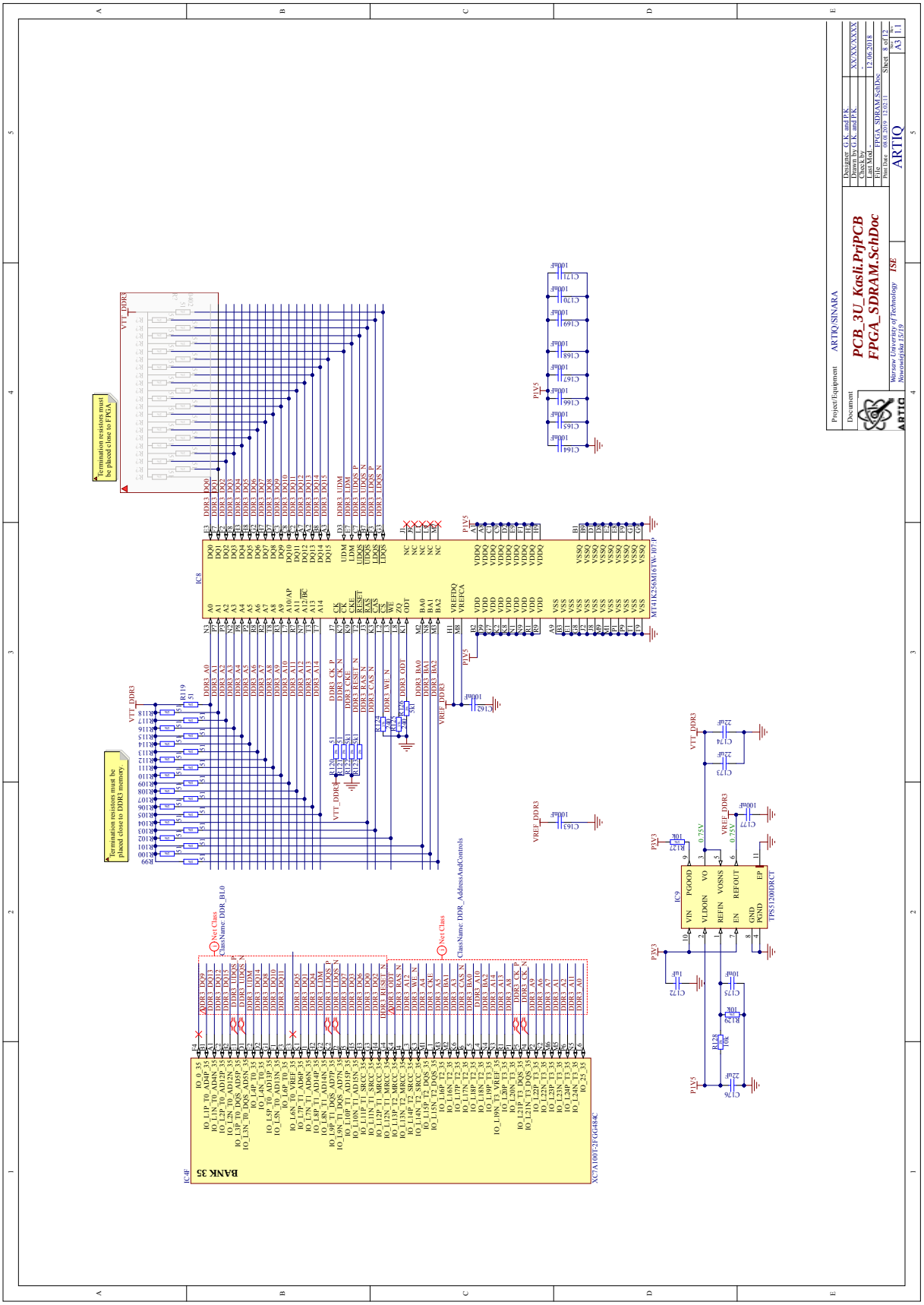

Maximum power calculation
P3V3 P2V5 LDO P2V5 P1V8 P1V5 LDO P1V2 P1V0
LED SFP1-3 27
LED user1, power good, fpga done, 3V3 LED 12
Any Frequency Clock Multiplier:
Jitter Attenuator Si5324 279
Clock Fanout Buffer ADCLK944B 165
Extensions, powered by Kasli (I2C power) x12 240
DDR Termination Regulator TPS51200 40
SFP+ 6,25gb 3x 900
Level-Shifting I2C Repeater TCA9517 x3 15
I2C Switch TCA9548 80
Level-Shifting Bus Transceiver SN74AVC2T245 x13 312
FPGA memory S25FL128SAGBHIA100
125 MHz LVDS clock 810FBA125M000 23
RAM x16, rev E MT41K256M16 274
Artix 7 FPGA XC7A100T 767 971 155 167 3838
LDO 1V2 167
1V8 buck TLV62565 595
2V5 LDO LT3045EDD 444
SUM [mA] 2932 790 444 971 429 167 3838
Current available [mA] 3500 1100 500 1000 800 200 4000
Voltage [V] 3,3 2,5 2,5 1,8 1,5 1,2 1
POWER [W] 9,68 1,98 1,11 1,75 0,64 0,20 3,84
Efficiency 0,88 0,78 1 0,89 0,76 1 0,74
POWER [W] 11,00 2,53 1,11 1,96 0,85 0,20 5,19
Total power: 19,56 W
Total P12V0 current: 1,63 A
Maximum P12V0 current: 5 A
P12V0 current left for EEMs: 3,37 A
Maximum P12V0 EEM current via ribbon cable: 1 A
Power-up sequence:
12 V -> 1,0 V -> 2,5 V -> 1,5 V & 1,2 V -> 3,3 V -> 1,8 V
Voltage Max. current
3,3 V 3,2 A
1,0 V 1,4 A
2,5 V 1,1 A
1,5 V 0,8 A
1,2 V 0,2 A
1,8 V 1,0 A
IC6
ADP5052ACPZ
IC23
TLV62565DBVT
U1
LP3991TL-1.2/NOPB
U1 may be populated in case when built-in ADP5052 LDO has insufficient current limit (200 mA and cannot be higher).
PM1
MAINS ADAPTER
GSM90B12-P1M
Cage1
Cage2
SFP+
Power_Good
Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB
PowerSupplies.SchDoc
Designer P.K.
Drawn by P.K. XX/XX/XXXX
Check by
Last Mod. 12.06.2018
File PowerSupplies.SchDoc
Print Date 08.08.2019 12:02:10
Sheet 4 of 12
Warsaw University of Technology ISE
Nowowiejska 15/19
ARTIQ
Size A3
Rev 1.0

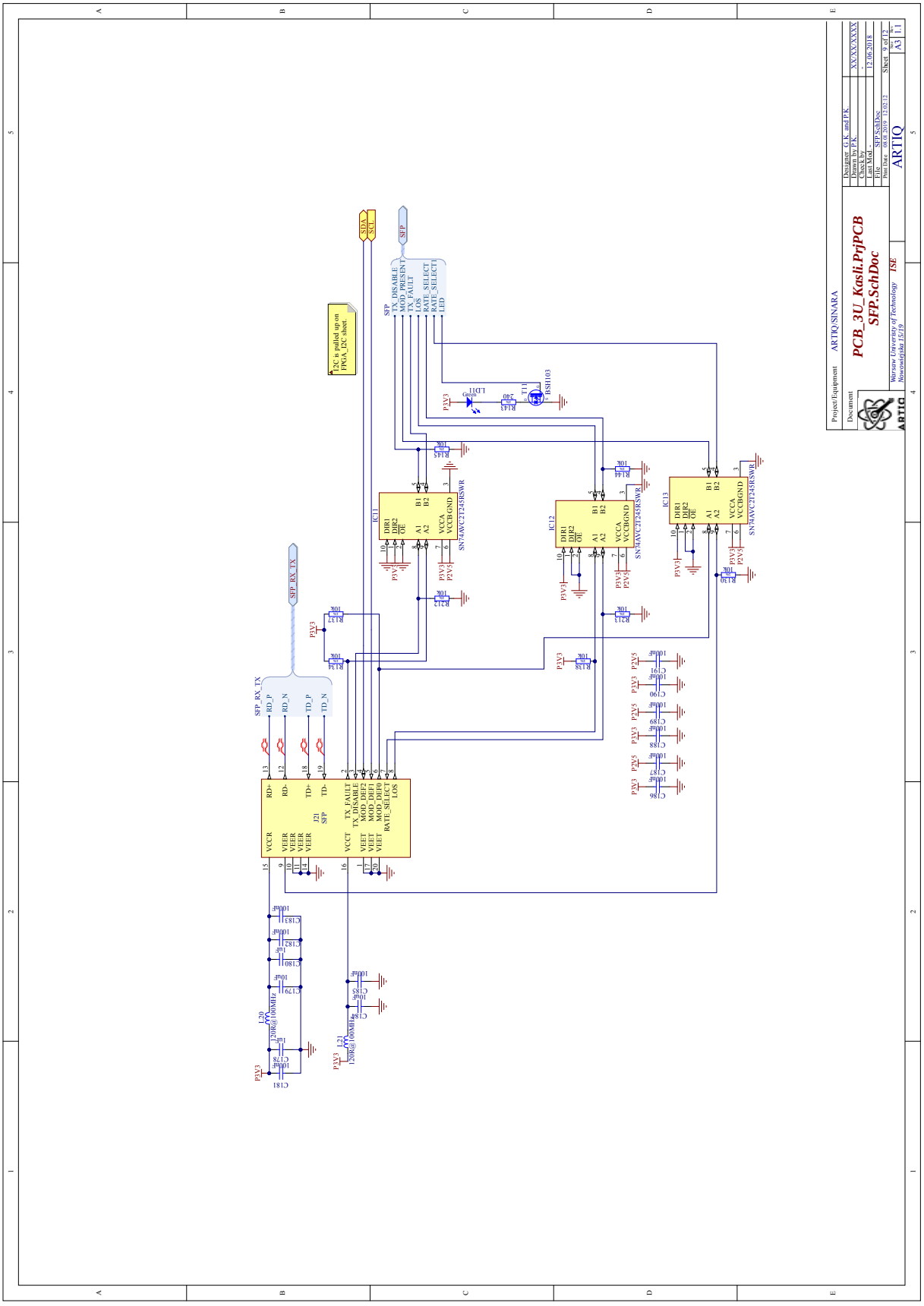

TMS and TDI have internal pull-up resistors, as specified by IEEE Std 1149.1, as do TDO and TCK.
001-master SPI
This pin (CFGVBS) selects the preconfiguration I/O standard type for the dedicated and multi-function configuration banks 0, 14, and 15. If the VCCO for banks 0, 14, or 15 is 2.5V or 3.3V, then this pin must be connect to VCCO_0. If the VCCO for banks 0, 14, and 15 are less than or equal to 1.8V, then this pin should be connected to GND.
IC4J
IC4G
IC4I
XC7A100T-2FGG484C
XADC_VCC
XADC_GND
FPGA_DONE
EN_USB_JTAG
Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB
FPGAPower&Misc.SchDoc
Designer G.K. and P.K.
Drawn by P.K. XX/XX/XXXX
Check by -
Last Mod. - 12.06.2018
File FPGAPower&Misc.SchDoc
Print Date 08.01.2019 12:02:11 Sheet 5 of 12
Warsaw University of Technology Nowowiejska 15/19
ISE
ARTIQ
Size A3 Rev 1.1

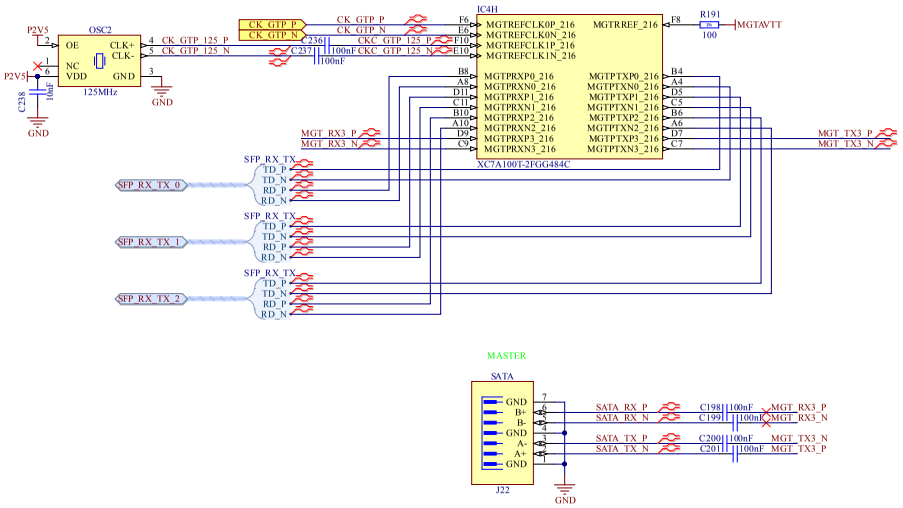


ARTIQ/SINARA
PCB_3U_Kasli.PrjPCB
ClockRecovery.SchDoc
Warsaw University of Technology
ISE
Nowowiejska 15/19
ARTIQ

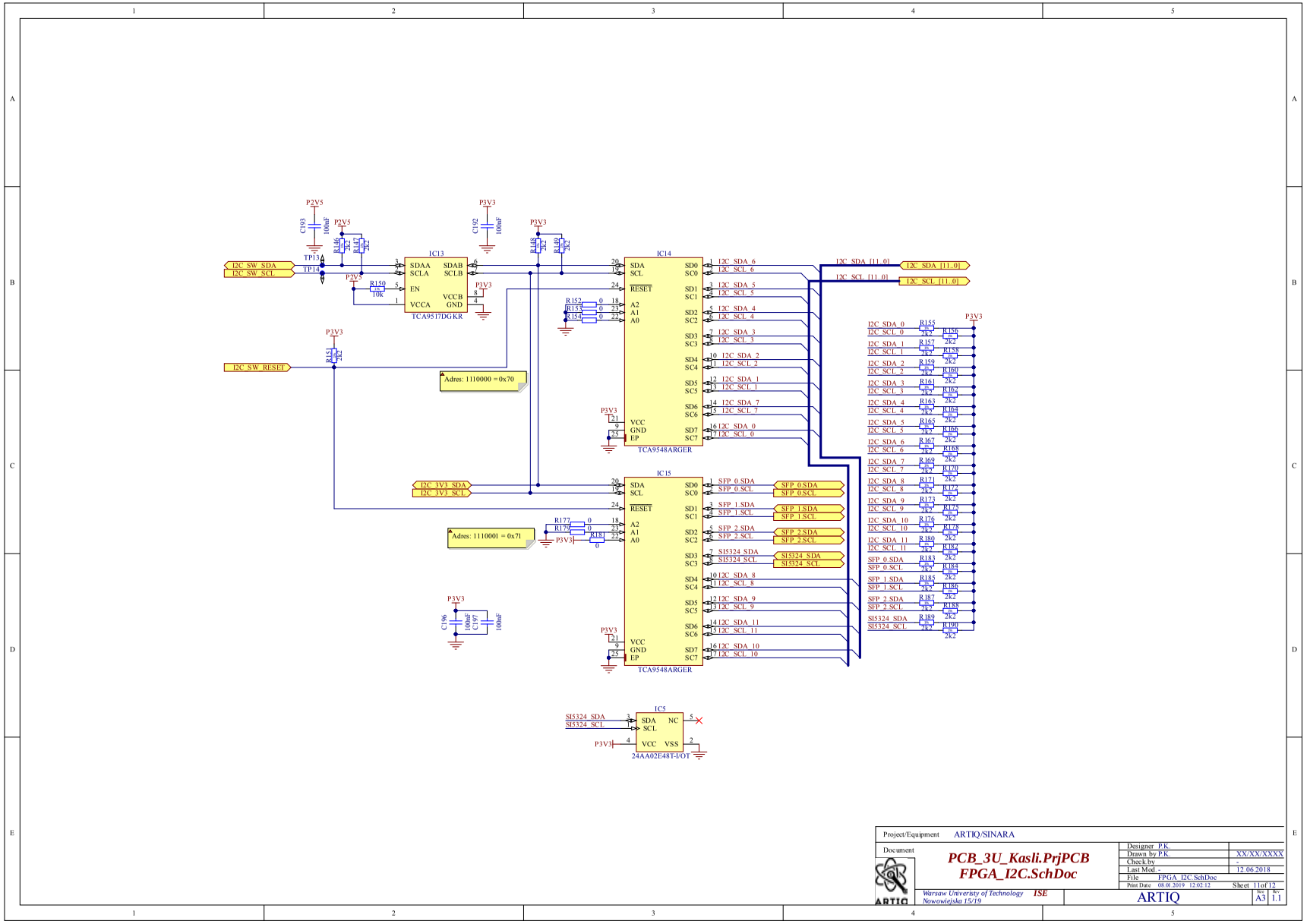

I2C_SDA [11..8]
I2C_SCL [11..8]
ExtensionConnector
EXT8
EXT9
EXT10
EXT11
CLK_EXT_P
CLK_EXT_N
J20A
J20B
J20C
J20D
D11
D12
D13
D14
PRTR5V0U2X
P3V3
I2C_SDA_8
I2C_SCL_8
I2C_SDA_10
I2C_SCL_10
I2C_SDA_9
I2C_SCL_9
I2C_SDA_11
I2C_SCL_11
C31
C32
C33
C34
100nF
GND
MTG1
MTG2
Mounting Holes
Project/Equipment ARTIQ/SINARA
Document PCB_3U_Kasli.PrjPCB
BackplaneConnector.SchDoc
Designer G.K.
Drawn by G.K. and P.K. XX/XX/XXXX
Check.by
Last Mod. 12.06.2018
File BackplaneConnector.SchDoc
Print Date 08.01.2019 12:02:11
Sheet 7 of 12
Warsaw Univeristy of Technology ISE
Nowowiejska 15/19
ARTIQ
Size A3
Rev 1.1


PCB_3U_Kasli.PrjPCB
FPGA_SDRAM.SchDoc
ARTIQ/SINARA
ARTIQ
Warsaw University of Technology
ISE
termination resistors must be placed close to FPGA
termination resistors must be placed close to DDR3 memory
VTT_DDR3
VREF_DDR3
P1V5
P3V3
IC8
MT41K256M16TW-107P
IC4F
BANK 35
XC7A100T-2FGG484C
IC9
TPS51200DRCT
Net Class
ClassName: DDR_BL0
ClassName: DDR_AddressAndControls

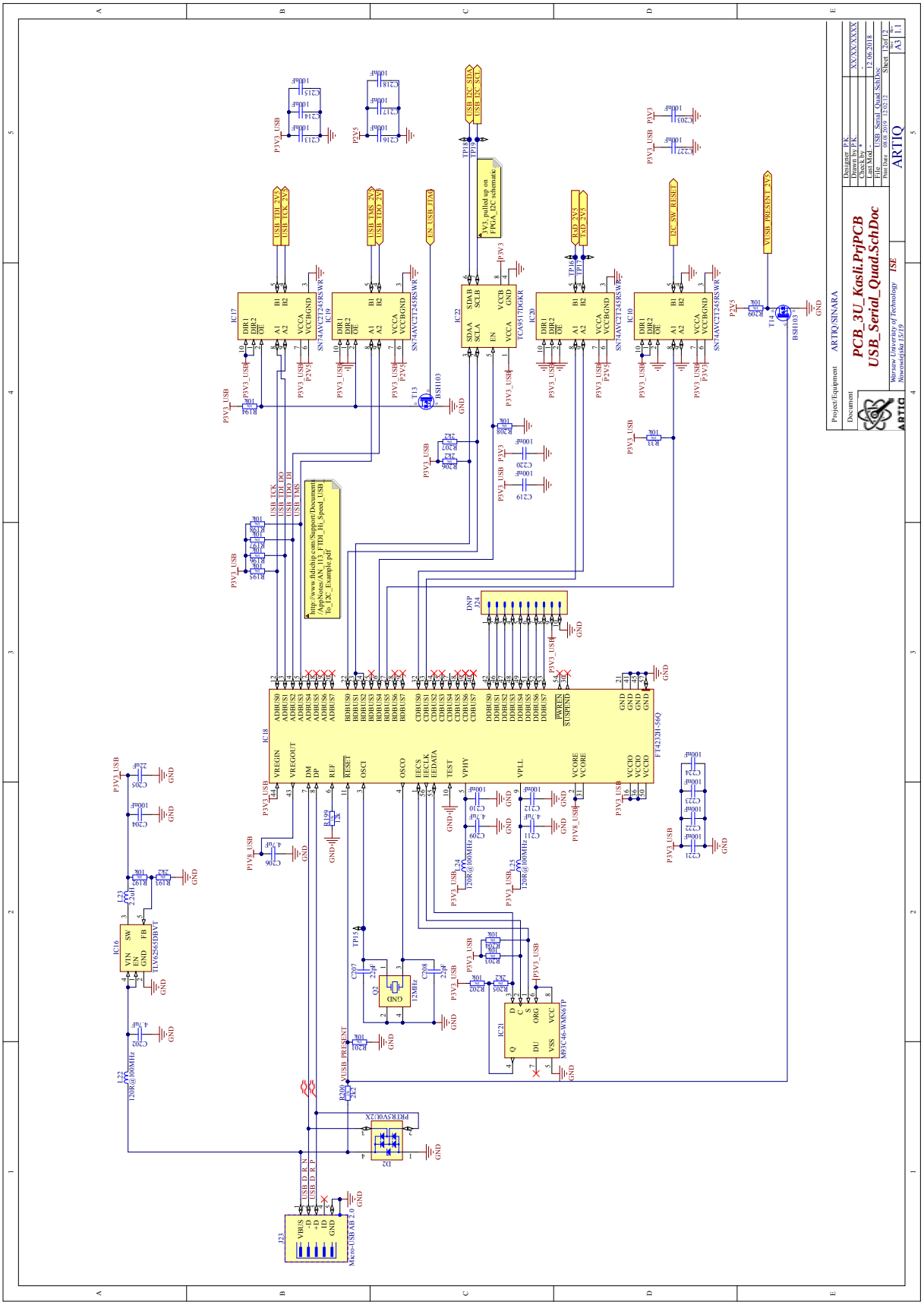

SDA
SCL
SFP
I2C is pulled up on FPGA_I2C sheet
SFP_RX_TX
SFP_RX_TX
RD_P
RD_N
TD_P
TD_N
J21
SFP
TX_DISABLE
MOD_PRESENT
TX_FAULT
LOS
RATE_SELECT
LED
IC11
IC12
IC13
SN74AVC2T245RSWR
P3V3
P2V5
L20
L21
120R@100MHz
Project/Equipment
ARTIQ/SINARA
Document
PCB_3U_Kasli.PrjPCB
SFP.SchDoc
Designer G.K. and P.K.
Drawn by P.K.
Check by
Edit Field
File SFP.SchDoc
Print Date 08.08.2019
12.02.12
Sheet 9 of 12
A3
ARTIQ
Warsaw University of Technology
Nowowiejska 15/19
ISE



Project/Equipment	ARTIQ/SINARA		
Document	PCB_3U_Kasli.PrjPCB FPGAGTP.SchDoc	Designer P.K. Drawn by P.K. Check.by Last Mod. - File FPGAGTPSchDoc Print Date 08.08.2019 12:02:12	XX/XX/XXXX - 12.06.2018 Sheet 10of 12
Warsaw Univeristy of Technology Nowowiejska 15/19	ISE	ARTIQ	Size A3 Rev 1.1



Project/Equipment	ARTIQ/SINARA		
Document	PCB_3U_Kasli.PrjPCB FPGA_I2C.SchDoc	Designer P.K. Drawn by P.K. Check.by Last Mod. - File FPGA_I2C.SchDoc Print Date 08.08.2019 12:02:12	XX/XX/XXXX 12.06.2018 Sheet 11of 12
Warsaw University of Technology Nowowiejska 15/19	ISE	ARTIQ	Size A3 Rev 1.1


PCB_3U_Kasli.PrjPCB
USB_Serial_Quad.SchDoc
ARTIQ/SINARA
ARTIQ
Warsaw University of Technology
ISE
FT4232H-56Q
SN74AVC2T245RSWR
TCA9517DGKR
M95C46-WMN6TP
ULV62565DBVT
PRTR5V0U2X
Micro-USB AB 2.0
BSH103
http://www.ftdichip.com/Support/Documents/AppNotes/AN_113_FTDI_Hi_Speed_USB_To_I2C_Example.pdf
P3V3_USB
P2V5
P1V8_USB
P3V3
P2V5
P1V8
USB_TDI_2V5
USB_TCK_2V5
USB_TMS_2V5
USB_TDO_2V5
USB_I2C_SDA
USB_I2C_SCL
I2C_SW_RESET
VUSB_PRESENT_2V5
EN_USB_JTAG
P3V3_pulled up on FPGA_I2C schematic
Designer P.K.
Drawn by P.K.
Check by *
Edit Field
File USB_Serial_Quad_SchDoc
Print Date 06.09.2019 12:00:12
Sheet 12of12
A3 1.1
Project/Equipment ARTIQ/SINARA
Document

C. Setting up ARTIQ on Ubuntu 16.04

To set up a working development environment for ARTIQ in which I had a possibility to generate my own gateware for FPGA I followed these steps:

1. Install git: `sudo apt install git`
2. Install latest Vivado IDE, downloaded from Xilinx' website in default `/opt/Xilinx` folder (this folder is used by ARTIQ to search for Vivado binary).
3. Install Miniconda for Python 3.x from `http://conda.pydata.org/miniconda.html`.
4. In terminal run:

   ```
   $ conda config --prepend channels http://conda.anaconda.org/conda-forge/label/main
   $ conda config --prepend channels http://conda.anaconda.org/m-labs/label/main
   $ conda config --prepend channels http://conda.anaconda.org/m-labs/label/dev
   ```

   These commands will add a conda-forge, ARTIQ stable and development package repositories.
5. Clone ARTIQ repository (master branch), and install it in editable mode, where changes made to files in ARTIQ repository will be used when using ARTIQ.

   ```
   $ mkdir ~/artiq-dev
   $ git clone --recursive https://github.com/m-labs/artiq ~/artiq-dev/artiq
   $ cd ~/artiq-dev/artiq
   $ conda env create -f conda/artiq-dev.yaml
   $ source activate artiq-dev
   $ pip install -e .
   ```

6. I also repeated install commands for Misoc and Migen repositories cloned from Github (optional):

   ```
   $ cd ~/artiq-dev
   $ git clone --recursive https://github.com/m-labs/misoc.git
   $ cd ~/artiq-dev/misoc
   $ pip install -e .
   $ cd ..
   $ git clone --recursive https://github.com/m-labs/migen.git
   $ cd ~/artiq-dev/migen
   $ pip install -e .
   ```

   This gave me a possibility to change pin assignments in Migen and Ethernet PHY properties in Misoc.
7. Uninstall conda Misoc and Migen packages
8. Configure OpenOCD:

   ```
   $ sudo adduser $USER plugdev
   $ sudo cp ~/.conda/envs/artiq-dev/share/openocd/contrib/60-openocd.rules ↲
       ↳ /etc/udev/rules.d
   $ sudo udevadm trigger
   ```

   And relogin to trigger group change.

Upgrading environment:

1. Pull latest version of ARTIQ (and Misoc and Migen if you also use repository versions):

   ```
   $ cd ~/artiq-dev/artiq
   $ git pull
   ```

   (Repeat for other repositories.)

2. Update environment, as packages used by ARTIQ could have also been updated to newer versions:

   ```
   $ conda env update --file conda/artiq-dev.yaml
   ```

3. Uninstall again any packages you removed previously during installation (e.g. Misoc and Migen).

This concludes setting up development environment. Now, each time a new console is opened `source activate artiq-dev` command has to be run to enable virtual environment with ARTIQ executables appended to `$PATH`. Now to set up ARTIQ on Kasli:

1. Build ARTIQ for Kasli v1.1:

   ```
   $ cd ~/artiq-dev/artiq
   $ python -m artiq.gateware.targets.kasli -V sysu --hw-rev v1.1
   ```

2. Connect 12 V power supply capable of sourcing at least 1 A (more if you connect EEMs to Kasli). Remember to also cool it.
3. Connect USB cable to Kasli and your computer.
4. Connect Kasli to 1 Gbit switch via SFP media converter to SFP0 port.
5. In separate terminal window open `/dev/ttyUSB2` with a program of your choice (e.g. minicom), using 115200 baud rate:

   ```
   $ minicom -D /dev/ttyUSB2
   ```

6. And flash previously built files:

   ```
   $ artiq_flash -t kasli -V sysu --srcbuild ~/artiq-dev/artiq/artiq_kasli
   ```

   You should see output similar to listing in appendix E in your terminal and console output from Kasli similar to one in appendix F.

7. Set IP and MAC addresses appropriate for your network:

   ```
   $ artiq_mkfs flash_storage.img -s mac xx:xx:xx:xx:xx:xx -s ip xx.xx.xx.xx
   $ artiq_flash -t kasli -f flash_storage.img storage start
   ```

   After triggering restart you should be able to ping Kasli on IP address you set up and see link up LED on your switch and on Kasli under SFP0. (**IMPORTANT:** Kasli **has** to be connected to 1 Gbit switch).

Now, to run a basic experiment:

1. Navigate to `~/artiq-dev/artiq/examples/kasli_basic`
2. Edit first line in `device_db_sysu.py` file with IP address you assigned to Kasli in previous steps.
3. Add file `led.py` to `repository` folder:

   ```
   from artiq.experiment import *

   class LED(EnvExperiment):
     def build(self):
       self.setattr_device("core")
       self.setattr_device("led")

     @kernel
     def run(self):
       self.core.reset()
   ```

```
while True:
  self.led.on()
  delay(500*ms)
  self.led.off()
  delay(500*ms)
```

ARTIQ program is called an experiment. Function `run` marked with `@kernel` decorator will run on real-time processor on FPGA. This function turns on and off user LED every second.

4. Now to run this experiment, ensure that Kasli is powered on and connected to your network and run in terminal:

```
$ cd ~/artiq-dev/artiq/examples/kasli_basic
$ artiq_run --device-db device_db_sysu.py repository/led.py
```

You should see one of the LEDs on the PCB switching states. When you are done, exit the infinite loop with `CTRL+C` key combination.

These steps are mostly based on ARTIQ manual, please read it to familiarize yourself with ARTIQ [11]. It includes further instructions and explanations.

D. Code used in project

D.1. FPGA project in Vivado

Listing D.1: Code used for testing GTP links

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;
entity top is
  Port(
    clk_fpgaio_n : IN STD_LOGIC;
    clk_fpgaio_p : IN STD_LOGIC;
    clk_rec_n : OUT STD_LOGIC;
    clk_rec_p : OUT STD_LOGIC;
    clk_gtp_p : IN STD_LOGIC;
    clk_gtp_n : IN STD_LOGIC;
    clk125_gtp_p : IN STD_LOGIC;
    clk125_gtp_n : IN STD_LOGIC;
    sfp_gtp_txp : OUT STD_LOGIC;
    sfp_gtp_txn : out STD_LOGIC;
    sfp_gtp_rxp : IN STD_LOGIC;
    sfp_gtp_rxn : IN STD_LOGIC;
    sfp_gtp_txp_1 : out STD_LOGIC;
    sfp_gtp_txn_1 : OUT STD_LOGIC;
    sfp_gtp_rxp_1 : IN STD_LOGIC;
    sfp_gtp_rxn_1 : IN STD_LOGIC;
    sfp_gtp_txp_2 : OUT STD_LOGIC;
    sfp_gtp_txn_2 : OUT STD_LOGIC;
    sfp_gtp_rxp_2 : IN STD_LOGIC;
    sfp_gtp_rxn_2 : IN STD_LOGIC;
    sata_gtp_txp : OUT STD_LOGIC;
    sata_gtp_txn : OUT STD_LOGIC;
    sata_gtp_rxp : IN STD_LOGIC;
    sata_gtp_rxn : IN STD_LOGIC;
    serial_tx : IN STD_LOGIC;
    serial_rx : OUT STD_LOGIC;
  );
end top;
architecture STRUCTURE of top is
  signal clk_fpgaio : STD_LOGIC;
  signal clk_gtp : STD_LOGIC;
  signal clk125_gtp, clk125_gtp_b : STD_LOGIC;
  signal clk_rec : STD_LOGIC;
  signal clk125, clk200, clk50 : std_logic;
  signal sys_rst, locked : std_logic;

  component ibert_7series_gtp_0
    PORT (
      TXN_O : out STD_LOGIC_VECTOR (3 downto 0) ;
      TXP_O : out STD_LOGIC_VECTOR (3 downto 0) ;
      RXOUTCLK_O : out STD_LOGIC ;
      RXN_I : in STD_LOGIC_VECTOR (3 downto 0) ;
      RXP_I : in STD_LOGIC_VECTOR (3 downto 0) ;
      GTREFCLK0_I : in STD_LOGIC_VECTOR (0 downto 0) ;
      GTREFCLK1_I : in STD_LOGIC_VECTOR (0 downto 0) ;
      SYSCLK_I : in STD_LOGIC
    );
  END COMPONENT;
```

```
begin
  serial_rx <= serial_tx;

  --generate 50 MHz, 125MHz and 200MHz clock from 125MHz input
  clocks : entity work.clocks_7s_extphy
  port map(
    sysclk => clk125_gtp,
    clko_125 => clk125,
    clko_200 => clk200,
    clko_50 => clk50,
    locked   => locked
  );

  u_ibert_core : ibert_7series_gtp_0
  port map(
    TXN_O(0) => sfp_gtp_txn,
    TXN_O(1) => sfp_gtp_txn_1,
    TXN_O(2) => sfp_gtp_txn_2,
    TXN_O(3) => sata_gtp_txn,
    TXP_O(0) => sfp_gtp_txp,
    TXP_O(1) => sfp_gtp_txp_1,
    TXP_O(2) => sfp_gtp_txp_2,
    TXP_O(3) => sata_gtp_txp,
    RXN_I(0) => sfp_gtp_rxn,
    RXN_I(1) => sfp_gtp_rxn_1,
    RXN_I(2) => sfp_gtp_rxn_2,
    RXN_I(3) => sata_gtp_rxn,
    RXP_I(0) => sfp_gtp_rxp,
    RXP_I(1) => sfp_gtp_rxp_1,
    RXP_I(2) => sfp_gtp_rxp_2,
    RXP_I(3) => sata_gtp_rxp,
    RXOUTCLK_O => clk_rec,
    GTREFCLK0_I(0) => clk_gtp,
    GTREFCLK1_I(0) => clk125_gtp_b,
    SYSCLK_I => clk50
  );

  --differential gtp clock buffers
  u_buf_q1_clk0 : IBUFDS_GTE2
    port map (
      O => clk_gtp,
      CEB => '0',
      I => clk_gtp_p,
      IB  => clk_gtp_n
    );

  u_buf_q1_clk1 : IBUFDS_GTE2
    port map (
      O => clk125_gtp_b,
      CEB => '0',
      I => clk125_gtp_p,
      IB  => clk125_gtp_n
    );

   u_bufg_q1_clk1: BUFG
    port map(
      i => clk125_gtp_b,
      o => clk125_gtp
    );

  --differential buffers
  clk_fpgaio_IOBUFDS : unisim.vcomponents.IBUFDS
    port map (
      I => clk_fpgaio_p,
      IB => clk_fpgaio_n,
      O => clk_fpgaio
```

```
    );
  clk_rec_IOBUFDS : unisim.vcomponents.OBUFDS
    port map (
      O => clk_rec_p,
      OB => clk_rec_n,
      I => clk_rec
    );
end STRUCTURE;
```

Listing D.2: Code used for testing EEM connectors

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;
entity top is
  Port(
     clk_fpgaio_n : IN STD_LOGIC;
     clk_fpgaio_p : IN STD_LOGIC;
     clk_rec_n : OUT STD_LOGIC;
     clk_rec_p : OUT STD_LOGIC;
     clk_gtp_p : IN STD_LOGIC;
     clk_gtp_n : IN STD_LOGIC;
     clk125_gtp_p : IN STD_LOGIC;
     clk125_gtp_n : IN STD_LOGIC;
     eem0_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem0_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem1_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem1_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem2_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem2_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem3_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem3_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem4_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem4_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem5_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem5_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem6_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem6_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem7_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem7_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem8_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem8_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem9_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem9_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem10_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem10_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem11_n : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     eem11_p : INOUT STD_LOGIC_VECTOR ( 7 downto 0 );
     user_led1 : OUT STD_LOGIC;
     serial_tx : IN STD_LOGIC;
     serial_rx : OUT STD_LOGIC;
  );
end top;
architecture STRUCTURE of top is
  signal clk_fpgaio : STD_LOGIC;
  signal clk_gtp : STD_LOGIC;
  signal clk125_gtp, clk125_gtp_b : STD_LOGIC;
  signal clk_rec : STD_LOGIC;
  signal eem0_I, eem0_O, eem0_S, eem0_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem1_I, eem1_O, eem1_S, eem1_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem10_I, eem10_O, eem10_S, eem10_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem11_I, eem11_O, eem11_S, eem11_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
```

```
  signal eem2_I, eem2_O, eem2_S, eem2_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem3_I, eem3_O, eem3_S, eem3_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem4_I, eem4_O, eem4_S, eem4_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem5_I, eem5_O, eem5_S, eem5_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem6_I, eem6_O, eem6_S, eem6_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem7_I, eem7_O, eem7_S, eem7_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem8_I, eem8_O, eem8_S, eem8_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem9_I, eem9_O, eem9_S, eem9_vio : STD_LOGIC_VECTOR ( 7 downto 0 );
  signal eem_direction : std_logic_vector ( 11 downto 0 );

  signal clk125, clk200, clk50 : std_logic;
  signal sys_rst, locked : std_logic;

  COMPONENT vio_0
    PORT (
    clk : IN STD_LOGIC;
    probe_in0, probe_in1, probe_in2, probe_in3, probe_in4, probe_in5, probe_in6, ↲
        ↳ probe_in7, probe_in8, probe_in9, probe_in10, probe_in11 : IN ↲
        ↳ STD_LOGIC_VECTOR(7 DOWNTO 0);
    probe_out0, probe_out1, probe_out2, probe_out3, probe_out4, probe_out5, ↲
        ↳ probe_out6, probe_out7, probe_out8, probe_out9, probe_out10, probe_out11 : ↲
        ↳ OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    probe_out12 : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
    probe_out13 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
  END COMPONENT;

  signal io_clk_sel : std_logic_vector(1 downto 0);
  signal io_clk : std_logic_vector(7 downto 0);
  signal io_sel: std_logic;
begin
  --high = input, low = output
  eem0_S <= (others =>  eem_direction(0));
  eem1_S <= (others =>  eem_direction(1));
  eem2_S <= (others =>  eem_direction(2));
  eem3_S <= (others =>  eem_direction(3));
  eem4_S <= (others =>  eem_direction(4));
  eem5_S <= (others =>  eem_direction(5));
  eem6_S <= (others =>  eem_direction(6));
  eem7_S <= (others =>  eem_direction(7));
  eem8_S <= (others =>  eem_direction(8));
  eem9_S <= (others =>  eem_direction(9));
  eem10_S <= (others =>  eem_direction(10));
  eem11_S <= (others =>  eem_direction(11));

  io_clk <= (others => 'Z') when io_clk_sel = "00" else
        (others => clk50) when io_clk_sel = "01" else
        (others => clk125_gtp) when io_clk_sel = "10" else
        (others => clk200) when io_clk_sel = "11";

  eem0_O <= io_clk when io_sel = '0' else eem0_vio;
  eem1_O <= io_clk when io_sel = '0' else eem1_vio;
  eem2_O <= io_clk when io_sel = '0' else eem2_vio;
  eem3_O <= io_clk when io_sel = '0' else eem3_vio;
  eem4_O <= io_clk when io_sel = '0' else eem4_vio;
  eem5_O <= io_clk when io_sel = '0' else eem5_vio;
  eem6_O <= io_clk when io_sel = '0' else eem6_vio;
  eem7_O <= io_clk when io_sel = '0' else eem7_vio;
  eem8_O <= io_clk when io_sel = '0' else eem8_vio;
  eem9_O <= io_clk when io_sel = '0' else eem9_vio;
  eem10_O <= io_clk when io_sel = '0' else eem10_vio;
  eem11_O <= io_clk when io_sel = '0' else eem11_vio;

  serial_rx <= serial_tx;

  EEM_IO : vio_0
```

```
PORT MAP (
  clk => clk50,
  probe_in0 => eem0_l,
  probe_in1 => eem1_l,
  probe_in2 => eem2_l,
  probe_in3 => eem3_l,
  probe_in4 => eem4_l,
  probe_in5 => eem5_l,
  probe_in6 => eem6_l,
  probe_in7 => eem7_l,
  probe_in8 => eem8_l,
  probe_in9 => eem9_l,
  probe_in10 => eem10_l,
  probe_in11 => eem11_l,

  probe_out0 => eem0_vio,
  probe_out1 => eem1_vio,
  probe_out2 => eem2_vio,
  probe_out3 => eem3_vio,
  probe_out4 => eem4_vio,
  probe_out5 => eem5_vio,
  probe_out6 => eem6_vio,
  probe_out7 => eem7_vio,
  probe_out8 => eem8_vio,
  probe_out9 => eem9_vio,
  probe_out10 => eem10_vio,
  probe_out11 => eem11_vio,

  probe_out12 => eem_direction,
  probe_out13(0) => user_led1,
  probe_out13(2 downto 1) => io_clk_sel,
  probe_out13(3) => io_sel
);

--generate 50 MHz, 125MHz and 200MHz clock from 125MHz input
clocks : entity work.clocks_7s_extphy
  port map(
    sysclk => clk125_gtp,
    clko_125 => clk125,
    clko_200 => clk200,
    clko_50 => clk50,
    locked   => locked
);

--differential gtp clock buffers
u_buf_q1_clk0 : IBUFDS_GTE2
  port map (
    O => clk_gtp,
    CEB => '0',
    I => clk_gtp_p,
    IB  => clk_gtp_n
);

u_buf_q1_clk1 : IBUFDS_GTE2
  port map (
    O => clk125_gtp_b,
    CEB => '0',
    I => clk125_gtp_p,
    IB  => clk125_gtp_n
);

u_bufg_q1_clk1: BUFG
  port map(
    i => clk125_gtp_b,
    o => clk125_gtp
);
```

```
--differential buffers
clk_fpgaio_IOBUFDS : unisim.vcomponents.IBUFDS
    port map (
     I => clk_fpgaio_p,
     IB => clk_fpgaio_n,
     O => clk_fpgaio
  );
clk_rec_IOBUFDS : unisim.vcomponents.OBUFDS
    port map (
     O => clk_rec_p,
     OB => clk_rec_n,
     I => clk_rec
  );

-- LVDS bidir buffers
eem0 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem0_I(I),
      IO   => eem0_p(I),
      IOB  => eem0_n(I),
      I => eem0_O(I),
      T => eem0_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem0;
eem1 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem1_I(I),
      IO   => eem1_p(I),
      IOB  => eem1_n(I),
      I => eem1_O(I),
      T => eem1_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem1;
eem2 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem2_I(I),
      IO   => eem2_p(I),
      IOB  => eem2_n(I),
      I => eem2_O(I),
      T => eem2_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem2;
eem3 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem3_I(I),
      IO   => eem3_p(I),
      IOB  => eem3_n(I),
      I => eem3_O(I),
      T => eem3_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem3;
eem4 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem4_I(I),
      IO   => eem4_p(I),
      IOB  => eem4_n(I),
      I => eem4_O(I),
      T => eem4_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem4;
```

```
eem5 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem5_I(I),
      IO   => eem5_p(I),
      IOB  => eem5_n(I),
      I => eem5_O(I),
      T => eem5_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem5;
eem6 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem6_I(I),
      IO   => eem6_p(I),
      IOB  => eem6_n(I),
      I => eem6_O(I),
      T => eem6_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem6;
eem7 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem7_I(I),
      IO   => eem7_p(I),
      IOB  => eem7_n(I),
      I => eem7_O(I),
      T => eem7_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem7;
eem8 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem8_I(I),
      IO   => eem8_p(I),
      IOB  => eem8_n(I),
      I => eem8_O(I),
      T => eem8_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem8;
eem9 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem9_I(I),
      IO   => eem9_p(I),
      IOB  => eem9_n(I),
      I => eem9_O(I),
      T => eem9_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem9;
eem10 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem10_I(I),
      IO   => eem10_p(I),
      IOB  => eem10_n(I),
      I => eem10_O(I),
      T => eem10_S(I) -- 3-state enable input, high=input, low=output
    );
end generate   eem10;
eem11 : for I in 0 to 7 generate
  lvds_buf : IOBUFDS
    PORT MAP (
      O => eem11_I(I),
      IO   => eem11_p(I),
      IOB  => eem11_n(I),
```

```
        I => eem11_O(I),
        T => eem11_S(I) -- 3-state enable input, high=input, low=output
      );
  end generate eem11;
end STRUCTURE;
```

D.2. ARTIQ EEM testing script

Listing D.3: ARTIQ EEM testing experiment

```
from artiq.experiment import *
from artiq.coredevice.exceptions import I2CError


class DioTest(EnvExperiment):
  def build(self):
    self.setattr_device("core")
    self.setattr_device("led0")
    ttls = ["ttl" + str(x) for x in range(96)]
    for s in ttls:
      self.setattr_device(s)
    self.setattr_device("i2c_switch0")
    self.setattr_device("i2c_switch1")
    self.setattr_device("i2c_rj45_dir")

    self.active_eems = [0] * 12
    self.readback = [[2] * 8 * 12] * 3

    # for j in range(12):
    #   for i in range(8):
    #     print("self.ttl{}, ".format(j * 8 + i), end='')
    #   print("")
    self.ttls = [self.ttl0, self.ttl1, self.ttl2, self.ttl3, self.ttl4, self.ttl5, ↲
        ↳ self.ttl6, self.ttl7,    # 0
            self.ttl8, self.ttl9, self.ttl10, self.ttl11, self.ttl12, self.ttl13, ↲
                ↳ self.ttl14, self.ttl15,  # 1
            self.ttl16, self.ttl17, self.ttl18, self.ttl19, self.ttl20, self.ttl21, ↲
                ↳ self.ttl22, self.ttl23,# 2
            self.ttl24, self.ttl25, self.ttl26, self.ttl27, self.ttl28, self.ttl29, ↲
                ↳ self.ttl30, self.ttl31,# 3
            self.ttl32, self.ttl33, self.ttl34, self.ttl35, self.ttl36, self.ttl37, ↲
                ↳ self.ttl38, self.ttl39,# 4
            self.ttl40, self.ttl41, self.ttl42, self.ttl43, self.ttl44, self.ttl45, ↲
                ↳ self.ttl46, self.ttl47,# 5
            self.ttl48, self.ttl49, self.ttl50, self.ttl51, self.ttl52, self.ttl53, ↲
                ↳ self.ttl54, self.ttl55,# 6
            self.ttl56, self.ttl57, self.ttl58, self.ttl59, self.ttl60, self.ttl61, ↲
                ↳ self.ttl62, self.ttl63,# 7
            self.ttl64, self.ttl65, self.ttl66, self.ttl67, self.ttl68, self.ttl69, ↲
                ↳ self.ttl70, self.ttl71,# 8
            self.ttl72, self.ttl73, self.ttl74, self.ttl75, self.ttl76, self.ttl77, ↲
                ↳ self.ttl78, self.ttl79,# 9
            self.ttl80, self.ttl81, self.ttl82, self.ttl83, self.ttl84, self.ttl85, ↲
                ↳ self.ttl86, self.ttl87,#10
            self.ttl88, self.ttl89, self.ttl90, self.ttl91, self.ttl92, self.ttl93, ↲
                ↳ self.ttl94, self.ttl95]#11

  @kernel
  def run(self):
    self.core.reset()

    self.led0.on()

    self.detect_io()
    self.io_test()
```

```
@kernel
def detect_io(self):
  i2c_channels = [7, 5, 4, 3, 2, 1, 0, 6, 4, 5, 7, 6]

  for i in range(12):
    if i < 8:  # Switch 0
      self.i2c_switch0.set(i2c_channels[i])
      self.i2c_switch1.unset()
    else:  # Switch 1
      self.i2c_switch0.unset()
      self.i2c_switch1.set(i2c_channels[i])

    try:
      if i % 2:  # odd eems are inputs
        self.i2c_rj45_dir.set(0x00)  # input
      else:  # even eems are outputs
        self.i2c_rj45_dir.set(0xFF)  # output
    except I2CError:
      pass
    else:
      self.active_eems[i] = 1

@kernel
def set_io(self):
  j = 0
  for i in self.active_eems:
    if i:
      for n in range(8 * j, 8 * j + 8):
        self.core.break_realtime()
        if j % 2:
          self.ttls[n].input()
        else:
          self.ttls[n].output()
    j = j + 1

@kernel
def outputs_on(self):
  j = 0
  for i in self.active_eems:
    if bool(i) and bool(not (j % 2)):
      for n in range(8 * j, 8 * j + 8):
        self.ttls[n].on()
        delay(1 * us)
    j = j + 1

@kernel
def outputs_off(self):
  j = 0
  for i in self.active_eems:
    if bool(i) and bool(not (j % 2)):
      for n in range(8 * j, 8 * j + 8):
        self.ttls[n].off()
        delay(1*us)
    j = j + 1

@kernel
def outputs_aa(self):
  j = 0
  for i in self.active_eems:
    if bool(i) and bool(not (j % 2)):
      for n in range(8 * j, 8 * j + 8):
        if n % 2:
          self.ttls[n].on()
        else:
          self.ttls[n].off()
```

```
            delay(1 * us)
        j = j + 1

    @kernel
    def sample_inputs(self):
        j = 0
        for i in self.active_eems:
            if bool(i) and bool(j % 2):
                for n in range(8 * j, 8 * j + 8):
                    self.ttls[n].sample_input()
                    delay(2*us)
            j = j + 1

    @kernel
    def get_samples(self, k):
        j = 0
        for i in self.active_eems:
            if bool(i) and bool(j % 2):
                for n in range(8 * j, 8 * j + 8):
                    self.readback[k][n] = self.ttls[n].sample_get()
                    delay(1*us)
            j = j + 1

    @kernel
    def io_test(self):
        self.set_io()
        delay(100*us)

        self.outputs_on()
        delay(100*us)
        self.sample_inputs()

        delay(100*us)

        self.outputs_off()
        delay(100*us)
        self.sample_inputs()

        delay(100*us)

        self.outputs_aa()
        delay(100*us)
        self.sample_inputs()

        delay(100*us)
        for i in range(3):
            delay(1*ms)
            self.get_samples(i)

    def analyze(self):
        # print(self.active_eems)
        # [print(i) for i in self.readback]

        passed = [-1] * 8 * 12

        j = 0
        for i in self.active_eems:
            if bool(i) and bool(j % 2):
                for n in range(8 * j, 8 * j + 8):
                    if self.readback[0][n] == 1 and self.readback[1][n] == 0 and ↲
                        ↳ self.readback[2][n] == (n % 2):
                        passed[n] = 1
                    else:
                        passed[n] = 0
            j = j + 1
```

```
    if 0 in passed:
      print("Failed")
      a = [i for i,x in enumerate(passed) if x == 0]
      [print("EEM%d, line %d" % (x//8, x%8)) for x in a]
    else:
      print("Passed, tested: ", end='')
      [print("EEM%d, " % i, end='') for i, x in enumerate(self.active_eems) if x]
      print("")
```

D.3. I²C code used for testing v1.1 version

Built and tested on Ubuntu 16.04. Prerequisites:

- Install libraries and development files for FTDI: `sudo apt install libftdi1 libftdi-dev` (assuming gcc is installed)
- Install open-source MPSSE library:

```
$ git clone https://github.com/devttys0/libmpsse.git
$ cd libmpsse/src
$ ./configure --disable-python
$ make
$ sudo make install
$ sudo ldconfig
```

- Compile: `gcc -o i2c_test i2c_test.c -lftdi -lmpsse`

Listing D.4: Code used for testing I²C and EEPROM on Kasli v1.1

```
#include <stdio.h>
#include <stdbool.h>
#include <ftdi.h>
#include <mpsse.h>


typedef enum
{
    FT4232H_MODE_BITBANG = 7,
    FT4232H_MODE_I2C = 5,
} ft4232h_mode_t;

typedef enum
{
    FT4232H_INTERFACE_A = 1,
    FT4232H_INTERFACE_B = 2,
    FT4232H_INTERFACE_C = 3,
    FT4232H_INTERFACE_D = 4,
} ft4232h_interface;

typedef struct
{
  uint32_t vendor_id;
  uint32_t product_id;
  uint8_t* description;
  uint8_t* serial;

  uint32_t clk_freq;
  ft4232h_mode_t mode[4];
  struct mpsse_context *mpsse[4];

} ft4232h_data;


int ft4232h_get_id_strings(ft4232h_data* ft4232h){
    struct ftdi_device_list *devlist;
```

```
    struct ftdi_context *ftdi;

    char manufacturer[30];
    char description[20];
    char serial[20];
    int mnf_len=sizeof(manufacturer);
    int desc_len=sizeof(description);
    int serial_len=sizeof(serial);

    int id=0;

    if (ft4232h->mpsse[0] == NULL) {
        if ((ftdi = ftdi_new()) == 0) {
            printf("Failed to allocate memory for a new ftdi_context.\n");
            return -1;
        }
    } else {
        ftdi = &ft4232h->mpsse[0]->ftdi;
    }

    // Get device list
    ftdi_usb_find_all(ftdi, &devlist, ft4232h->vendor_id, ft4232h->product_id);
    // Read ID strings from the FTDI device
    ftdi_usb_get_strings(ftdi, devlist[id].dev, manufacturer, mnf_len, description, ↲
        ↳ desc_len, serial, serial_len);

    if (ft4232h->mpsse[0] == NULL) {
        ftdi_free(ftdi);
    }

    printf("Description: %.*s\n", desc_len, description);
    printf("Serial: %.*s\n", serial_len, serial);

    return 0;
}

int write_block_i2c(ft4232h_data *ft4232h, ft4232h_interface interface, uint8_t address↲
    ↳ , const uint8_t* src, size_t len){
    if (ft4232h->mpsse[interface - 1] == NULL || ft4232h->mpsse[interface - 1]->mode !=↲
        ↳ I2C) {
        printf("FTDI I2C interface (%d) not initialized.\n", interface);
        return -1;
    }
    size_t written_bytes = 0;

    if (address > 127) {
        printf("Invalid I2C address (%02x>0x7f)!\n", address);
    }

    printf("Writing %d bytes to I2C slave 0x%02x\n", (int) len, (uint8_t) address);

    // Generate start condition
    Stop(ft4232h->mpsse[interface - 1]);
    Start(ft4232h->mpsse[interface - 1]);

    // Write address
    uint8_t i2c_address = address << 1;

    Write(ft4232h->mpsse[interface - 1], &i2c_address, 1);

    // Verify ACK
    if (GetAck(ft4232h->mpsse[interface - 1]) != 0) {
        printf("Transfer NACK-ed\n");
        return -1;
    }
```

```
    // Iterate over bytes to send
    uint8_t byte;
    for (size_t i = 0; i < len; i++, written_bytes++) {
        byte = src[i];
        Write(ft4232h->mpsse[interface - 1], &byte, 1);

        // Verify ACK
        if (GetAck(ft4232h->mpsse[interface - 1]) != 0) {
            printf("Transfer NACK-ed\n");
            return -1;
        }
    }

    // Generate stop condition
    Stop(ft4232h->mpsse[interface - 1]);

    return written_bytes;
}

int read_block_i2c(ft4232h_data *ft4232h, ft4232h_interface interface, uint8_t address,↲
    ↳ uint8_t* dest, size_t len){
    if (ft4232h->mpsse[interface - 1] == NULL || ft4232h->mpsse[interface - 1]->mode !=↲
        ↳ I2C) {
        printf("FTDI I2C interface (%d) not initialized.\n", interface);
        return -1;
    }

    size_t read_bytes = 0;

    printf("Reading %d bytes from I2C slave 0x%02x\n", (int) len,(uint8_t) address);

    // Generate start condition
    Start(ft4232h->mpsse[interface - 1]);

    // Write address
    uint8_t i2c_address = (address << 1) | 1;
    Write(ft4232h->mpsse[interface - 1], &i2c_address, 1);

    // Verify ACK
    if (GetAck(ft4232h->mpsse[interface - 1]) != 0) {
        printf("Transfer NACK-ed\n");
        return -1;
    }

    // Iterate over bytes to read
    uint8_t* byte;
    SendAcks(ft4232h->mpsse[interface - 1]);
    for (size_t i = 0; i < len; i++, read_bytes++) {
        if (i == len - 1) {
            SendNacks(ft4232h->mpsse[interface - 1]);
        }

        byte = Read(ft4232h->mpsse[interface - 1], 1);
        dest[i] = *byte;
    }

    // Generate stop condition
    Stop(ft4232h->mpsse[interface - 1]);

    return read_bytes;
}

int write_i2c(ft4232h_data* ft4232h, ft4232h_interface interface, uint8_t address, ↲
    ↳ uint8_t data){
    uint8_t value = data & 0xFF;
    write_block_i2c(ft4232h, interface, address, &value, 1);
```

```
    return 1;
}

int read_i2c(ft4232h_data* self, ft4232h_interface interface, uint8_t address, uint8_t ↲
    ↳ *dest){
    uint8_t value = 0;
    read_block_i2c(self, interface, address, &value, 1);
    *dest = value & 0xFF;

    return 1;
}

int read_block(ft4232h_data *ft4232h, uint64_t address, uint8_t* dest, size_t len){
    if (address > 0xFF) {
        return -1;
    }
    printf("Reading␣register␣:␣0x%x␣with␣%lu␣bytes\n", (uint32_t)address, len);

    uint8_t buffer[1];
    buffer[0] = (address & 0xFF);

    write_block_i2c(ft4232h, FT4232H_INTERFACE_B, 0x50, buffer, 1);
    read_block_i2c(ft4232h, FT4232H_INTERFACE_B, 0x50, dest, len);

    return len;
}

void read_eui(ft4232h_data *ft4232h){
    uint8_t eui[6];
    uint8_t eui_start_addr = 0xfa;
    read_block(ft4232h, eui_start_addr, eui, sizeof(eui));

    printf("EEPROM␣EUI:␣");
    for (int i = 0; i < 5; i++) {
        printf("%02x-", eui[i]);
    }
    printf("%02x\n", eui[5]);
}

int main(){
    uint8_t sw0, sw1;
    ft4232h_data ft4232h = {
        .vendor_id = 0x0403,
        .product_id = 0x6011,

        .clk_freq = ONE_HUNDRED_KHZ,

        .mode[1] = FT4232H_MODE_I2C,

        .mpsse[0] = NULL,
        .mpsse[1] = NULL,
        .mpsse[2] = NULL,
        .mpsse[3] = NULL
    };

    ft4232h.mpsse[1] = Open(ft4232h.vendor_id, ft4232h.product_id, ft4232h.mode[1], ↲
        ↳ ft4232h.clk_freq, MSB, 2, ft4232h.description, ft4232h.serial);
    if (ft4232h.mpsse[1] == NULL || ft4232h.mpsse[1]->open != 1) {
        printf("Failed␣to␣initialize␣FTDI␣interface␣(%d).\n", 1 + 1);
        return -1;
    }

    // Get device description and serial
    ft4232h_get_id_strings(&ft4232h);
```

```
    // Enable voltage translator
    PinHigh(ft4232h.mpsse[1], 0);
    PinHigh(ft4232h.mpsse[1], 1);

    // Check reset of switches
    write_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x70, 1 << 7);
    write_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x71, 1 << 3);


    read_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x70, &sw0);
    read_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x71, &sw1);
    printf("Before␣reset:\tSwitch␣0:␣%02x\tSwitch␣1:␣%02x\n", sw0, sw1);

    // Reset
    PinLow(ft4232h.mpsse[1], 1);
    PinHigh(ft4232h.mpsse[1], 1);
    printf("Reset\n");

    // Workaround a bug in a library
    Start(ft4232h.mpsse[1]);
    Stop(ft4232h.mpsse[1]);

    read_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x70, &sw0);
    read_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x71, &sw1);
    printf("After␣reset:\tSwitch␣0:␣%02x\tSwitch␣1:␣%02x\n", sw0, sw1);

    // Read EEPROM EUI
    write_i2c(&ft4232h, FT4232H_INTERFACE_B, 0x71, 1 << 3);
    read_eui(&ft4232h);

    Close(ft4232h.mpsse[1]);

    return 0;
}
```

E. OpenOCD listing

Listing E.1: openOCD listing

```
Design: top;UserID=FFFFFFFF;COMPRESS=TRUE;Version=2018.2
Part name: 7a100tfgg484
Date: 2018/11/28
Time: 16:15:38
Bitstream payload length: 0x2a4d34
Open On-Chip Debugger 0.10.0-00013-gbb7beda (2018-02-13-15:56)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
none separate
adapter speed: 25000 kHz
Info : ftdi: if you experience problems at higher adapter clocks, try the command "↲
    ↳ ftdi_tdo_sample_edge falling"
Info : clock speed 25000 kHz
Info : JTAG tap: xc7.tap tap/device found: 0x13631093 (mfg: 0x049 (Xilinx), part: 0↲
    ↳ x3631, ver: 0x1)
Info : gdb server disabled
TEMP 56.49 C
VCCINT 0.990 V
VCCAUX 1.780 V
VCCBRAM 0.995 V
VPVN 0.000 V
VREFP 0.000 V
VREFN 0.000 V
VCCPINT 0.000 V
VCCPAUX 0.000 V
VCCODDR 0.000 V
loaded file /home/pawel/miniconda3/envs/artiq-dev/share/bscan-spi-bitstreams/↲
    ↳ bscan_spi_xc7a100t.bit to pld device 0 in 0s 232663us
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
flash 'jtagspi' found at 0x00000000
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
Info : sector 0 took 1878 ms
Info : sector 1 took 1888 ms
Info : sector 2 took 108 ms
Info : sector 3 took 119 ms
Info : sector 4 took 103 ms
Info : sector 5 took 102 ms
Info : sector 6 took 111 ms
Info : sector 7 took 113 ms
Info : sector 8 took 96 ms
Info : sector 9 took 107 ms
Info : sector 10 took 106 ms
Info : sector 11 took 111 ms
Info : sector 12 took 101 ms
Info : sector 13 took 111 ms
Info : sector 14 took 102 ms
Info : sector 15 took 103 ms
Info : sector 16 took 109 ms
Info : sector 17 took 102 ms
Info : sector 18 took 103 ms
Info : sector 19 took 108 ms
Info : sector 20 took 102 ms
Info : sector 21 took 101 ms
Info : sector 22 took 104 ms
Info : sector 23 took 102 ms
Info : sector 24 took 103 ms
Info : sector 25 took 104 ms
Info : sector 26 took 107 ms
Info : sector 27 took 107 ms
Info : sector 28 took 109 ms
```

```
Info : sector 29 took 103 ms
Info : sector 30 took 104 ms
Info : sector 31 took 104 ms
Info : sector 32 took 101 ms
Info : sector 33 took 113 ms
Info : sector 34 took 105 ms
Info : sector 35 took 101 ms
Info : sector 36 took 108 ms
Info : sector 37 took 103 ms
Info : sector 38 took 98 ms
Info : sector 39 took 104 ms
Info : sector 40 took 98 ms
Info : sector 41 took 97 ms
Info : sector 42 took 107 ms
erased sectors 0 through 42 on flash bank 1 in 8.067723s
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
wrote 2772276 bytes from file /tmp/artiq_bdttmiwm_top.bit to flash bank 1 at offset 0↲
    ↳ x00000000 in 17.909012s (151.170 KiB/s)
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
read 2772276 bytes from file /tmp/artiq_bdttmiwm_top.bit and flash bank 1 at offset 0↲
    ↳ x00000000 in 1.617850s (1673.394 KiB/s)
contents match
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
flash 'jtagspi' found at 0x00000000
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
Info : sector 64 took 112 ms
Info : sector 65 took 118 ms
erased sectors 64 through 65 on flash bank 1 in 0.230112s
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
wrote 73108 bytes from file /home/pawel/artiq-dev/artiq/builds/kasli_latest_sysu/sysu/↲
    ↳ software/bootloader/bootloader.bin to flash bank 1 at offset 0x00400000 in ↲
    ↳ 0.474333s (150.516 KiB/s)
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
read 73108 bytes from file /home/pawel/artiq-dev/artiq/builds/kasli_latest_sysu/sysu/↲
    ↳ software/bootloader/bootloader.bin and flash bank 1 at offset 0x00400000 in ↲
    ↳ 0.043349s (1646.971 KiB/s)
contents match
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
flash 'jtagspi' found at 0x00000000
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
Info : sector 69 took 112 ms
Info : sector 70 took 110 ms
Info : sector 71 took 104 ms
Info : sector 72 took 111 ms
Info : sector 73 took 106 ms
Info : sector 74 took 107 ms
Info : sector 75 took 98 ms
Info : sector 76 took 108 ms
Info : sector 77 took 117 ms
erased sectors 69 through 77 on flash bank 1 in 0.974010s
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
wrote 543560 bytes from file /home/pawel/artiq-dev/artiq/builds/kasli_latest_sysu/sysu/↲
    ↳ software/runtime/runtime.fbi to flash bank 1 at offset 0x00450000 in 3.504707s ↲
    ↳ (151.459 KiB/s)
Info : Found flash device 'sp s25fl128' (ID 0x00182001)
read 543560 bytes from file /home/pawel/artiq-dev/artiq/builds/kasli_latest_sysu/sysu/↲
    ↳ software/runtime/runtime.fbi and flash bank 1 at offset 0x00450000 in 0.320616s ↲
    ↳ (1655.626 KiB/s)
contents match
```

F. ARTIQ boot output on Kasli

Listing F.1: openOCD listing

```
 __   __ _ _____           ____
|  \/  (_) ___|  ___   / ___|
| |\/| | \___ \ / _ \| |
| |  | | |___) | (_) | |___
|_|  |_|_|____/ \___/ \____|

MiSoC Bootloader
Copyright (c) 2017-2018 M-Labs Limited

Bootloader CRC passed
Gateware ident 5.0.dev+127.g57caa7b1;sysu
Initializing SDRAM...
Read leveling scan:
Module 1:
00000000001111111111100000000000
Module 0:
00000000000111111111110000000000
Read leveling: 15+-5 16+-5 done
SDRAM initialized
Memory test passed

Booting from flash...
Starting firmware.
[     0.000008s]  INFO(runtime): ARTIQ runtime starting...
[     0.003928s]  INFO(runtime): software ident 5.0.dev+127.g57caa7b1;sysu
[     0.010471s]  INFO(runtime): gateware ident 5.0.dev+127.g57caa7b1;sysu
[     0.017008s]  INFO(runtime): log level set to INFO by default
[     0.022739s]  INFO(runtime): UART log level set to INFO by default
[     0.028917s]  INFO(runtime): press 'e' to erase startup and idle kernels...
[     1.030570s]  INFO(runtime): continuing boot
[     1.311543s]  INFO(board_artiq::si5324): waiting for Si5324 lock...
[     9.481708s]  INFO(board_artiq::si5324):    ...locked
[     9.485382s]  WARN(runtime): using default MAC address 02-00-00-00-00-21; consider ↲
    ↳ changing it
[     9.493989s]  INFO(runtime): using default IP address 192.168.1.70
[     9.502121s]  INFO(runtime::mgmt): management interface active
[     9.516266s]  INFO(runtime::session): accepting network sessions
[     9.531384s]  INFO(runtime::session): running startup kernel
[     9.535887s]  INFO(runtime::session): no startup kernel found
[     9.541663s]  INFO(runtime::session): no connection, starting idle kernel
[     9.548546s]  INFO(runtime::session): no idle kernel found
```

G. Kasli v1.0 phase noise plots

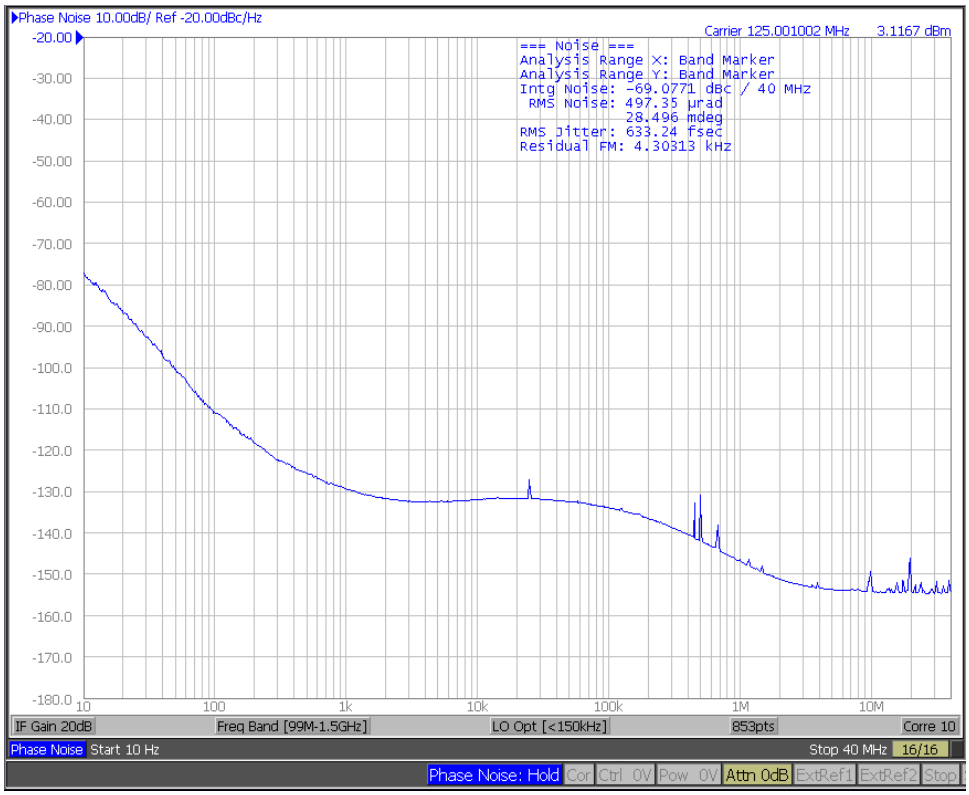


Figure G.1: MMCX clock output phase noise – J1 connector

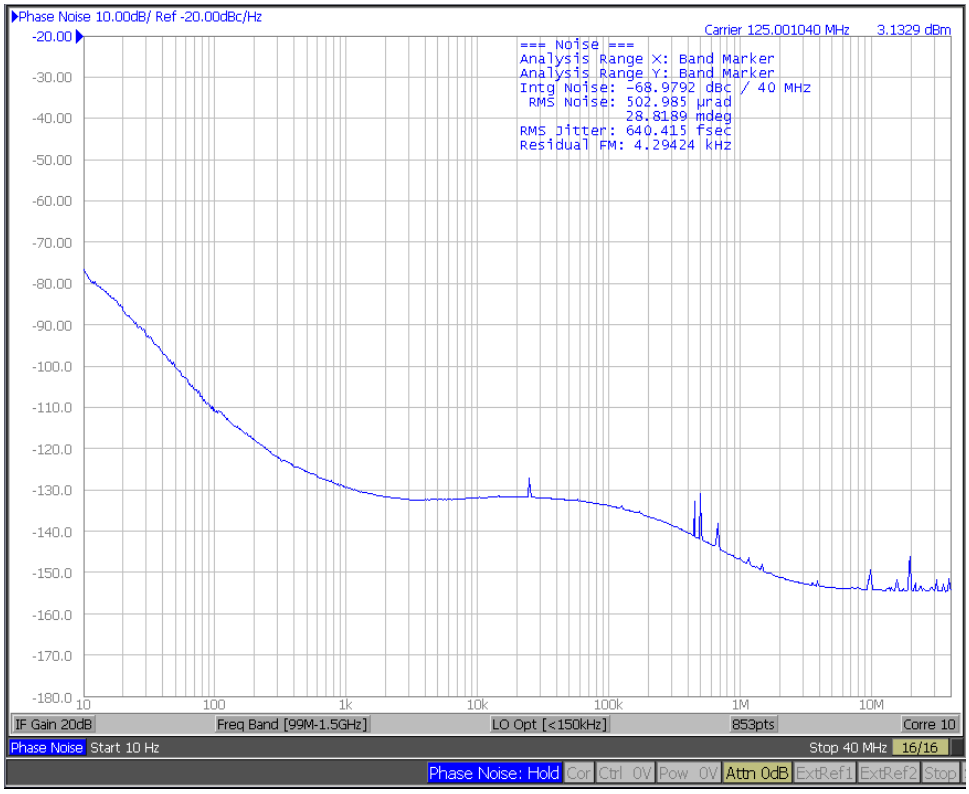


Figure G.2: MMCX clock output phase noise – J2 connector

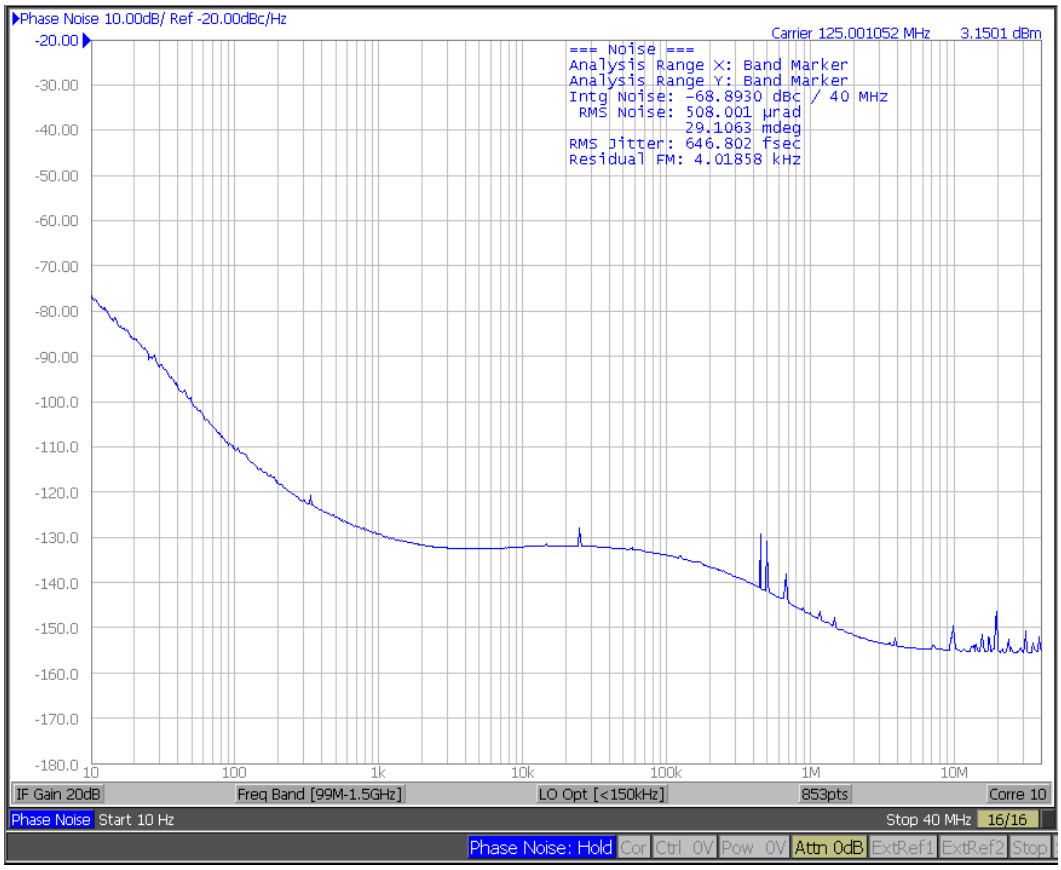


Figure G.3: MMCX clock output phase noise – J3 connector

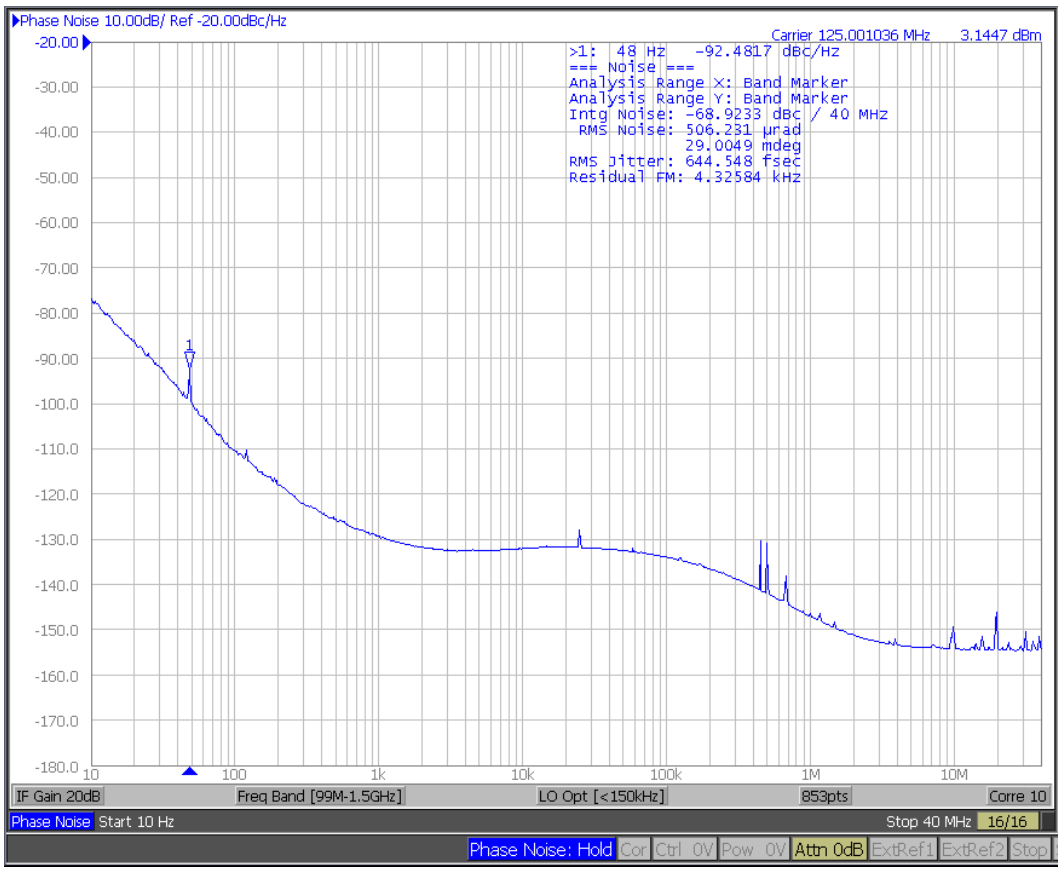


Figure G.4: MMCX clock output phase noise – J4 connector

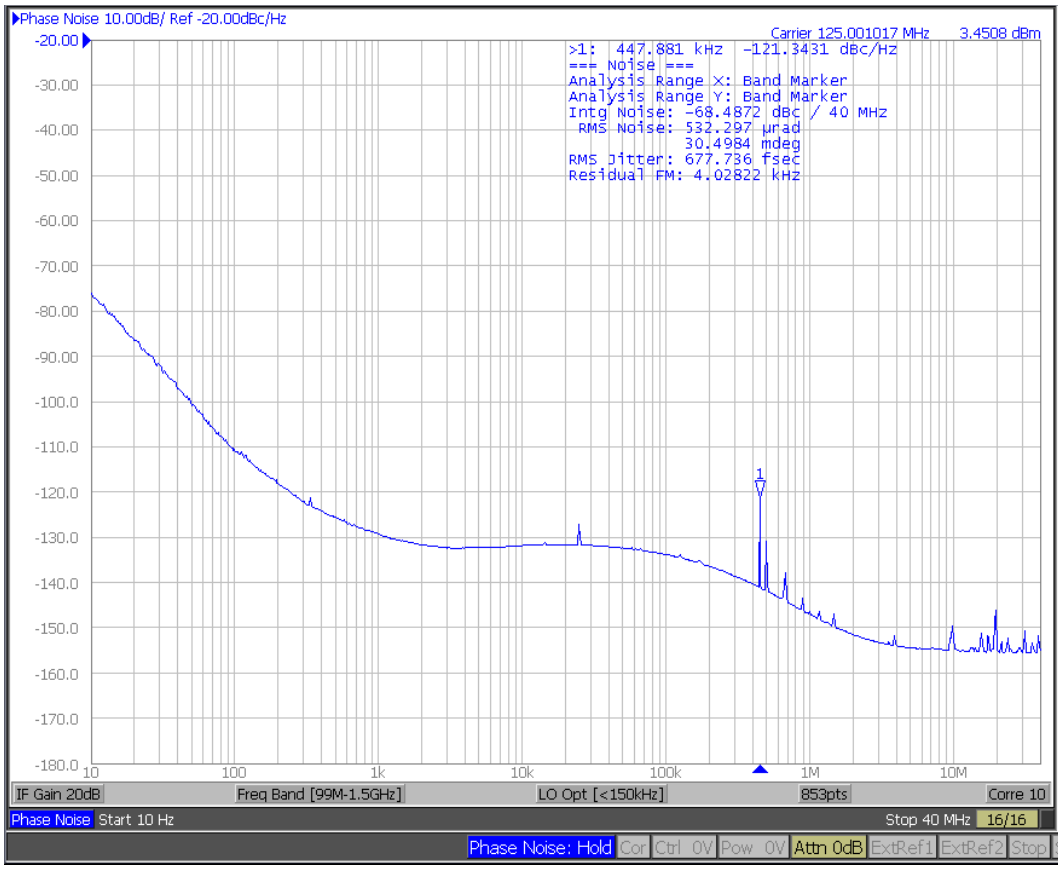


Figure G.5: MMCX clock output phase noise – J5 connector

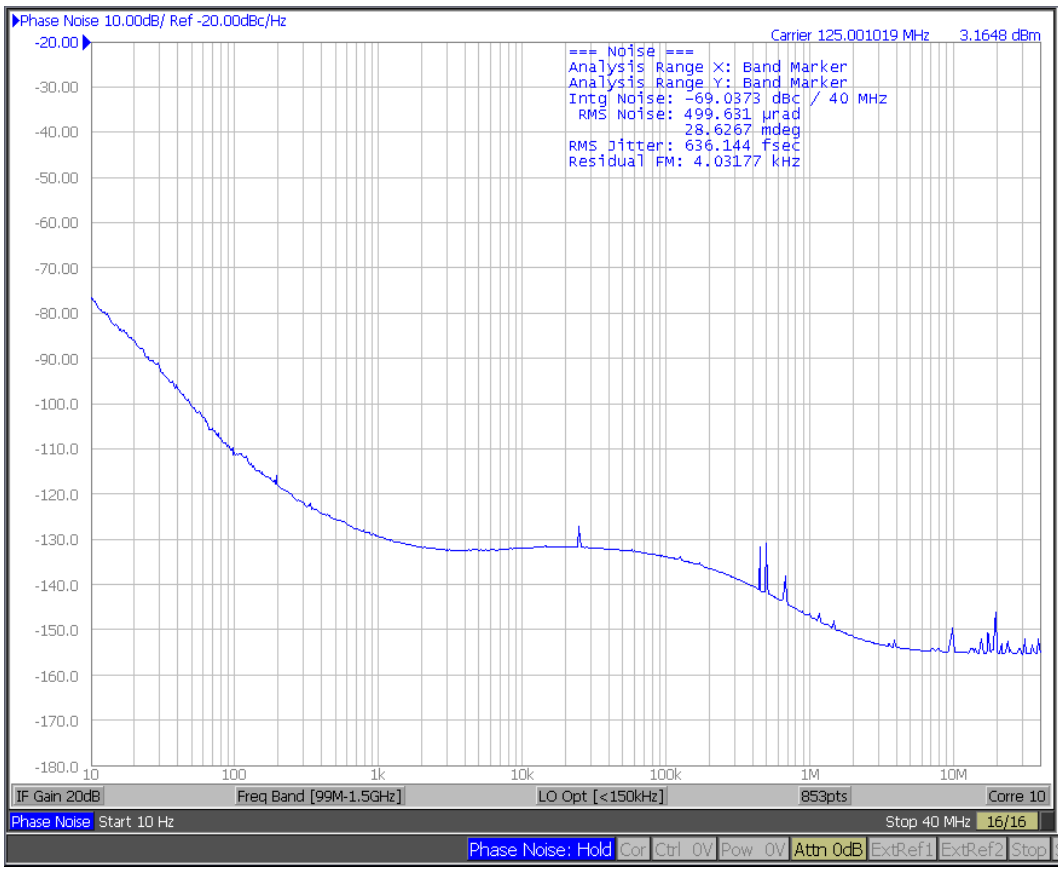


Figure G.6: MMCX clock output phase noise – J6 connector

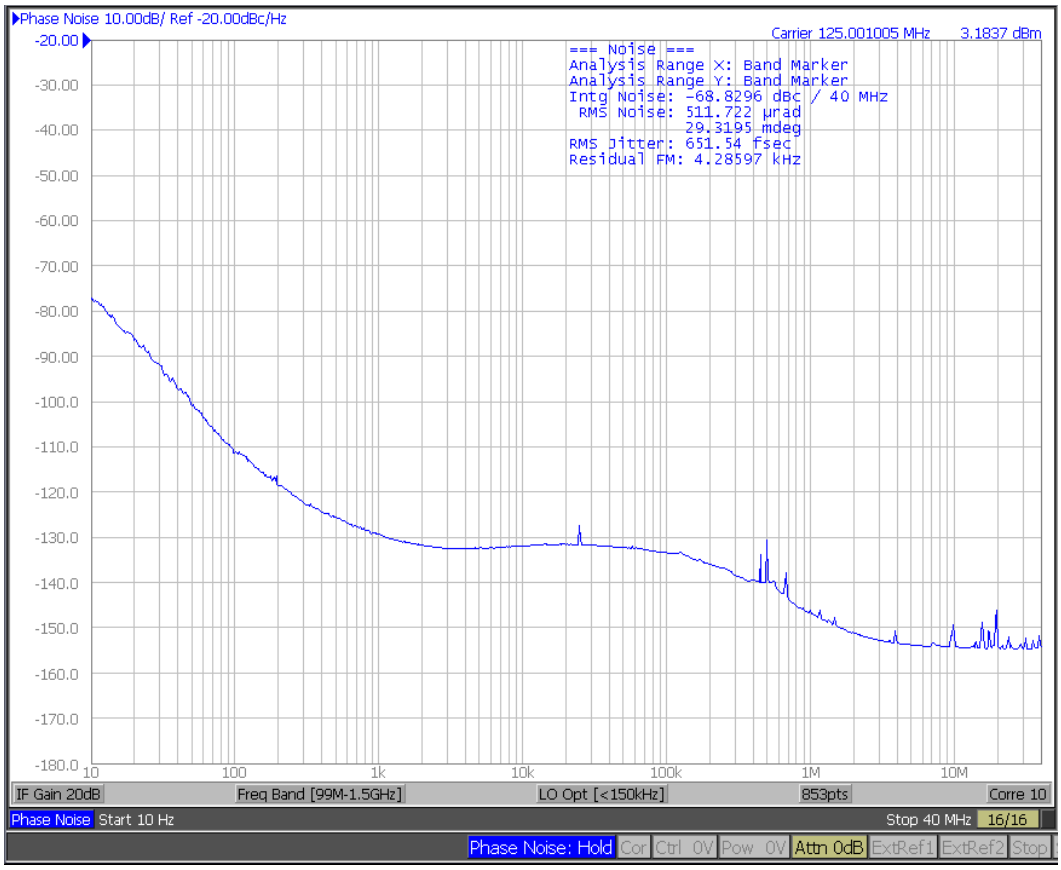


Figure G.7: MMCX clock output phase noise – J7 connector

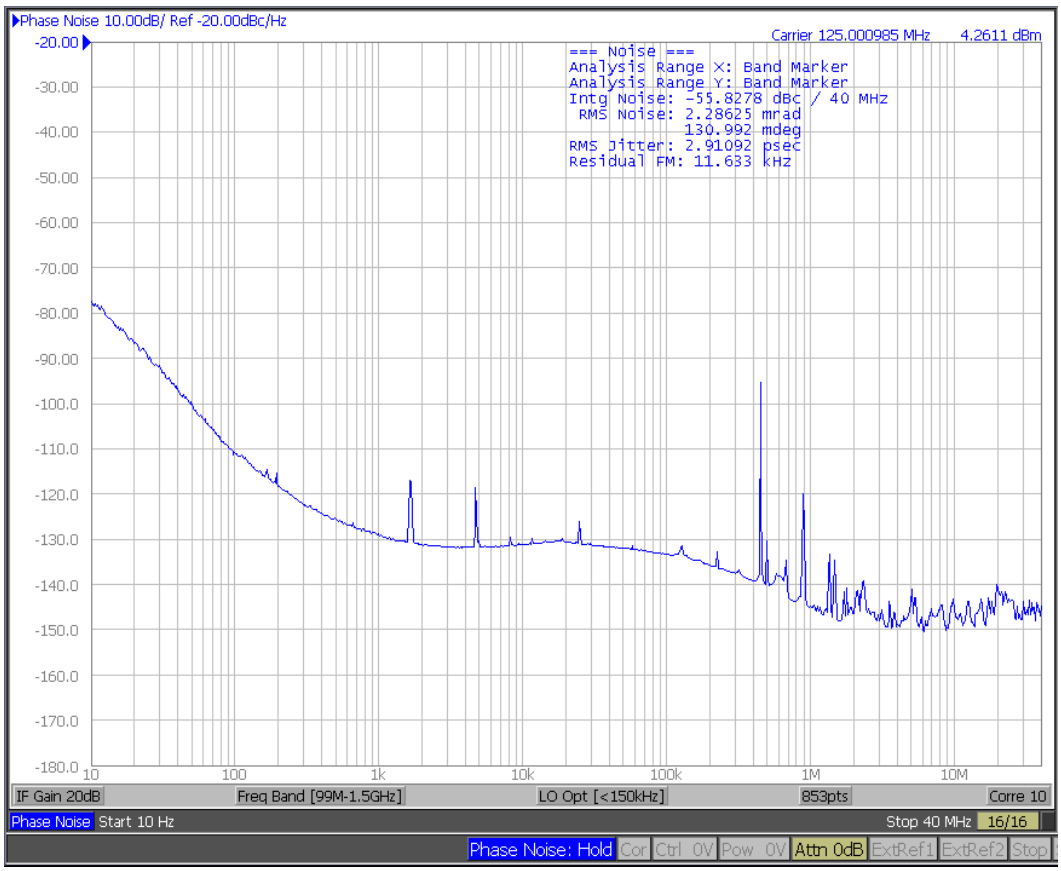


Figure G.8: MMCX clock output phase noise – connector on backplane adapter

H. Kasli v1.1 phase noise plots

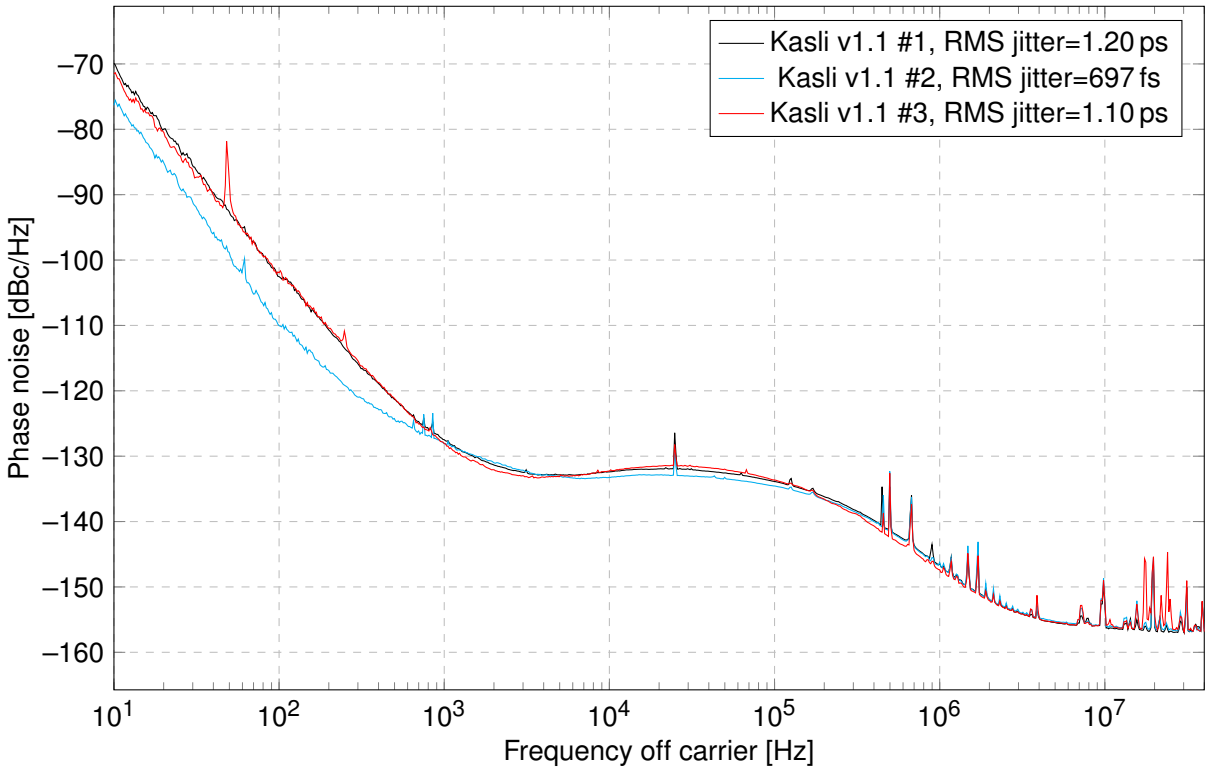


Figure H.1: Phase noise comparison between different Kasli v1.1 – J1 MMCX connector

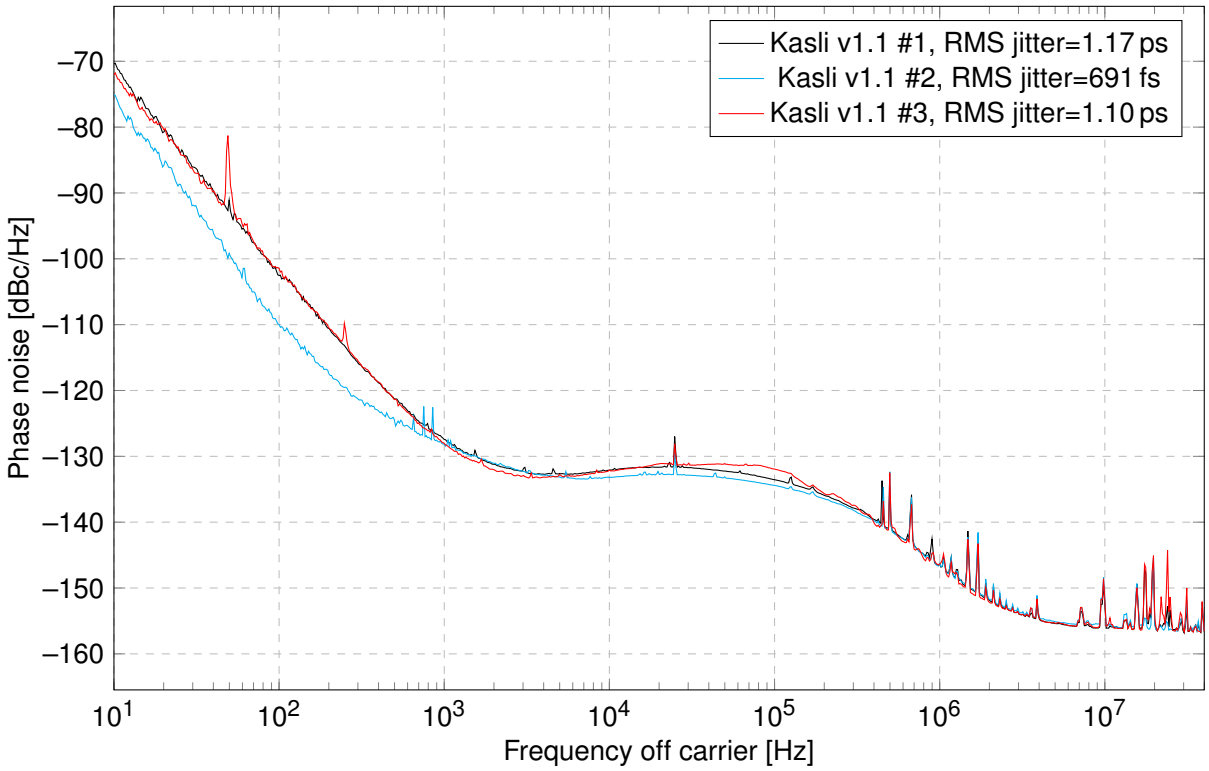


Figure H.2: Phase noise comparison between different Kasli v1.1 – J2 MMCX connector

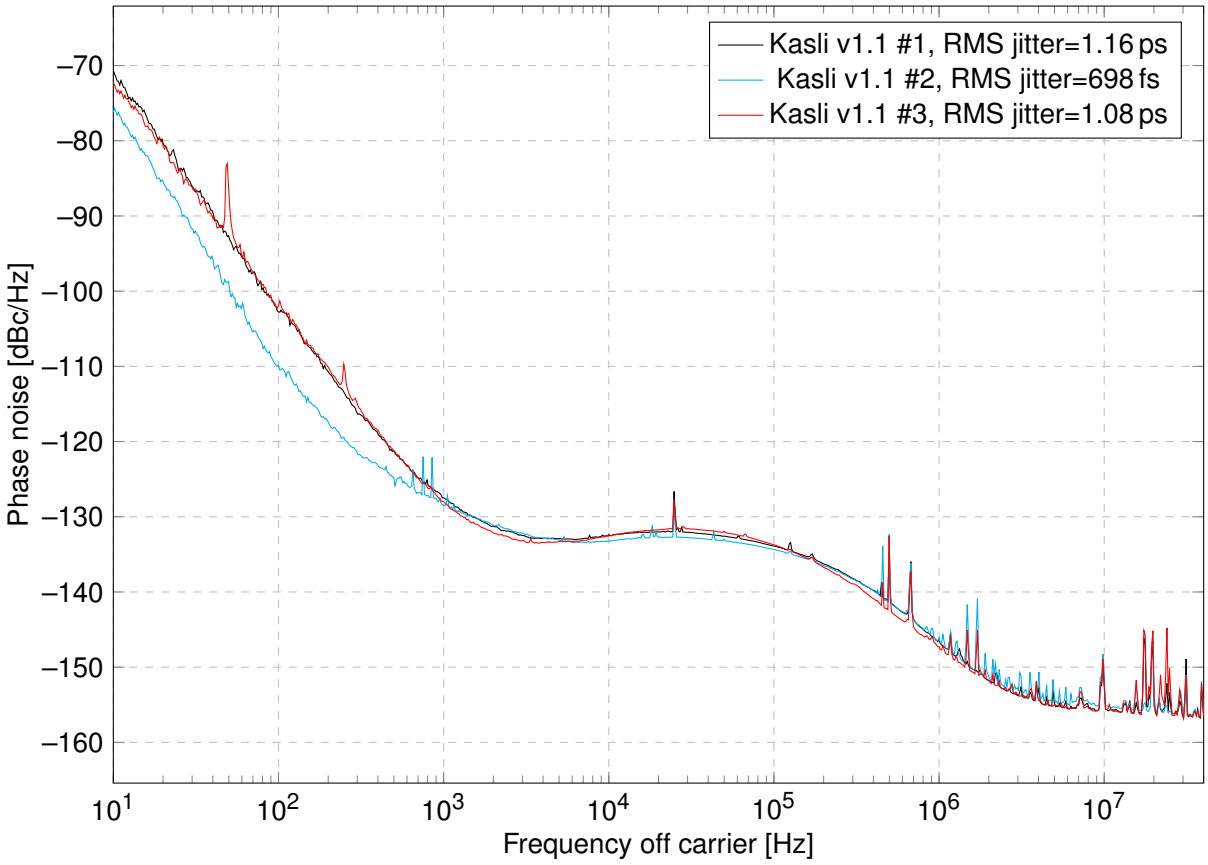


Figure H.3: Phase noise comparison between different Kasli v1.1 – J3 MMCX connector

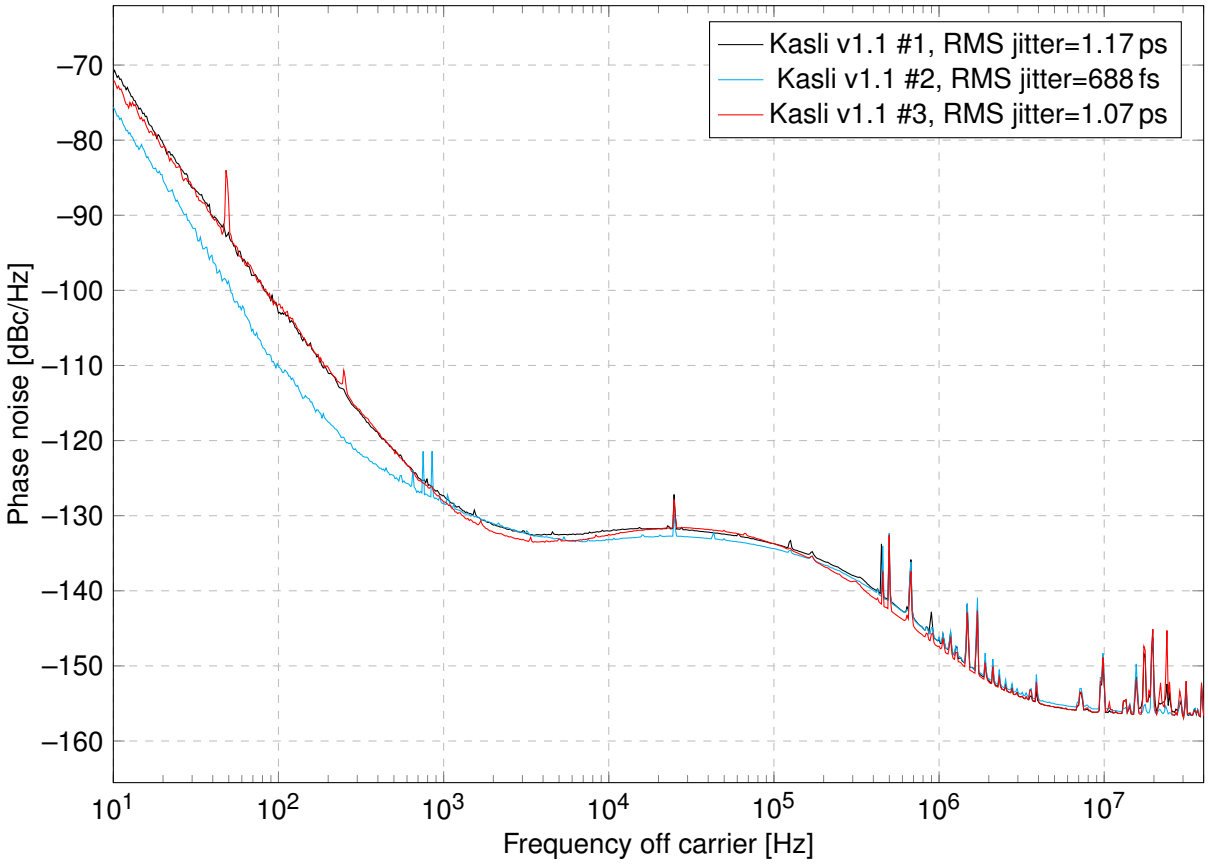


Figure H.4: Phase noise comparison between different Kasli v1.1 – J4 MMCX connector