FPGA Board Reference Guide

Author: Rico Möckel
Supervisor: Prof. Auke Jan Ijspeert

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

BIOLOGICALLY INSPIRED
ROBOTICS GROUP (BIRG)

FPGA Board Reference Guide

Date	Version	Changes
October 18, 2004	1.0	-

Table of Contents

Purpose

This document describes the FPGA Board rev 1 (2004-08-30) designed for the Modular Robot Unit from BIRG (Biologically Inspired Robotics Group).

Audience

This document is written for users of the Modular Robot Unit from BIRG or just of the FPGA board alone. The user should be familiar with the ISE development environment from XILINX and should have some basic knowledge about FPGA.

Organization

The following chapters “Overview” and “Getting Started / First Test of FPGA Board” will give a first introduction in using the FPGA board while the chapters afterwards go more into detail. For schematics and assembly instruction please see the appendix.

Reference

In this document the following references are used.

[1]	Spartan3 datasheet	http://www.xilinx.com/bvdocs/publications/ds099.pdf
[2]	K6R4016 datasheet	http://www.samsung.com/Products/Semiconductor/SRAM/AsyncFastSRAM/4Mbit/K6R4016V1D/ds_k6r4016v1d_rev40.pdf
[3]	Bluetooth Board Reference Guide	

Table 1 References

Overview

The FPGA board contains the following main components:

- XC3S400-4TQ144C
  Spartan3 FPGA with 400000 gates
- 256Kx16 bit high speed SRAM (10ns)
- JTAG connector
- Connector providing access to 8 GPIO pins
- Push button for resetting the FPGA
- 1 GPIO LED (red)
- LED connected to DONE pin of FPGA (green)
- Access to slave serial configuration pins via pads
- Ultra low dropout regulator with 2.5V output
- Oscillator with 50 MHz

A picture of the top and bottom side of the FPGA board is shown in figure 1 and 2.

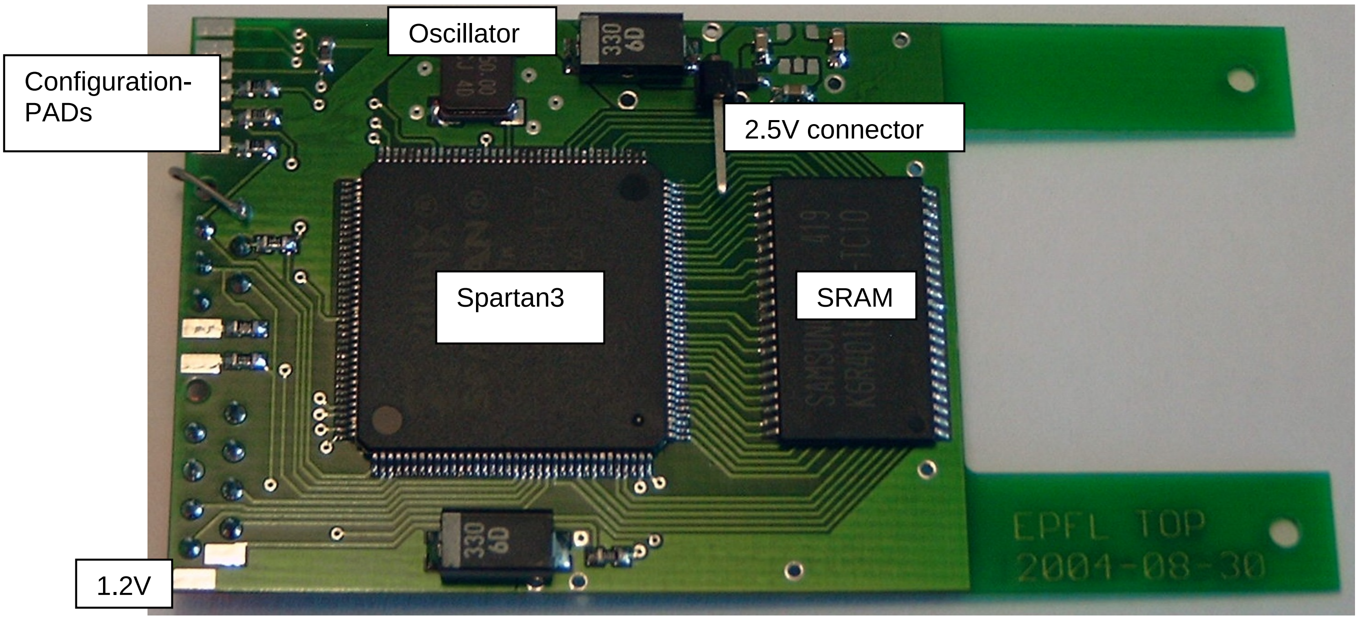


Figure 1 FPGA Board top

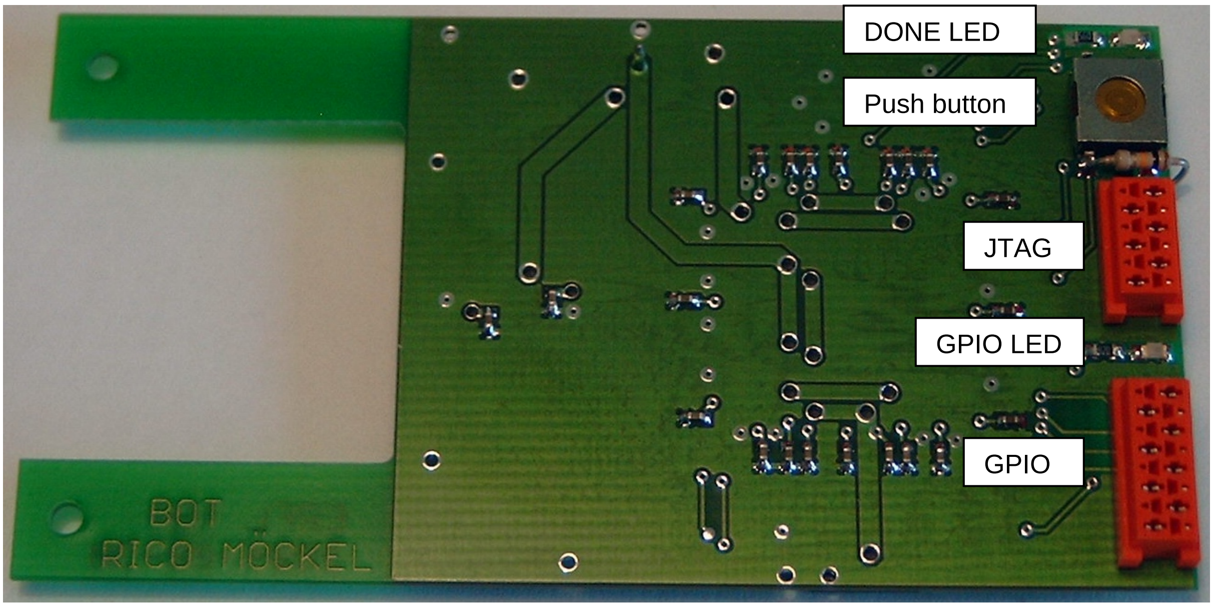


Figure 2 FPGA Board bottom

Getting Started / First Test of FPGA Board

There are two small VHDL programs written for the FPGA board to do a first test.

- Test1Spartan3 contains a VDHL program "led.vhd" that lets the GPIO-LED blinking.
- Test2Spartan3 contains a VDHL program "led.vhd" to control the GPIO-LED with the help of the push button.

For loading one of the programs into the Spartan3 please do the following:

A Hardware setup

When the FPGA board is not included in a robot unit

1. Connect the Ground to the FPGA board.
2. Connect the 3.3V and 1.2V power supply with the FPGA board.
3. Connect the hardware programmer to the PC and to the JTAG connector of the FPGA board. The reference voltage of the hardware programmer should be connected to the 2.5V connector (please see figure 1).

When the FPGA board is already included in a robot unit

1. Switch the power supply on
2. Connect the JTAG connector to the FPGA board.

B Programming of the FPGA

1. Start the ISE Development Environment from Xilinx by double clicking the project file "Test1Spartan3.npl" or "Test2Spartan3.npl".
2. Double click "Configure device (iMPACT)". Please see figure 3.

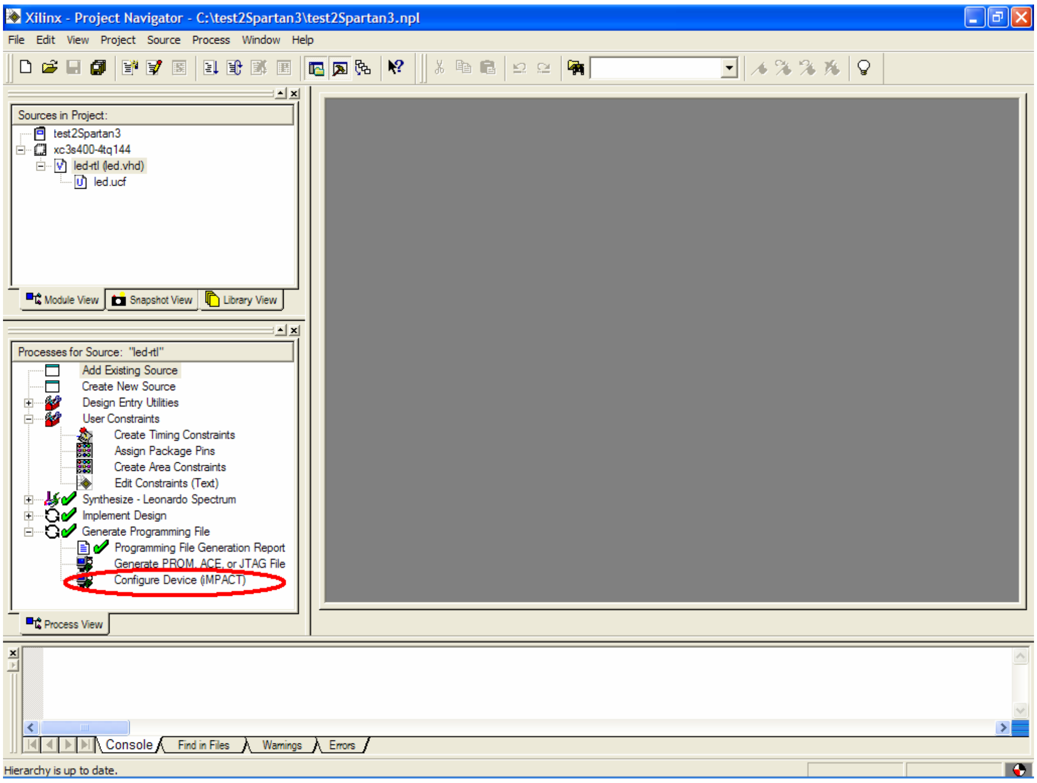
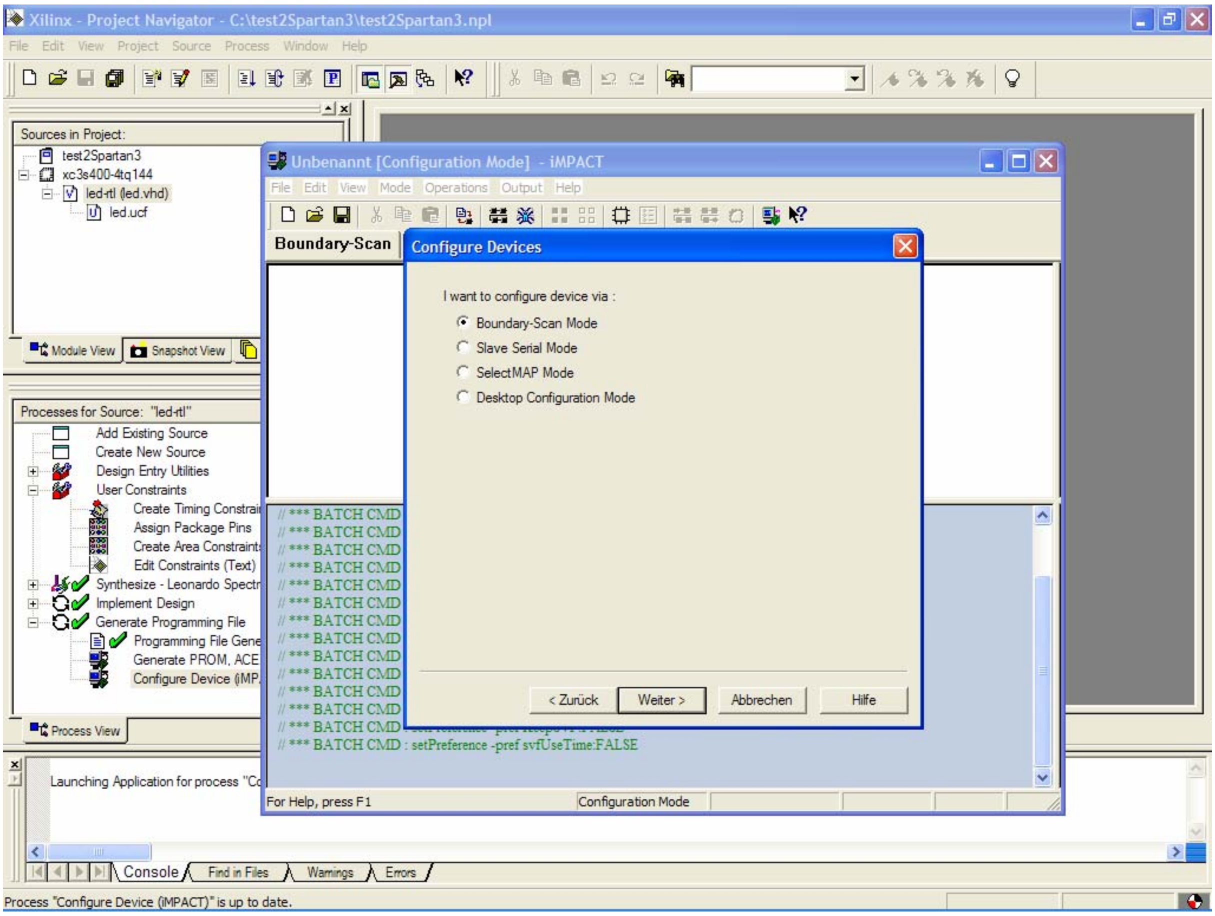


Figure 3

3. A new wizard turns up (Figure 4). Please select Boundary-Scan Mode and click the button “next”.



I want to configure device via :

● Boundary-Scan Mode
○ Slave Serial Mode
○ SelectMAP Mode
○ Desktop Configuration Mode

```
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD
// *** BATCH CMD : setPreference -pref svfUseTime:FALSE
```

Launching Application for process "C

Process "Configure Device (iMPACT)" is up to date.

Figure 4

4. Confirm all messages turning up.
5. After programming the FPGA the green DONE-LED on the FPGA board should be turned on and the Xilinx tool should give back the message “Programming Succeed”.

Why using a FPGA

Although more and more improved FPGA remain bigger and use still more energy than e.g. a microcontroller. But on the other hand they provide a lot of flexibility and speed that could be very interesting for algorithm that contain a lot of parallel processes and must be performed in real time. A comparison of some main features of FPGA and microcontrollers is given in table 2.

	FPGA	microcontroller
Pro	- Parallel processing, very fast if there are parallel tasks (like parallel storing data from a sensor and controlling the motor and controlling the Bluetooth) - Low power for applications with high processing intensity (good results with less clock frequency) - High flexibility for hardware tasks - High number of available pins - Contain place for several Soft-Processors so called MicroBlaze	- Good results for serial tasks - Low power consumption - Special power saving modes - Optimized for special tasks like serial floating point calculation
Con	- Low density of logic - For special calculation not as fast as optimized microcontroller - More power consumption as microcontroller	- Bad for parallel processing - No flexible hardware - Low number of available pins

Table 2 FPGA and microcontroller

The best choice for a special problem highly depends on the following questions:

- Could the problem be separated into parallel or does it consists of one serial task?
- Do microcontrollers exist that are optimized for the tasks?
- Does time play a role (throughput, real time)?
- What power consumption is acceptable?
- Is there enough develop and turn around time? It is much easier to debug software on a microcontroller then a FPGA.
- Is flexible hardware required?

To provide a platform for as many different tasks as possible I decided to include a FPGA in addition to the ARM on the Bluetooth board but also to give the users the chance to get rid of the FPGA board if the FPGA is not required. Users for whom the calculation power of the ARM on the Bluetooth board is not enough might load several soft processors called MicroBlaze into the FPGA or replace the FPGA board by a board with a microcontroller fitting to their purposes.

Why choosing the XC3S400-4TQ144C

Responsible for choosing this FPGA were its following features:

- The Spartan3 is very new and supposed to be available and to be supported for a long time.
- It is cheap in comparison to other FPGAs like from the Virtex family.
- It provides 16 blocks of internal block RAM with 294219 bits, digital clock manager and 16 dedicated hardware multipliers.
- Containing 400000 gates the XC3S400 meets most requirements for digital logic. E.g. a MicroBlaze loaded into the XC3S400 only needs 15% of the FPGA (please see Spartan3 datasheet [1]).
- The XC3S400-4TQ144C comes in the package TQ144 which is small enough to fit into the robot. It does not have a BGA but pins not producing the problems BGA bring into a design and manufacturing process.

For more details please see the Spartan3 datasheet [1].

Reconfiguration of the FPGA

The Spartan3 features 5 different configuration modes selectable with the configuration pins M0, M1 and M2:

- Slave Serial
- Slave Parallel
- Master Serial
- Master Parallel
- Boundary-Scan / JTAG

During the slave modes the CCLK pin of the FPGA acts as an input. The FPGA receives the configuration data only on the input pin DIN (serial mode) or parallel on the input pins D0 – D7 (parallel mode). Every new data bit is loaded into the FPGA with a rising edge on the CCLK pin.
When using the master modes the FPGA is providing a clock signal at the CCLK pin that is configured as an output. Like in the slave mode the FPGA expects the new data on pin DIN (serial mode) or on the pins D0 – D7 (parallel mode) when there is a rising edge at pin CCLK.
The boundary scan / JTAG mode make uses of the protocol described by IEEE 1532 / IEEE 1149.1. It is much more complex than the modes mentioned before.

The FPGA board inside the modular robot is designed for slave serial and JTAG configuration. The pins M0 - M2 are left unconnected. Because there are weak pull up resistors for these configuration pins inside the FPGA slave serial is the default mode. However, the JTAG interface is available independent from the voltage level at the pins M0 – M2.

I decided not to choose parallel modes because of the limited size of the FPGA board and robot where additional wires take unnecessary place.
The master mode is only available when using a PROM memory that is providing the data on the output using the clock signal on the FPGA pin CCLK. Such PROMs exists e.g. from XILINX but are quite expensive and can only be programmed by a JTAG interface requiring a complex programming protocol. In addition these PROMs are not usable for a lot of other applications.
Another possibility is using a microcontroller or CPLD implementing a protocol that converts the clock signal CCLK into some address signals that could be used to work with a normal FLASH memory. However, using a CPLD increases costs of the modular robot and requires additional space.
When a user in spite of this whishes to program the FPGA in master mode there is still the option using the ARM inside the ZV4002 as a protocol converter. Because the mode selection pins M0 - M2 are left unconnected a low level can easily added to the pins by small changes of the board or just by using some cables.

I chose to provide the serial mode because it is a quite powerful but still simple way of configuring the FPGA. It also takes profit from the ZV4002 and the FLASH memory that are already there for the Bluetooth avoiding additional space consummating and expensive devices. The FLASH memory chosen is a dual bank memory allowing reading data from one bank while erasing or programming the other one. For more information about the Flash memory on the Bluetooth board please read the Bluetooth Board Reference Guide [3]. The FPGA board provides access to all configuration pins that are necessary for slave serial configuration by pads that could be easily connected to GPIO pins of the ZV4002.

The FPGA board also supports JTAG mode because it serves as a powerful interface not only for configuration by also providing test methods. That is why the user may wish to have an easier plug and play connection mechanism to connect the FPGA directly e.g. to a PC. This demand is supported by a micromatch connector.

For more details about FPGA reconfiguration please see the Spartan3 datasheet [3].

Partial reconfiguration

Partial reconfiguration allows programming of only some parts of the FPGA instead of resetting and reprogramming the whole device. This could have several advantages because a bit stream with configuration data for only some part of the FPGA is much smaller than one for changing the whole device. This means:

- Saving place inside a memory for storing the bit files.
- Reducing time for programming the FPGA and sending the data e.g. via Bluetooth.
- Less power consumption for reconfiguration.

However, when designing a board for partial reconfiguration some extra attention must be taken because

- The reconfigurable part or module of the FPGA must have a height that is equivalent to the full height of the device.
- Horizontal placement of the reconfigurable parts must be on a four slice boundary.

The Spartan3 is placed on the FPGA board in a way so that the SRAM is parallel on one side of the FPGA. Arranging it on the top or bottom of the Spartan3 would mean that a MicroBlaze could not use it while the device is used for partial reconfiguration.

Please note that unmodified logic does not remain active during a partial reconfiguration process.

For more details about partial reconfiguration see appendix I and appendix II.

Pin Description

Table 3 gives an overview about the available pins.

Pin	FPGA pin	Connected to	Description
CLK	55	50MHz Oscillator	Clock signal for FPGA
RESET	92	Push button	Reset for FPGA
GND	GND	Micromatch-6 (X2)	Power supply; Ground
+3V3	VCCO	Micromatch-6 (X2)	Power supply; +3.3V
+1V2	VCCINT	Pad	Power supply; +1.2V
INIT_B	58	Pad	Configuration pad for Slave Serial Mode; 3.3V resistant
DIN	65	Pad	Configuration pad for Slave Serial Mode; 3.3V resistant
PROG_B	143	Pad	Configuration pad for Slave Serial Mode; 3.3V resistant
Done	71	Pad	Configuration pad; High when programming succeed; 3.3V resistant
CCLK	72	Pad	Configuration pad for Slave Serial Mode; 3.3V resistant
TDO_3V3	109	Pad	Configuration pad for Boundary Scan Mode (JTAG); 3.3V resistant
TCK_3V3	110	Pad	Configuration pad for Boundary Scan Mode (JTAG); 3.3V resistant
TMS_3V3	111	Pad	Configuration pad for Boundary Scan Mode (JTAG); 3.3V resistant
TDI_3V3	144	Pad	Configuration pad for Boundary Scan Mode (JTAG); 3.3V resistant
TCK_2V5	110	Micromatch-6 (X2)	Configuration pad for Boundary Scan Mode (JTAG); for 2.5V connector
TDO_2V5	109	Micromatch-6 (X2)	Configuration pad for Boundary Scan Mode (JTAG); for 2.5V connector
TMS_2V5	111	Micromatch-6 (X2)	Configuration pad for Boundary Scan Mode (JTAG); for 2.5V connector
TDI_2V5	144	Micromatch-6 (X2)	Configuration pad for Boundary Scan Mode (JTAG); for 2.5V connector
GPIO_104	104	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_105	105	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_107	107	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_108	108	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_128	128	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_129	129	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_131	131	Micromatch-8 (X3)	General purpose IO pin of FPGA
GPIO_130	130	Micromatch-8 (X3)	General purpose IO pin of FPGA

Table 3 Pin description

For more information about the available pins please also see the schematic of the FPGA board and the assembly in the appendix.

Connecting 2.5V configuration pins and ZV4002

To allow configuring the FPGA via JTAG and Bluetooth it was necessary to provide an interface to the 2.5V using configuration pins that is making them 3.3V tolerant.

This is achieved by adding serial resistors for the 2.5V using configuration pins that act as an input to reduce input current and applying pull up resistors for those acting as outputs for improving noise margin.

For more information about 3.3V tolerance of the configuration pins see page 40 of the Spartan3 datasheet [1].

SRAM

The FPGA board contains an additional external SRAM to run more powerful programs on a MicroBlaze. The K6R4016V1D-TC10 is a high speed SRAM from Samsung Electronics providing the following main features:

- Configuration: 256Kx16 bit
- Power supply 2.7-3.6V
- Small package: 44-TSOP2
  Using a BGA would result in an even smaller package but provide more difficulties with regard to designing and manufacturing the PCB.
- High speed of 10ns that allows running a MicroBlaze with the highest possible speed of 85MHz.

For more details please see the K6R4016 datasheet [2].

Ultra low dropout regulator with 2.5V output

The Spartan3 needs to work together with logic using 3.3V e.g. a SRAM or the ZV4002 three different voltages:

- For the inputs / outputs: 3.3V.
- For the core 1.2V.
- An auxiliary voltage of 2.5V for e.g. the JTAG.

To reduce density of devices and save place on the power board the linear regulator for providing the 2.5V is directly placed on the FPGA board. Only few applications use three different voltages. So arranging FPGA and 2.5V regulator together also supports the idea of modularity in terms of electronics. Replacing the FPGA board does not mean still using unnecessary place on the power board and producing costs for devices that are not needed.

Layout

To achieve a good performance of the FPGA and the SRAM there is a big ground plane on the bottom layer under the devices and a 3.3V plane on the top. All power supply inputs are bypassed by 100nF capacitors. Because of the distance to the power supply board there are 330µF tantalum capacitors both for the 3.3V and for 1.2V to improve the supply.

Bill of Materials

Symbol	Qty	Value	Manufacturer	Description	Package	Order Number	Price		Min pc.	Price [CHF]	Distributor
U1	1		Xilinx	Spartan3 | 400.000 gates	TQ144	XC3S400-4TQ144C	30,23	USD	10	38,01	Memec AG
U2	1		Samsung	SRAM | 256k x 16bit | 3.3V | 10ns	44-TSOP2	K6R4016V1D-TC10	5,05	CHF	10	5,05	WBC GmbH
U3	1	250mA, 2.5V	National Semiconductor	Ultra Low-Dropout Regulator	SOT23-5 (3mm x 3mm)	LP2992IM5-2.5	0,79	CHF		0,79	EBV Elektronik GmbH
		Alternatively you can replace the LP2992IM5-2.5 through the following device:									
U3	0	250mA, 2.5V	Texas Instruments	Ultra Low-Dropout Regulator	SOT23-5 (3mm x 3mm)	REG102NA-2.5					
X1	1		Jauch	Oscillator		O50.0-VX3MH	3,90	CHF	10	3,90	Dätwyler Electronics AG
X2	1		Tyco	Micromatch connector | 6 pins		128483	0,75	CHF	1	0,75	Distrelec
X3	1		Tyco	Micromatch connector | 8 pins		128484	0,95	CHF	1	0,95	Distrelec
S1	1		Diptronic	Push button | DTSJW-68S	SMD	210041	1,50	CHF	5	1,50	Distrelec
D1	1		Central Semiconductor	Schottky diode	SOD-323 (2.6mm x 1.35mm)	CMDSH-3	1,20	CHF		1,20	Computer Controls AG
C1, C5	2	330u		10% | 6.3V	D 7343(7.3mm x 4.3mm x 2.8mm)	812192	1,70	CHF	1	3,40	Distrelec
C2	1	10n	Murata	10% | X7R | 50V	SMD 0603 (1.6mm x 0.8mm)	830121	0,07	CHF	25	0,07	Distrelec
C3	1	4.7u	Murata	10% | X5R | 6.3V	SMD 0603 (1.6mm x 0.8mm)	823504	0,18	CHF	25	0,18	Distrelec
C4	1	1u	Murata	10% | X5R | 25V	SMD 0603 (1.6mm x 0.8mm)	823495	0,20	CHF	25	0,20	Distrelec
C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23, C24, C25, C26, C27	22	100n	AVX	+80/-20% | Y5V | 16V	SMD 0402 (1.0mm x 0.5mm)	316581	0,07	CHF	10	1,54	Farnell
R2	1	56R	MULTICOMP	5% | 0.063W | 50V	SMD 0603 (1.6mm x 0.8mm)	612339	0,05	CHF	10	0,05	Farnell
R10	1	82R	MULTICOMP	5% | 0.063W | 50V	SMD 0603 (1.6mm x 0.8mm)	612352	0,05	CHF	10	0,05	Farnell
R6, R8	2	10k	MULTICOMP	1% | 0.063W | 50V	SMD 0603 (1.6mm x 0.8mm)	911355	0,05	CHF	50	0,10	Farnell
R1, R3, R4, R5, R7, R9	6	100R	MULTICOMP	5% | 0.063W | 50V	SMD 0603 (1.6mm x 0.8mm)	612364	0,05	CHF	10	0,30	Farnell

LED2	1	Red	Agilent Technologies	LED Red	SMD 0603 (1.6mm x 0.8mm)	250161	0,35	CHF	5	0,35	Distrelec
LED1	1	Green	Agilent Technologies	LED Green	SMD 0603 (1.6mm x 0.8mm)	250164	0,35	CHF	5	0,35	Distrelec
									total:	58,74	

Appendix I – Partial Reconfigurability Frequently Asked Questions

The following text is taken from http://www.xilinx.com/ise/advanced/partial_reconf_faq.htm.

How do I obtain Partial Reconfigurability?

Partial Reconfigurability is built into the ISE product, and requires the Modular Design flow which is also part of ISE. The two flows have characteristics that are substantially similar, and rather than create a whole new flow and set of constraints for Partial Reconfigurability, it was natural to leverage Modular Design's capabilities.

Where do I learn about all the details, the "nuts and bolts", of how to do a Partial Reconfiguration design?

The best place to start is the Partial Reconfiguration Flow Application Note. XAPP290.

Does Partial Reconfigurability really allow me to have part of my FPGA running, while another part is being reconfigured?

Yes it does. The logic, storage elements, interconnect, and I/O of the sections not being reconfigured will all stay active and operational in the system.

How many reconfigurable sections can there be for a design? How many different designs can be created and configured into each section?

Reconfigurable sections must be on column boundaries. Once these boundaries are created, there is no limit the number of different reconfigurable designs that can be created and configured for that section. Designs may also have more than one reconfigurable section in a design, but for simplicity's sake, we recommend only one.

What changes to I need to make to a design to allow for Partial Reconfigurability?

1. The design must be partitioned into modular portions. The fixed part is one (or more) modules, and each reconfigurable portion must be unique modules. Port placement, if you're familiar with Modular Design, is not required as this is superseded by the Bus Macro approach (see details below).
2. Bus Macros will need to be instantiated into the HDL design to accomplish the communication signals to and from reconfigurable blocks. One macro is needed for each four signals.
3. Clocks, because they are part of their own, unique bitstream frames, must be designed at the top-level, so that all clocks that will ever be used in any configuration of the FPGA are present.
4. Area constraints must be provided, constraining the reconfigurable blocks that span the full vertical dimension of the chip (i.e. full columns). The bus-macros will need to be floorplanned into their own specific areas, straddling the module boundaries. The Floorplanner will write out new "route-area" constraints corresponding to the area-groups, ensuring that the routing conforms to the rules of reconfigurable designs.
5. Xilinx is STRONGLY recommending that the earliest designs using this flow keep things as simple as possible. This means that one fixed portion and one reconfigurable portion should be the general rule.

What's the basic design flow for Partial Reconfigurability?

1. Design communication between modules on TBUF bus macros.
2. Create clock template for the design, using HDL template provided by Xilinx.
3. Set area/routing constraints using the Floorplanner.
4. Implement...
   - a) each fixed module (using "active module implementation phase" of Modular Design).
   - b) each reconfigurable module (using "active module implementation phase" again).
   - c) Implement a full design (using assembly phase) to create "full" design implementation that will be the initial power-up state of the design.
5. Create bitstreams...
   - a) for reconfigurable sections.
   - b) for full design for initial power up configuration.

Why is a TBUF bus required to allow inter-module communication?

Partial configuration demands that resources used by a reconfigurable module cannot exist out of the bitstream frame boundary. Moreover, the routing that is used to connect signals crossing reconfigurable module boundaries cannot change when a module is reconfigured. There are no mechanisms in the software to allow the user to specify a hard (bitstream frame) boundary, and creating such mechanisms is well beyond the scope of Emerald. The TBUF bus is the simplest mechanism we can provide to give absolutely predictable control the use of routing resources of signals that cross such boundaries.

Exactly, how does one use a TBUF bus to allow signals to cross module boundaries?

Xilinx will provide a "Bus Macro" NMC file that fixes TBUF placement and the exact routing of the longlines. Up to four bits of data can be shared per macro, and the user will need to instantiate as many of these macros as is needed to cover all the cross-module signals in his design.

Creating the Clock Template is a process that seems unclear, how does one go about this?

The clocks in a reconfigurable design have their own unique bitstream frames - separate from the either the reconfigurable or fixed logic portions of the design. Xilinx will provide HDL coding examples and/or templates that a user should use to create the clocks in their design.

What checking does the software provide to make sure the design is correct?

1. Functional design correctness, as always, should be verified by the designer using simulation, timing analysis and other traditional techniques.

Bitstream compatibility and correctness: If all the rules are adhered to by the user so that the fixed and reconfigurable portions of the design are completely constrained to their respective regions, and the Bus Macro is used properly such that it is the only means of communication between reconfigurable modules, then the user can be sure that the bitstreams created will be correct by construction. If these rules are not followed, the software has no means to check that bitstreams are incompatible and if they are in fact incompatible, may actually result in device damage. The one main rule that needs to be followed is that a reconfigurable module must not use resources outside of its column boundaries. No routing, no IOBs, no logic, belonging to a reconfigurable module may reside outside of the column boundaries defined for that module. Though the flow and software will guide a user towards following these rules, there are few explicit restrictions is place to check for and prevent one from willfully violating these rules.

Appendix II – Partial Reconfiguration in a Spartan3

The following text is taken from http://www.fpga-faq.com/archives/60625.html

---

"Steven K. Knapp" <steve.knappNO#SPAM@xilinx.com> wrote in message
news:<XFb9b.2983$n22.624510994@twister1.starband.net>...

ICAP, or the Internal Configuration Access Port, is not supported in Spartan-3 FPGAs. Glimpses of the ICAP interface appear in various tools, either because it was too difficult to remove this function from the software, or the software mistakenly assumed that Spartan-3 had ICAP.

Dynamic reconfiguration is still supported in Spartan-3 via the external SelectMAP interface or JTAG, just not through the ICAP interface. The decision to remove it was due to silicon resource requirements and testing cost. Although dynamic reconfiguration is a powerful concept, few consumer-oriented applications use it.

The following is some background on partial reconfiguration in Spartan-3 and the ICAP primitive.

Does Spartan-3 Support Partial Reconfiguration?

Virtex/E, Virtex-II, and Virtex-II Pro devices - generically called Virtex throughout this article - support a feature called partial reconfiguration.

Using this feature, an application can modify a portion of the bitstream programming inside an FPGA to change the FPGA's functionality. Spartan-3 FPGAs support some of these same capabilities, but with limitations compared to Virtex. Via today's design software, partial bitstream changes must be performed on an entire IOB, CLB, or Block RAM column basis in both Virtex and Spartan-3 FPGAs. For example, to change a single bit within a single LUT, the application must update all the CLBs in the affected column. Any unmodified CLBs within the column are overwritten with the same configuration data.

Perhaps the most important difference between Virtex and Spartan-3 FPGAs is how the FPGA logic behaves during the reconfiguration process. In the Virtex devices, any unmodified bits in the affected column continue to operate normally. Consequently, if bits within a column are unchanged, then the surrounding logic continues to function normally. In Spartan-3 FPGAs, however, even unmodified bits in a column are temporarily reset during the reconfiguration process, which greatly complicates using partial reconfiguration. Partial reconfiguration works in Spartan-3 FPGAs, just with extra complications.

A column consists of multiple configuration frames. Physically, the Virtex hardware supports configuration changes at the frame level, but software currently just supports changes at the column level. The Spartan-3 hardware supports bitstream changes at the column level only.

The application can partially reconfigure the FPGA via a variety of means, including the parallel SelectMap configuration interface and the FPGA's JTAG port. Virtex-II and Virtex-II Pro families also support another means called the ICAP (Internal Configuration Access Port). The ICAP interface is similar to the parallel SelectMAP interface, but is available from within the FPGA. Although the Spartan-3 architecture is based on the Virtex-II and Virtex-II Pro architectures, the Spartan-3 family does not support the ICAP interface.

Table 1 summarizes how partial reconfiguration compares between families.

Table 1. Partial Reconfiguration Support in Virtex-II vs. Spartan-3.

Software supports...
Virtex: Column-based reconfiguration
Spartan-3: Column-based reconfiguration

Hardware supports...
Virtex: Frame-based reconfiguration
Spartan-3: Column-based reconfiguration

Unmodified logic remains active during reconfiguration?
Virtex: Yes
Spartan-3: No

Reconfigure via SelectMAP?
Virtex: Yes
Spartan-3: Yes

Reconfigure via JTAG?
Virtex: Yes
Spartan-3: Yes

Reconfigure via ICAP?
Virtex: Virtex-II and Virtex-II Pro only
Spartan-3: No

For more information on partial reconfiguration, visit the following web links:

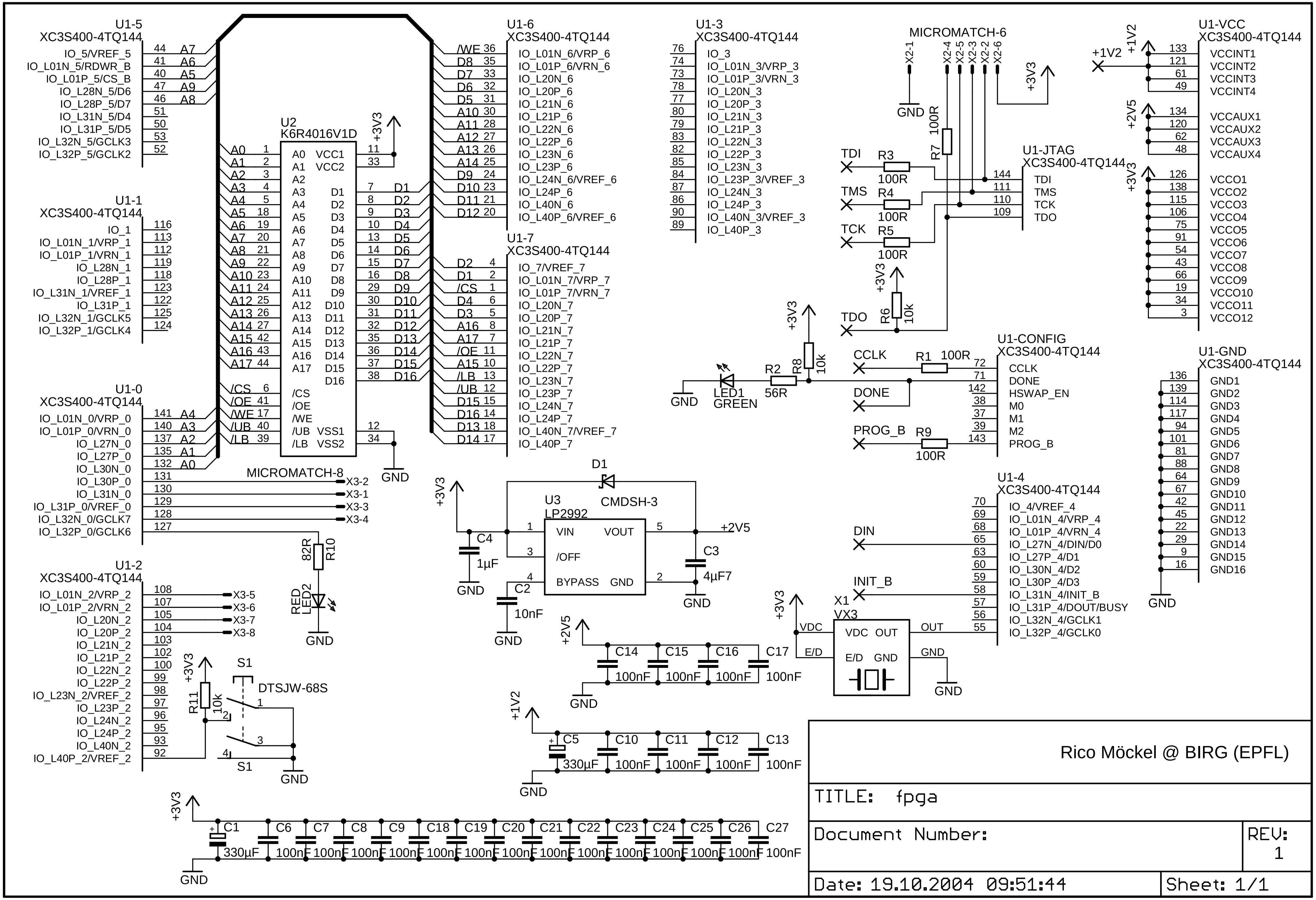
Partial Reconfigurability Frequently Asked Questions
http://www.xilinx.com/ise/advanced/partial_reconf_faq.htm

XAPP151: Virtex Series Configuration Architecture User Guide
http://support.xilinx.com/xapp/xapp151.pdf

XAPP290: Two Flows for Partial Reconfiguration: Module Based or Small Bit Manipulations

http://www.xilinx.com/xapp/xapp290.pdf

Steven K. Knapp
Applications Manager, Xilinx Inc.
Spartan-3/II/IIE FPGAs
http://www.xilinx.com/spartan3



Rico Möckel @ BIRG (EPFL)

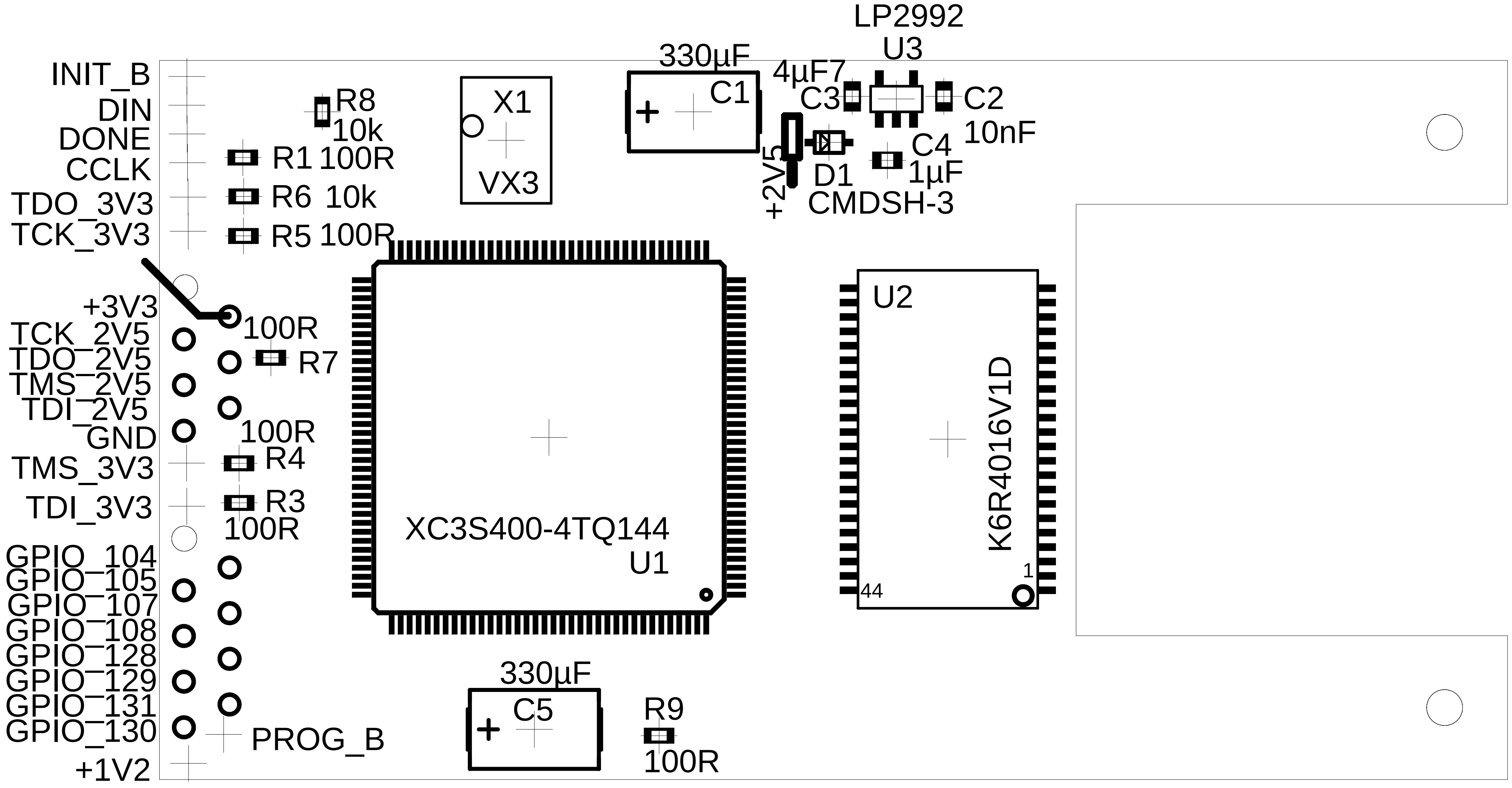
TITLE: fpga

Document Number:

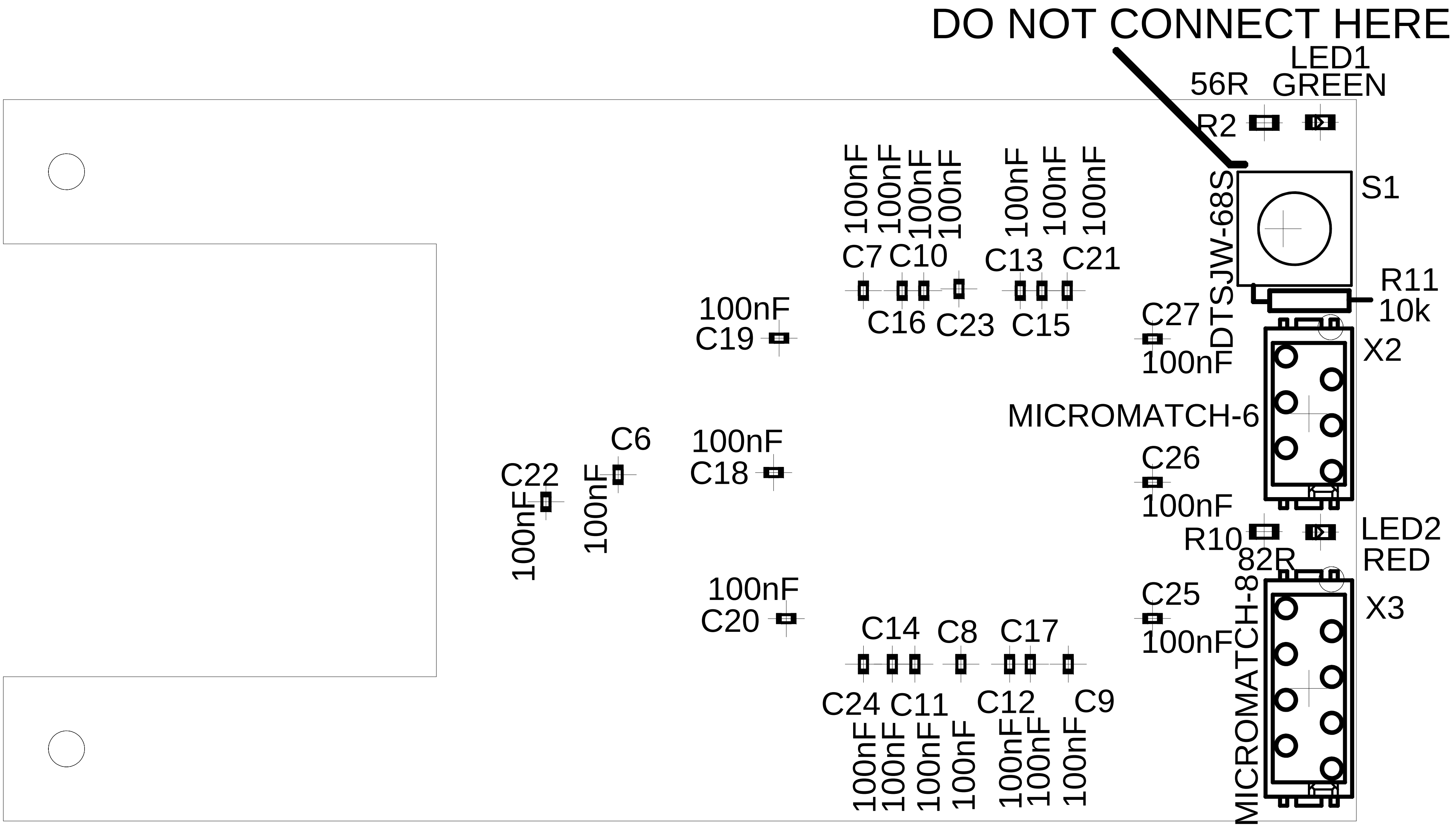
REV: 1

Date: 19.10.2004 09:51:44

Sheet: 1/1



Rico Möckel @ BIRG (EPFL)	
FPGA board	Rev 1
Assembly top	



Rico Möckel @ BIRG (EPFL)	
FPGA Board	Rev 1
Assembly bottom	