Universiteit Utrecht

Scientific Instrumentation

ITS upgrade
RUv2 testprocedure

Project	ITS upgrade (W0006577)	**Author(s)**	M.J. Rossewij
		Date	31-October-2018
Name	RUv2 testprocedure	**ID**	W0006577_001
		Status	Concept
Filename	RUv2Test Procedure1v5.docx	**Version**	1v5
Printed	14-5-2019 17:02		

University of Utrecht, Faculty of Science
Scientific Instrumentation
P.O.Box 80004, 3508 TA Utrecht
Tel: 030 2532293, Email: WWW: http://www.science.uu.nl/instrumentatie/

Document History

Version	Date	Author	Section	Description
1.2	18-March-2019	M.J. Rossewij	-	Document updated for NIKHEF testsystem

Contents

Introduction

The RUv2_1 test procedure consists of the following steps:

1) Probe bed test system
    a. Impedance check for shorts
    b. Ramp up and measurement of power rail voltages and currents
    c. Program PA3
    d. Program FX3 I2C PROMs
2) Loopback tests
    a. Upload firmware to US
    b. Start long-loopback test
    c. Start short-loopbacktest
    d. Read SCA ADC values and test GPIO & I2C to xilinx
    e. Check dipswitch, pushbuttons & JitterCleaner Clock
3) Program/Fuse GBTx
4) Test Selectmap & Flash interface and read bad flash-blocks
5) Check CAN interface

1 Probe bed test system

Testing the board with the probe bed test system requires following steps:

1) Place 3 jumpers: J33, J30 & J31 on pin1-2
2) Set S9 switch2 to + (S9= 0+000000)
3) Place board in probe bed
4) Connect blue USB3 cable
5) Close lid
6) Start ProbeBedTest.vi (if not already running)
    a. Windows start button->NI Labview 2018 (64 bit)->ProbeBedTest.vi
    b. Press run arrow (upper left corner): Relay module, Power Supply & Keysight must be ready
7) Start measurement procedure pressing right start button
8) Provide (4-digit) serial number (using the scanner)
9) The probe bed test system subsequently does the following actions:
    a. Impedance check for shorts
    b. Ramp up and measurement & check of power rail voltages and currents
    c. Program PA3
        i. User checks verify ok and closes DOS command window
    d. Ask user to Program FX3 I2C PROMs
        i. Start (if not already run) USB (windows start button->) control center & move up left
        ii. Select "Cypress FX3 USB Bootloader Device"
        iii. USB Control Center Taskbar->Program->FX3->I2C EEPROM
        iv. Select file "slfifo_uart_i2c_SN0_20160712.img"
        v. Wait until Control Center states "Programming of I2C EEPROM Succeeded"
        vi. Press "Programming finished" in Labview ProgramCypressFX3.vi window => Ready!

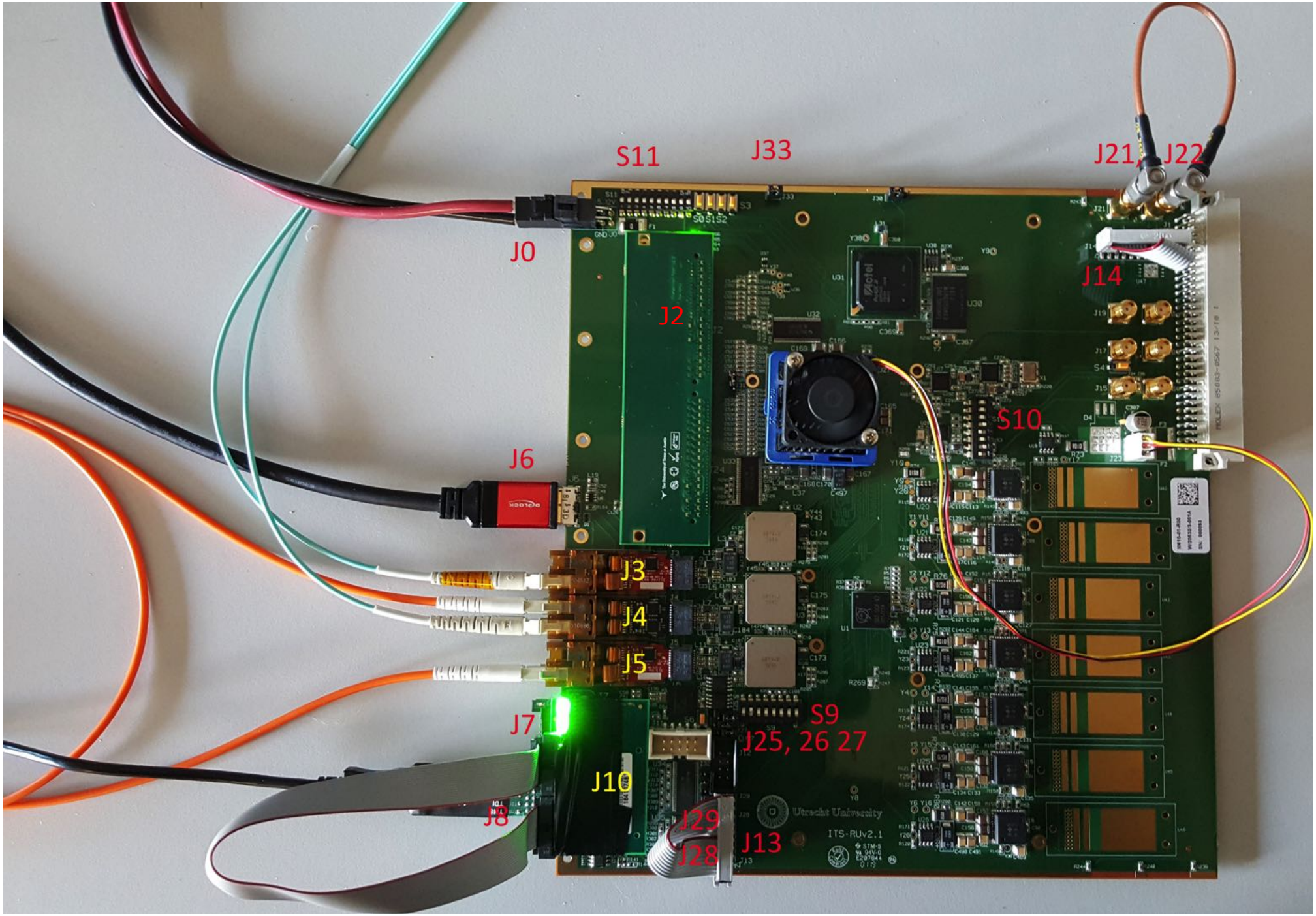
2 Configure FPGAs (Xilinx-US & Microsemi-PA3) + loopbacktest

Place 5 jumpers:

- J25 2-3,
- J26 2-3,
- J27 2-3,
- J28 on 1-2,
- J29 on 2-3

Dipswitch S11 all to OFF (not ON) "0000000000"
Dipswitch S10 to "+-0+-000"
Place FAN
Place both VTRx, VTTx on J3-5 & connect fiber CRU->VTRx0-Rx, VTTxUp->VTRx2, VTTxDown->CRU
Connect Power cable (5-12V) to J0
Connect USB connector J6
Place all loopback cables:

- J7 Aux/BUSY plug (connects 3↔4, 5↔6, 9↔10, 11↔12, 15↔16, 17↔18)
- J10 PU mezzanine with loopbackcable
- J2&J24 transistion loopback-board
- J13 header (connects 1↔11, 3↔13, 5↔15, 7↔17, 9↔19)
- J14 header (connects 1↔11, 3↔13, 5↔15, 7↔17, 9↔19)
- SMA J21-J22


S11
J33
J21, J22
J0
J14
J2
S10
J6
J3
J4
J5
J7
S9
J25, 26 27
J10
J8
J29
J28
J13
Utrecht University
ITS-RUv2.1

2.1 Configure Xilinx

Plug Xilinx/Digilent download-cable in J8:
Switch on power supply (5,4V, 1,4A)
Start Vivado2018.3-> open project “TestSystem”->hw. manager->open target (auto connect) ->Program device
Current on PS increases to 2,5…3,1A
INIT LED (D2) green->green
DONE LED (D3) red-> green
Check if all 6 LEDs are blinking.

2.2 testAll.py: All loopbacks & GBT long loop test

Checks all loopback-cables/boards and the long-GBT-loopback (CRU->VTRx0-Rx, VTTxUp->VTRx2, VTTxDown->CRU)

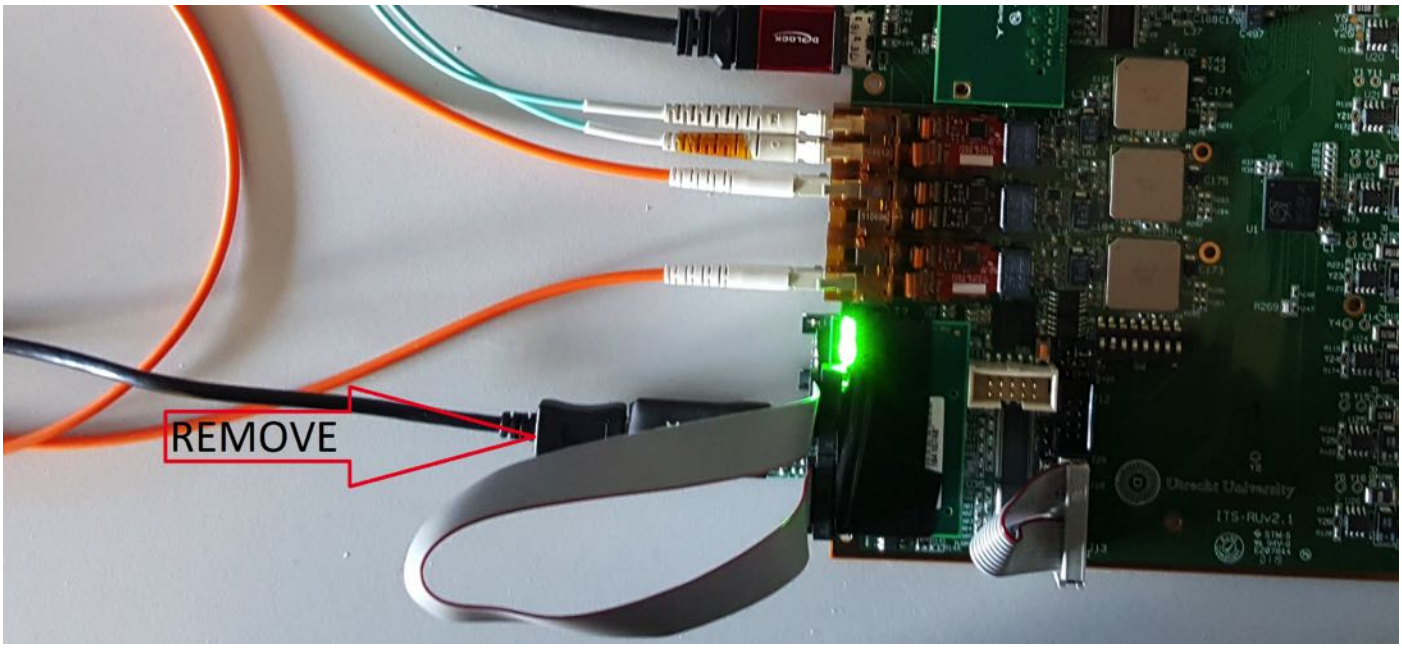
- cd /home/its/git/TestSystem/software/py/
- ./testAll.py
- Give boardID using the scanner
- Test runs first for 5 sec to check all cable are ok.
    - If not ok, break program (CTRL-C), write comment, fix problem and restart.
- Wait 5 minutes to complete the test
- Check log for errors

When there was a GBT error, the RUv0_CRU red LEDs will be on. Number of errors can be read with:
./testbench.py cru get_data_error_counter
./testbench.py cru get_link_error_counter_dis
./testbench.py cru get_fec_counter
In addition, the RUv0_CRU ILA can be used to check whether the PRBS or CBT pattern is present.

2.3 testSCA.py: adc, GPIO, I2C->XCKU & JTAG lines

Test the I2C lines to US, SCA GPIO lines to PA3 & US, reads SCA ADC values and checks if they are within range. It also reads Xilinx JTAG ID. It also implicitly checks the GBTx0<-> SCA interface (EC).

- **Change fibers** (power supply can remain on)
- Remove JTAG cables (J8 Xilinx/digilent cable and if connected J11 PA3 flashpro cable)
- ./testSCA.py
- Give boardID using the scanner
- Check log for errors



2.4 testGBT.py: GBT short loop test

Checks the short GBT loopback only (CRU->VTRx0-Rx, VTRx0-Tx->CRU):

- fibers as above in testSCA.py
- cd /home/its/git/TestSystem/software/py/
- ./testGBT.py
- Give boardID using the scanner
- Wait 200 seconds to complete the test
- Check log for errors

2.5 DIPSwitch, Pushbutton

Test DIPswitch, PUSHbutton, LEDs connected to PA3 & US. Also compares GBTx->JitterCleaner Clock to local Clock.

- Dipswitch S11-1 to '1', the rest to '0'
- Dipswitch S10: "+-0+-0xx"
- ./testBUTTON.py
- Give boardID using the scanner
- Switch sub-sequentially the other dipswitches to '1'.
- Then press sub-sequentially the pushbuttons.

If procedure fails due to button bounce, retry. Note if there is a structural problem.

2.6 LED PushButton -> LED test

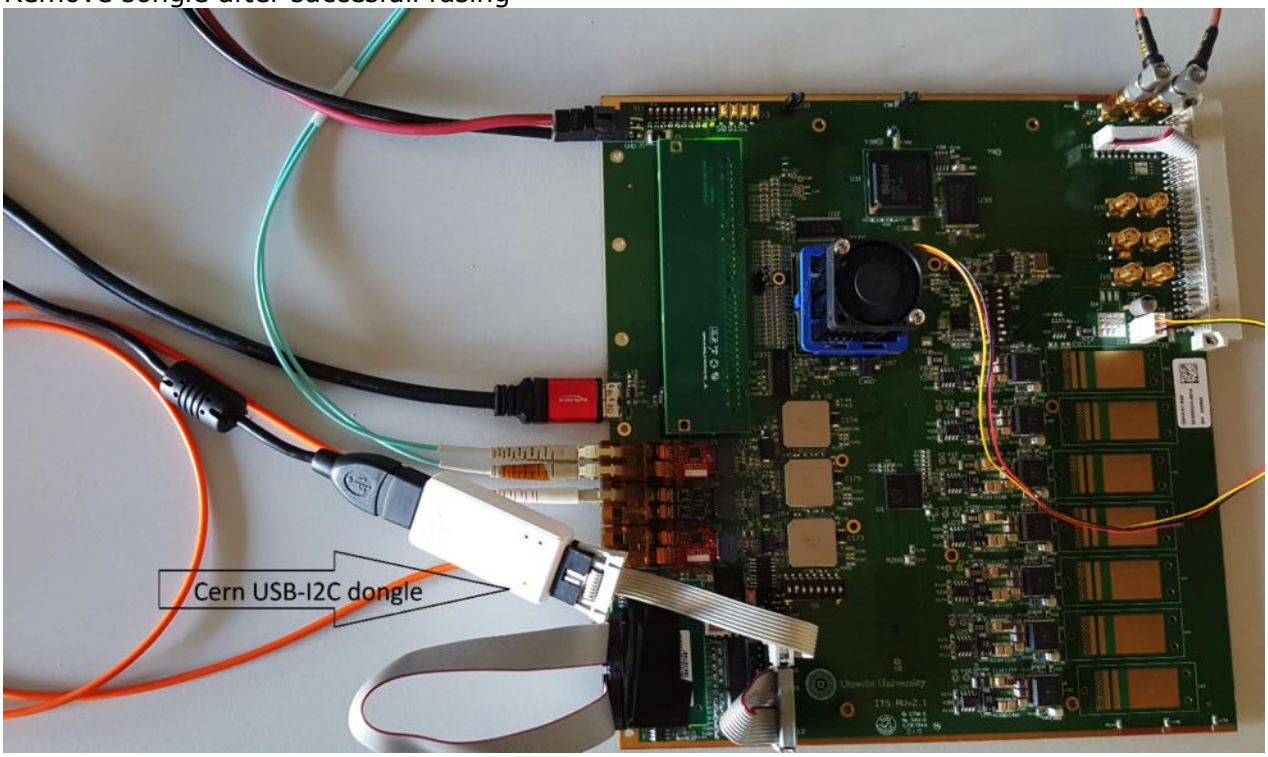
Subsequently the 4 Push buttons and check if LEDs go on.

- Press S0 => 2 LEDs on
- Press S1 => 2 LEDs on
- Press S2 => 1 LED on
- Press S3 => 1 LED on

3 Check & Fuse GBTx0 using USB-I2C dongle

Detects presence all 3 GBTx (on I2C address 0c1, 0x3 and 0x5) and fuse GBTx0.

- Connect USB-I2C dongle to J12
- fibers as above in testSCA.py
- cd /home/its/git/TestSystem/software/py/
- ./fuseGBTx.sh
    - Give its password (written on PC, all small letters)
    - Checks if all 3 GBTx (0x1, 0x3 & 0x5) are present
    - Fuses GBTx0 (0x1) (Check if FUSE LED blinks red)
- Remove songle after succesfull fusing



4 PA3: SCA-I2C, US selectmap-ID, CAN & Flash bad blocks

Before executing the scripts below, first update the PA3 firmware with.

- Connect FlasPro4 cable to J11
- Open FlashPro from windows PC desktop
- Open project C:_GIT\...\TOP_fp\TOP.pro
- Press run button

4.1 Test I2C to PA3

Tests the SCA I2C links to the PA3

./initPA3I2C.sh

./testPA3I2C.py

4.2 Read Selectmap register

In this test, the PA3 reads from the Xilinx US select map interface the selectmap ID (=13919093) and sends it to the PC where it is checked to be ok.

./testSelectMap.py

4.3 TestCAN

Set (probably already ok) dipswitch S11 CAN address to 0xFF (i.e. S11 switch 3 to 10 to '1')

- ./initCAN.sh (only needed the first time)
- Connect CAN cable to J9
- ./testCAN.py
- Check after 30 seconds for 0 failures and all successful reads and writes

4.4 Write/Read to Samsung Flash page0

This test is intended to verify the PA3 Samsung Flash interface: First the PA3 writes some random numbers to the Flash page 0. Then it is read back and checked to be ok.

./testFlashReadback.py

4.5 Write/Read to Samsung Flash bad blocks

Only if the former tests (4.4: Write/Read to Samsung Flash page0) was ok, do the test below:
Read the bad-block locations from memory and stores it in the local database.

./readFlashBadBlocks.sh

The database can be accessed with sqlite or sqlitebrowser

This test takes 22 minutes. Tester could prepare next board and perform probe bed test in the mean time.

5 Final

5.1 Packing the Readout Unit

- Switch off power supply channel
- Remove all cable and boards, except the VTRx on J3 and the VTTx on J4
- Place dustcaps back on VTRx and VTTx
- Pack board in ESC sack and box
- Add traveler (log-paper) to stack

5.2 Storing on EOS / lxplus

The log-files on the Linux-ubuntu machine should be copied at the end of the day with:

- cd /home/its/git/TestSystem/software/py/results
- ./copyLog.sh

The Bad block database should be copied at the end of the day with:

- Check if databasefile is reasonable size
- cd /home/its/git/TestSystem/software/py/Magnus
- ./copyDB.sh

The probe-bed log files on the windows-PC should be copied at the end of the day with:

- Cmd
- copyLog.bat

Scan paper logs and send them also to lxplus

scp scans.pdf cernUserID@lxplus.cern.ch:/eos/project/i/itslogs/RU/travelers/

Appendix A Requirements

Appendix A.1 Hardware

The table below shows the hardware needed for the different tests:

General			
1	Ubuntu (16.04) PC with 2*USB3		
1	Optional: Windows PC with 2*USB 3		
1	Xilinx or digilent HS2/3 download cable	J8	
1	Microsemi Flash pro4/5 cable (flash-pro5 required for Linux)	J11	
1	USB3 A-microB	J6	
1	FAN for cooling RUv2	J23	
Power-on test			
1	Multimeter		
1	PSU (5-12V, >3A)		
1	Power cable	J0	
1	Optional: Probe station		
Loopbacktests			
1	Optional: BAR code scanner (KE-5500 at electronixs.nl or bol.com)		
1	Loopbackplug	J7	
1	Power mezzanine with loopbackcable	J10	
2	Pinheader loopback cable	J13 & J14	
1	SMA cable	J21 & J22	
1	Transistion loopbackboard	J2 & J24	
Programming GBTx			
1	USB-I2C dongle	J12	
GBT link tests			
1	CRU (emulator + cable USB3 A to B)		
1	Optical fibre		
1	VTRx	J3	
CAN interface test			
1	PCAN-USB IPEH-002021 (PCAN-USB IPEH-002021)	J9	

Appendix A.2 Software

The following software is needed:

General			
1	Linux (tested with Ubuntu 16.04)		
1	Optional Windows		
1	Vivado 2018.3 (preferably full version or at least the lab edition), preferable under linux		
1	Microsemi libero 11.8 or 11.9 or at least the FlashPro sw.		
1	Windows: Cypress USB control center Linux: cyusb_linux		
1	Gbtx downloadcable sw: programmerv2.20180725.jar		
1	Sw for flashing GBTx		
1	Reading the bad blocks		
1	Optional: Probe station software		