SALT128 Wafer Screening

Internal Note

Issue: 1
Revision: 0

Reference: LHCb-XX-2017
Created: February 8, 2017
Last modified: July 19, 2017

Prepared by: Christopher Betancourt, Iaroslava Bezshyiko [a]
[a]University of Zurich, Switzerland

Abstract

This is a description of wafer probe tests for the SALT128 mass production quality assurance. The mechanical and electrical layout of the chip as relevant for the tests, types of tests to be carried out, information on the wafer probe card and a timeline is provided.

Contents

List of Figures

List of Tables

1 Introduction

SALT (**S**ilicon **A**SIC for **L**HCb **T**racking) is a 128 channel ASIC to be used as a readout chip for silicon sensors in the UT of LHCb experiment. Quality assurance tests need to be carried out after mass production of the chip, insuring high yield and proper functionality before installation into the experiment. The large volume of chips ($\sim$ 5k) requires tests to be carried out in an automatic fashion, with each chip classified into three catagories; GOOD (full functionality of every channel), OK (at most one bad channel) and BAD (two or more bad channels). The chips will be tested on wafer before dicing.

Tests will be carried out with a semi-automatic probe station housed in a clean room located at CERN. The probe station is equipped with special imaging allowing for proper alignment and planarity. A wafer probe card will be used to make electrical contact with the chip. The probe card will be connected to a DAQ system controlled by a PC which also controls power supplies and movement of the probe station chuck, as illustrated in figure 1.

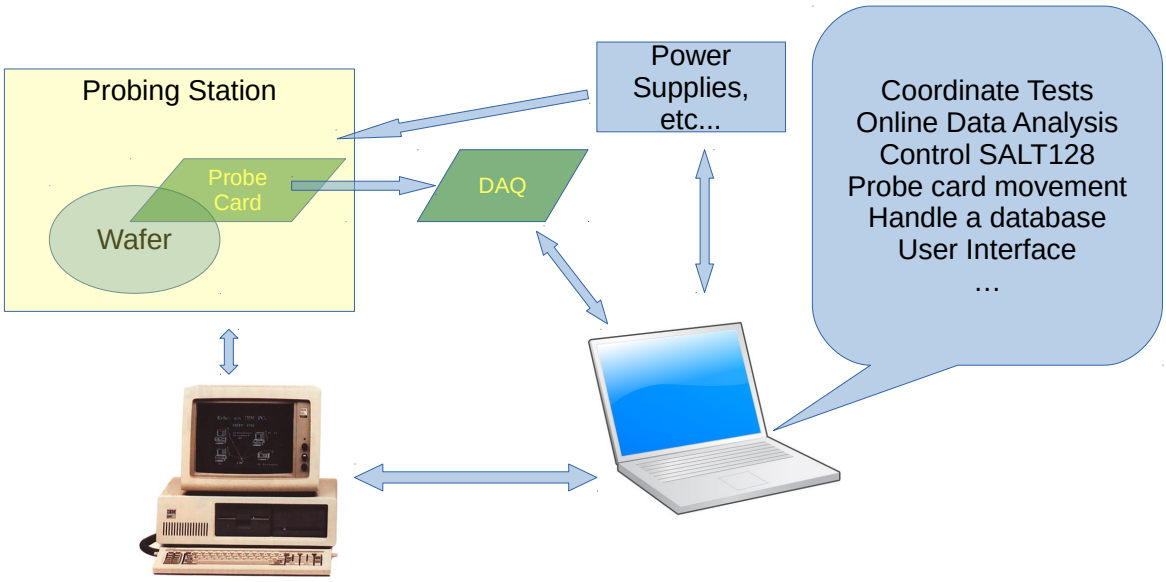

Figure 1 Illustration of the measurement set up.

2 SALT128 layout

SALT128 will contain a total of 246 pads: 128 input pads which will not be probed during the wafer probe tests and will be bonded to hybrids after screening, 4 ground pads, two on each side of the 128 input pads, 78 pads on the opposite to the input pads which will be tested during wafer screening and also subsequently bonded to hybrid, and two sets of 18 pads which will only be used during wafer screening. A layout of the chip indicating the pad locations and their pin numbers is shown in figure 2. A list of pads to be probed during wafer screening is indicated in table 1.

Pin	Name	Probed?
1	VSSAI	?
2	VSSAI	?
131	VSSAI	?
132	VSSAI	?
133	PRE_BUF1	weak no
134	SH1_BUF1	weak no
135	SH2_BUF1	weak no
136	SH_BUF1	weak no
137	AP_BUF1	weak no
138	AN_BUF1	weak no
139	I_BUF1	weak no
140	VSSPSTD_BUF1	weak no
141	VSSA_BUF1	YES
142	VDDA_BUF1	YES
143	ADC_INP	weak no
144	ADC_INN	weak no
145	V_S2D_DAC	YES
146	V_PULSE_DAC	YES
147	V_SLVS_VBIAS_DAC	YES
148	V_SLVS_INREF_DAC	YES
149	SCAN_ENABLE	YES
150	SDI	YES
151	SDO	weak no
152	TEST_MODE	YES

153	VDD	YES
154	VSS	YES
155	VDD	YES
156	VSS	YES
157	VDDPST	YES
158	ID0	YES
159	ID1	YES
160	ID2	YES
161	I2C_SCL	YES
162	I2C_SDA	YES
163	RST_N	YES
164	MAIN_CLK_N	YES
165	MAIN_CLK_P	YES
166	VSSPST	YES
167	VDD	YES
168	VSS	YES
169	VDD	YES
170	VSS	YES
171	VDDAPST	YES
172	VSSAPST	YES
173	VDDADC	YES
174	VSSADC	YES
175	VDDADC	YES
176	VSSADC	YES
177	VREFD	YES
178	VSSADC	YES
179	VREFD	YES
180	VSSADC	YES
181	VDDADC	YES
182	VSSADC	YES
183	VDDADC	YES
184	VSSVCM	YES
185	VDDVCM	YES
186	VSSVCM	YES
187	VDDVCM	YES
188	VSSAPST_IN	YES
189	VDDAPST_IN	YES
190	VSSAPST_IN	YES
191	VDDAPST_IN	YES
192	VSSAI	YES
193	VDDA	YES
194	VSSAI	YES
195	VDDA	YES
196	VSSAI	YES
197	VDDA	YES
198	VSSA	YES
199	VDDA	YES
200	VSSA	YES
201	VDDA	YES
202	VSSA	YES
203	VDDA	YES
204	VSS	YES
205	VDD	YES
206	VSS	YES

207	VDD	YES
208	VSSPST	YES
209	DDR_TFC_N	YES
210	DDR_TFC_P	YES
211	DDR_OUT0_N	YES
212	DDR_OUT0_P	YES
213	DDR_OUT1_N	YES
214	DDR_OUT1_P	YES
215	DDR_OUT2_N	YES
216	DDR_OUT2_P	YES
217	DDR_OUT3_N	YES
218	DDR_OUT3_P	YES
219	DDR_OUT4_N	YES
220	DDR_OUT4_P	YES
221	DDR_OUT5_N	YES
222	DDR_OUT5_P	YES
223	VDDPST	YES
224	VSS	YES
225	VDD	YES
226	VSS	YES
227	VDD	YES
228	DATA_CLK_OUT_N	YES
229	DATA_CLK_OUT_P	YES
230	V_SH_DAC	YES
231	V_KRUM_DAC	YES
232	V_PRE_DAC	YES
233	VMCD	YES
234	VCMC	YES
235	VCMB	YES
236	VCMA	YES
237	VDDA_BUF0	weak no
238	VSSA_BUF0	weak no
239	VSSPSTD_BUF0	weak no
240	I_BUF0	weak no
241	AN_BUF0	weak no
242	AP_BUF0	weak no
243	SH_BUF0	weak no
244	SH2_BUF0	weak no
245	SH1_BUF0	weak no
246	PRE_BUF0	weak no

Table 1: List of pads to be tested during wafer screening.

3 Probe station, wafer probe card and DAQ

All wafer probe tests will make use of a semi-automatic probe station located in a clean room at CERN. A photo of the set up is shown in figure 4.

4 Functionality

The SALT ASIC, containing 128 channels and an additional two channels for testing and monitoring, is broken into several components:

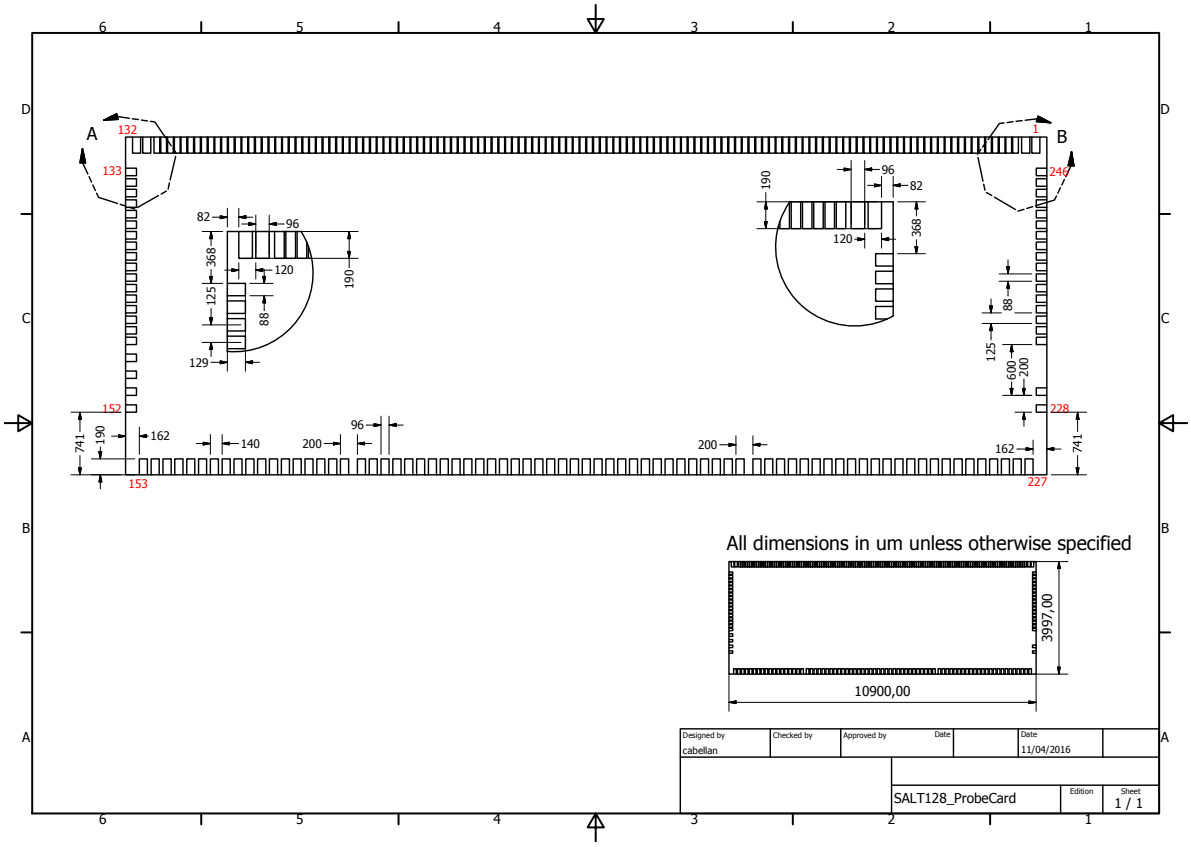

Figure 2 Layout of the pads on SALT128.

- **Analogue front-end** - Each channel on the front-end consists of a preamplifier, a shaper and a signal-to-differential converter.
- **Analog to Digital Converter** - Each channel contains a 6-bit ADC. An additional six ADCs a present for monitoring.
- **DACs** - Seven DACs used to bias different blocks of the SALT chip. Additionally, each channel contains a 7-bit DAC to set the front-end baseline voltage.
- **Clock generation** - 40 MHz external clock is used in the digital part of the chip. A DLL is used to shift this clock to get the ADC signal and generate test pulses. An internal PLL is used to obtain a fast 160 MHz clock for sterilization and transmission of data.
- **Digital Signal Processing** - Signal processing performs pedestal subtraction, mean CMS, and zero suppression.
- **Back-end processing** - Processing after the DSP chain is associated with the TFC commands.

5 Tests to be performed

- **Power tests** - Need to test proper start up of the chip. Make sure the digital and analogue currents are being drawn correctly. Need to simulate working conditions for the LHCb UT by applying expected trigger rate.
- **DAC tests** - Record output of each DAC over range of possible values to confirm proper behavior.
- **Digital tests** - Make sure the chip can be configured. This includes being able to read/write registers, read chip addresses, pass signals in both directions.

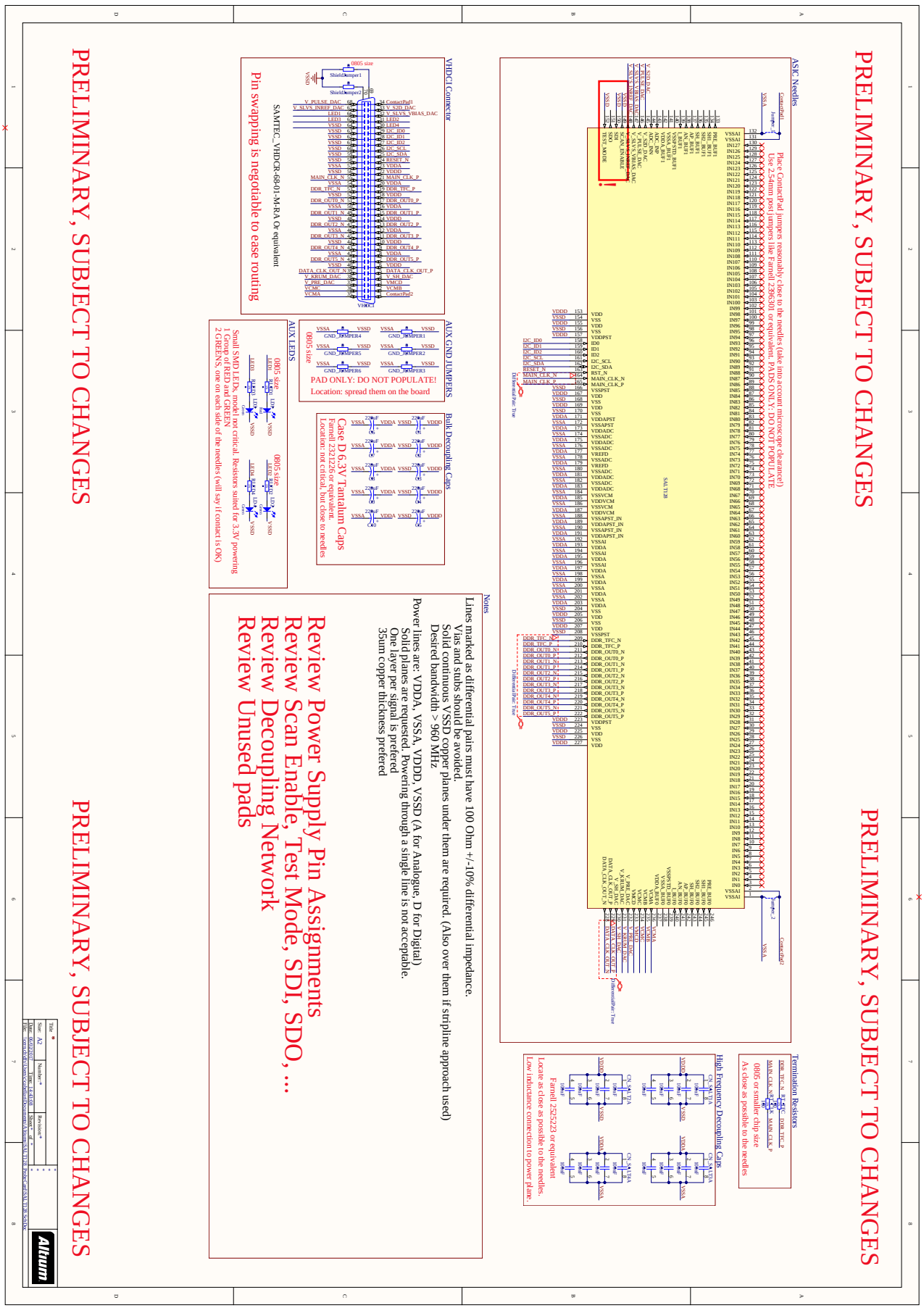

Figure 3 Schematic of the SALT128 chip with the input pads labeled and lines to the DAQ.

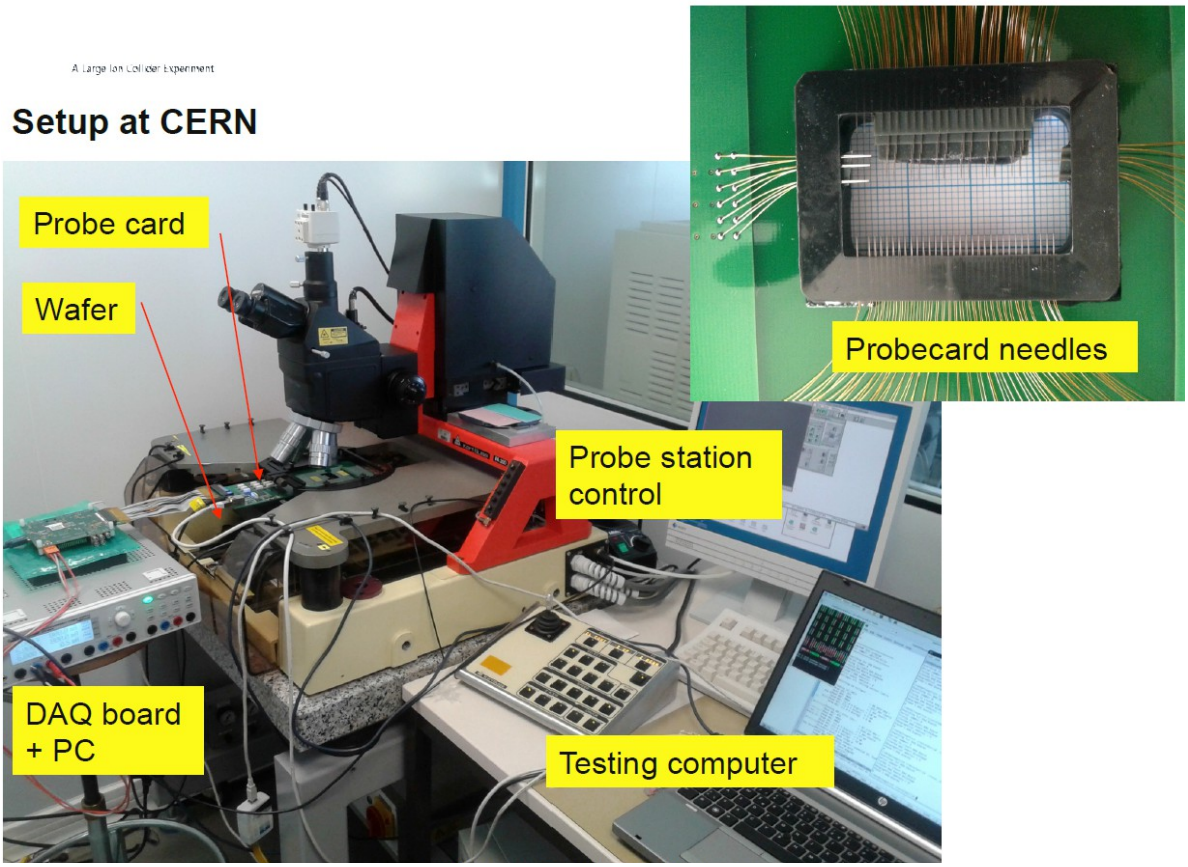

Figure 4 Picture of the semi-automatic probe station with connections to the PCs and DAQ (PRILIMARY PICTURE, TAKEN FROM *P. Riedler, 5th ALICE ITS, MFT and O2 Meeting*).

- **Analogue tests** - Should perform five point gain tests (inject five different charges) and read out ADC values to assess analogue functionality.
- **Gain uniformity** - Check output signal uniformity across channels for a given input. Should test for different input charge.
- **Clock checks** - Check different phase shifts of the clock and data lines to confirm optimal value.
- **Cross-talk studies** - Confirm that cross-talk is less than 5% between neighboring channels.
- **Pedestal and CMS** - Check proper Pedestal subtraction and CMS.
- **Noise studies** - Measure the intrinsic and CMS noise per channel.

6 Measurement details

6.1 Power tests

Perhaps the single most important test is that of power consumption. In the majority of cases, if the chip draws the correct current, then most of the functionality should work properly. On the other had, a large deviation from the expected power consumption usually indicates a serious issue or defective chip. These tests will be performed using external ADCs.

Two power consumption tests per chip are envisioned:

- Power consumption test right after powering up SALT128.
- Power consumption test after all other tests once the chip has been fully configured.

Both tests should check if the current being drawn is within $\pm 20\%$ of expectations, otherwise the chip is flagged as BAD.

6.2 Digital communication

SALT communicates with the outside world with three interfaces: a slow (100 KHz) I^2C read-write interface that controls all digital logic of the chip; two fast (320 Mb/s) read (TFC) and write (DAQ) interfaces. Proper chip functionality requires testing of all three interfaces as well as proper clock configuration and synchronization (DLL and PLL).

- **I^2C** - Testing proper I^2C involves ensuring proper writing and reading of registers. This is accomplished with the `pattern_cfg` register of the Serializer. $1/10^6$ incorrect bits indicates a faulty chip, thus about 200k patterns need to be tested.
  Test details
  1. Reset and make sure I^2C is not busy.
  2. Configure PLL by writing b10001101 to the `pll_main_cfg` register. If unable to configure, flag chip as BAD.
  3. Write a random bit pattern to the `pattern_cfg` register. Read back the pattern in the `pattern_cfg` register and make sure it is the same as was written. If the read pattern is different than the written one, then flag the chip as BAD.
  4. Reapeat step 3. 200k times.
- **DLL configuration** - The following configuration of the DLL is based on section 5.1.1 of the SALT ver 3 chip documentation.
  Test details
  1. Set correct value of CP current by writing `h9A` to the `dll_cp_cfg` register.

2. Set start value of `h60` to the `dll_vcdl_cfg` register.
3. Wait for 1 μs.
4. Read `dll_cur_ok` bit from the `dll_vcdl_mon` register and make sure it is 0, otherwise increase start value of `dll_vcdl_cfg` by 1.
5. Repeat step 4 until `dll_cur_ok` bit from the `dll_vcdl_mon` register is 0. If failure to reach 0 after N tries, then classify chip as BAD.
6. Decrease `dll_vcdl_cfg` by 1.
7. Wait for 1 μs.
8. Read `dll_cur_ok` bit from the `dll_vcdl_mon` register and make sure it is 0, otherwise decrease start value of `dll_vcdl_cfg` by 1.
9. Repeat step 8 until `dll_cur_ok` bit from the `dll_vcdl_mon` register is 0. If failure to reach 0 after N tries, then classify chip as BAD.
10. Set `dll_start` to 1
11. Read the `dll_vcdl_voltage` bits in the `dll_vcdl_mon` register and make sure this is 0. A value of ± 8 is also acceptable, although suboptimal. If value is outside this range, then flag chip as BAD.

- **PLL configuration** - The following configuration of the PLL is based on section 5.2.1 of the SALT ver 3 chip documentation.
  Test details

  1. Make sure PLL is configured by reading the `pll_main_cfg` register and making sure it is set to b10001101. If not, write b10001101 to the `pll_main_cfg` register.
  2. Read value of the `pll_cp_cfg` register. If not the default value of b10011010, then write b10011010 to the `pll_cp_cfg` register.
  3. Read value of the `pll_vco_cur` bits in the `pll_vco_cfg` register.
  4. If value of the `pll_vco_cur` is equal to 0 ± 8, then PLL is properly configured.

- **DAQ-clock synchronization** - Checking proper functionality of the DAQ lines (E-links) requires synchronization with the clock using a known pattern and making sure pattern can be read out properly. The chosen pattern will be `hAB`.

  1. Set the correct pattern for synching by writing `hAB` to the `pattern_cfg` register.
  2. Reset `ser_source_cfg` and set output of DAQ to the pattern by writing `h22` to the `ser_source_cfg` register.
  3. Reset DAQ by writing `h01` to the `DAQ_CFG` FPGA register.
  4. Clear FPGA FIFO by writing `h10` to the `DAQ_CFG` FPGA register.
  5. Start a synchronization cycle by writing `h02` to teh `DAQ_CFG` FPGA register.
  6. Continuously read the `DAQ_CFG` FPGA register until it b2 reads 1, which indicates that the synchronization procedure is done and all e-links have seen at least one good synchronization.
  7. Properly read the output of the E-links and make sure the correct pattern (`hAB`) is being read out. If output is `hAB`, wait 1 us and go back to step 3.
  8. If synchronization fails, i.e. if read back pattern is not `hAB`, then need to shift phase in PLL. This can be done by writing `h01` to register 0x40018 (PLL register, no name yet) of the FPGA, which defines the phase to shift by 1/8 of the clock period, and writing 1 to b5 of the register 0x40020 of the FPGA, which defines the shift to increment up.
  9. Make sure you can synchronize 10k times in a row. If yes, then synchronization complete, otherwise flag chip as BAD.

- **TFC check** - There are 8 TFC commands that are recognized by SALT as outlined in the SALT manual: BXReset, FEReset, Header, NZS, BxVeto, Snapshot, Synch, and Calib. Here we test some basic commands: BXReset, FEReset, Heaser and BxVeto, .
  Test details

1. Reset the TFC state machine and set all related registers to default values. This is done by writing b0=1 in the `TFC_CFG` FPGA register.
2. Set TFC to single transmission by writing 1 to b1 in the `TFC_CFG` FPGA register.
3. Set the appropriate Sync data packets to their reset values by writing to the `syncX_cfg` registers.
4. Reset the BXID counter and define Sync data output. First define the TFC length (number of commands, in this case 2) by writing `h02` to the `TFC_LENGTH` FPGA register. To reset the BXID, write b00000001 to the `TFC_WR` FPGA register. Next set the Sync command by writing b01000000 to the `TFC_WR` FPGA register. Write `h01` to the `TFC_TRIGGER` FPGA register to begin TFC command execuation.
5. Keep reading the `TFC_TRIGGER` FPGA register until is reads `h00`. Then read out data packet and verify that the correct bxid and sync configurations are being read out (see Figure 16, Sync5 in the SALT manual for the correct format). If Sync failed, classify chip as BAD.
6. Reset TFC registers and empty data buffers by performing an FEReset by writing b00000010 to the `TFC_WR` FPGA register and trigger execution of this command with the the `TFC_TRIGGER` FPGA register. Verify execuation was completed by reading the `TFC_TRIGGER` FPGA register until is reads `h00`.
7. Check Header command by writing b00000100 to the `TFC_WR` FPGA register and trigger execution of this command with the the `TFC_TRIGGER` FPGA register. Keep reading the `TFC_TRIGGER` FPGA register until is reads `h00`. Make sure only a header is read as the first byte. If not, classify chip as BAD.
8. Repeat step 6.
9. Check BxVeto command by writing b00010000 to the `TFC_WR` FPGA register and trigger execution of this command with the the `TFC_TRIGGER` FPGA register. Keep reading the `TFC_TRIGGER` FPGA register until is reads `h00`. Make sure only a header is read as the first byte. If not, classify chip as BAD.

6.3 Digital signal processing

- **NZS data packet test** -
- **Pedestal subtraction test** -
- **ADC and DSP synchronization** -
- **Memory overflow test** -

6.4 Analogue functionality

1. Power tests
   Check proper power consumption of ANA and DIG parts after power-up and reset
2. Communication (I^2C) tests
   $\rightarrow$ There are about 450 registers $\sim$ 10kb total amount of data, which should not be a with limited FPGA RAM
   (a) Serializer registers
   (b) DSP registers NZS path
   (c) DSP registers rest
   (d) SEU registers
   (e) Basing DACs and monitoring ADCs basic functionality test
   $\rightarrow$ Need to estimate time needed to test all values of DAC (input from Carlos needed)

3. DLL, PLL, Bandgap tests (using internal monitors) and configuration
   → DLL, PLL values close to 0 ADC would mean proper functionality
   → We need to learn how to configure different blocks
   (a) DLL test SALT documentation section 5.1.2.
   (b) PLL test SALT documentation section 5.2.1.
   (c) Bandgap test and readout of temperature sensor
4. Serializer and Deserializer tests and configuration
   → Need to check if we can move first byte and correct pattern, then this works correctly
   → The PLL should be fully functional before we get to this stage
   → Need to see if Serializer works before testing Deserializer
5. Basic TFC commands and data packet tests
   (a) SYNC (FeReset)
   (b) HeaderOnly, BxVeto, BxReset
6. NZS data packet and Pedestal Subtraction tests (in meantime also mask register could be tested)
   → Need to send a pedestal value to each channel. How do we upload pedestal value to every channel several times?
   → Need to send to each channel several mask registers
   → Need input from JC, since he is testing this now
7. ADC and front-end baseline correction DAQ tests
   → Need to discuss with JC, he is doing baseline correction tests → Want to test baseline DAQ and test ADC
   → Send some data, min code, max doce, middle code and make sure DAQ are reading correctly and see how they change
   → Through the I^2C prgram DAQ, and through ADC program a few values and check
   → Are we going to record for individual channels or not?
   → Need to get offset and spread, which is needed to correct pedestals. Need to have several points here to find correct value.
   → We can read all channels at the same time, since this is doen in parallel
   → We are testing Trim DAQ and base ADC functionality
   → How wide a range can we test? Not the full range of ADC but we need to test full range of Trim DAQ.
   → No ADC at this point, only an offset (this is a static measurement, since we only change the offset)
   Points 1-7 have validated most important blocks
8. Front-end performance tests (using internal calibration)
   (a) Gain, Offset, Crosstalk
   → 5 point Gain curve. Can do for every channel in parallel (or every fourth channel). Will there be a problem of occupancy of the chip?
   → What about timing, delays, etc.?
   → Need to set correct value of delay in DLL before measurement of gain and offset. Can get some basic numbers from current tests.
   → Can we use internal calibration even if pad is connected to something else?
   → Do we need to test the input pad? Probably not, since for these tests we are only using internal calibration
   → Need to define how many input needles we need for the probe card
   → Will a huge load capacitance (i.e. when sensor is connected) affect the performance of ths chip? Should ask people who design ASICs for ATLAS. Do we need to test inputs? Should decide this soon, but maybe it's not an issue
   → Need to define minimum number of needles needed for power (See point above about how many needles needed on probe card). Carlos, Chris, Iaroslava and Olaf can figure this out. Olaf will check what was done for the Beetle.

Shift Start

Check I2C communication for external ADCs

Start tests

Failed, Bad Chip

Power up SALT128

Power checks

Power consumption. Within tolerance?

No

Yes

Digital Communication & Clock checks

I2C tests

Fail

Pass

DLL & PLL confirguration

Fail

Pass

DAC tests (digital comunication)

Fail

Pass

DAQ clock synchronization

Fail

Pass

TFC tests

Fail

Pass

Digital Signal Processing

NZS data packet & Pedestal subtraction test

Fail

Pass

ADC & DSP synchronization (& memory overflow)

Fail

Pass

Analogue Functionality

DAC tests (analogue functionality)

Fail

Pass

ADC & FE baseline correction & noise tests

Fail

Pass

Pulse shape, gain, uniformity & crosstalk

bandgap tests

Fail

Pass

Power checks

Power consumption. Within tolerance?

Fail

Pass

Finished. Classify # of bad channels