## Following problems occured during CAN FD controller development:

(in code marked with TODO)

### 16.5.2015

**Prescaler.vhd**  - wrong hard synchronization. The bit timing needs to be set to the position after sync segment! **SOLVED -** Prescaler is reipmplemented in **Prescaler\_v2.vhd !**

**BitStuffing.vhd -** Check the "recursive" behavior of bit-stuffer!!! **SOLVED -** Bit Stuffing is reipmplemented in **BitStuffing\_v2.vhd ! Recursive behaviour verified in new testbench!!**

### 20.5.2015

**CRC.vhd** possible optimization in 'trig' signal routing for crc calculation. After synthesis check RTL schematics if clocks are not gated before entering the register.

**SOLVED -** Trig signal comparison moved so that calculated value on output stayes also after the circuit operation is disabled (after CRC calculation it is transmitted)

### 22.5.2015

**rx.vhd** check synthesis if multiplier was not used in for loop iteration while storing recieved frame into FIFO memory!!!! Check STA of rxBuffer.vhd!!

**SOLVED -** for loop not used for synthesis due to huge size of final design. Whole adress logic generated 16 times instead of one time!! Sequential storing used instead! (see rxBuffer.vhd)

### 25.5.2015

**rxBuffer.vhd** check if modulo operation can be deleted for synthesis!!! ModelSim testbench throws an error when increasing the pointer above the range. In Hardware overflow should appear and following should be equal (write pointer range 0 to 31) :

write\_pointer<=write\_pointer+1 (31 +1 =32 - overflow write\_pointer<=0)

is equal to

write\_pointer<=(write\_pointer+1) mod buffSize

Consequence : buffSize can only has size of 2^k;

SOLVED : Modulo operation is not removed for synthesis! Assuming synthesis automatically upper part of resulting std\_logic\_vector. All other pointers used without modulo operation and overflow is assumed! 11.7.2015

### 25.5.2015

Finish automatic test for rxBuffer (File rxBuffer\_tb2.vhd) with generating random data for writing and reading at random times

SOLVED: Automatic test including recieve buffer functionality implemented in top\_level\_tb\_2.vhd

### 30.5.2015

Delete the driving bus signals for deprecated timestamp generator. Add at least one range filter for **messageFilter.vhd** . Think of optimization of combination functions for the filters!!! TODO: Testbench for range fitler!!

SOLVED: Driving bus signals deleted. Filter D replaced with range filter! 11.7.2015

Change the explicit values of rx\_buffer message indexes and txt\_buffer message indexes for constants!!!

### 4.6.2015

Implement testbench for TXBuffer!!!

DEPRECATED: TXBuffer not used in the end, testbench not necessary 11.7.2015

### 12.6.2015

For **bitStuffing** and bit Destuffing implemenent logic for switching triggering signals based on what type of Frame is beeing Recieved or Transcieved (Nominal bit time, Data bit time).

**SOLVED -** Bit Stuffing moved to the CAN Core and triggering the communication with bs\_trig, bds\_trig which are driven by sp\_control (type of bit time)!

For **Interrupt manager** check the functionality of TX,RX, DOV interrupt. Due to handShake protocol it might happend interrupt will be called twice!

For **rxBuffer.vhd** check if message is not stored twice due to handshake protocol between CAN Core and RxBuffer.

SOLVED - Message is not stored twice since handshake protocol is not used. Only rec\_valid signal is used which is active for one clock cycle 11.7.2015

### 15.6.2015

**faultConf.vhd** sets output error\_valid as combination function of errors. Error input is hold for whole BIT TIME! Therefore lot of interrupts would be fired! TODO: Edge detection on the bit errors, or synchronisation with the bit time!!!!!!

SOLVED: Fault confinement reimplemented and error\_valid signal is hold only for one clock period!

11.7.2015

### 17.6.2015

**ProtocolController.vhd , bitStuffing.vhd** check whenever bit Stuffing counter is erased when SOF.

**SOLVED -** Bit Stuffing controller is erased properly. Communication with Bit Stuffing, Destuffing tested in **core\_top\_tb1.vhd** with big amount (thousands of frames) of random frames and data. Data send by NODE 1 and recieved by NODE 2. Then compared.

For CAN Base format, the IDE bit belongs to ARBITRATION field in my implementation. When it is recessive it is supposed to go on with ARB field. If it is dominant then it is last dominant bit of ARB field. Therefore the node which sends this bit as dominant always wins the arbitration.

IDEA: What if two nodes are sending at the same time : one CAN Data frame , second CAN FD Data frame with same BASE Identifier. Then Collision will appear.

**SOLVED -** The collision is supposed to appear as well as in normal CAN Frame

### 18.6.2015

Protocol Core, CRC. Verify whenever correct value is chosen for CRC field, based on data length. Verify functionality of CAN Frame with longer DLC than 8.

SOLVED - CRC functionality verified by calculating the CRC by both transciever and reciever!

11.7.2015

### 22.6.2015

Verify reciever functionality of CAN Core!

SOLVED - Functionality verified. core\_top\_tb implemented with various tests over random data!

11.7.2015

### 23.6.2015

Implement the Bit Error and stuff Error validation functionality of **Fault Confinement.vhd.** Implement counters assignment for all error\_counters!

SOLVED - Implemented in Protocol Control and Fault COnfinement reimplemented! 11.7.2015

### 24.6.2015

Solve out all TODOs (mostly bit Error detection in special fields of PC State) in **protocolControl.vhd**

### 24.6.2015

Test Bit Stuffing for arbitration field has all zero data. SOF Skip technique might cause troubles!!!!

SOLVED - SOF Skip technique not causing trouble correct, Bit Stuffing and bit Destuffing verified on testbench with random data as well as with all zero data and identifier! Therefore mistakes in stuffing would be known because maximum amount of stuff bits were inserted! 11.7.2015

### 29.6.2015

Prescaler\_v2.vhd verify if Latch is not inferred for hard\_sync\_valid signal.

SOLVED: all latches which were inferred were repaired to inferr flip flops!

Also check the behaviour of resynchronisation properly for short Bit times!!

Also check if hard synchronisation might cause glitches when the edge appears in PH2 therefore causing skypping sync segment (not having sync\_nbt tirgger) and causing having two sample points following without transcieve trigger!! Bits affected (EDL,r0, SOF)...

SOLVED: Sycnhronisation reimplemented in prescaler\_v3 !! (and tested)

### 1.7.2015

Abort Transmittion logic implemented. Implement testbench for aborting transmittion!

SOLVED : Aborting transmittion tested in top\_level\_2\_tb. 11.7.2015

**txBuffer.vhd** edge detection on store command implemented so that when drv\_store\_tx becomes logic 1, message is stored only once into the buffer.

SOLVED : Tx buffer deprecated so not needed to modify. 11.7.2015

**txtBuffer.vhd** edge detection on store command implemented so that when drv\_store\_txt becomes logic 1, message is stored only once into the buffer.

SOLVED: 1: Txt buffer looks for edge on drv\_store\_txt signal. Therefore data is stored only once! 11.7.2015

SOLVED: The same logic implemented for rxBuffer.vhd except there is used negative edge detection to move to the next data when the read finishes! 11.7.2015

Implemente testbench for Message counters!

SOLVED: TestBench implemented in top\_level\_tb\_2.vhd

### 2.7.2015

Possible optimalisaion in txBuffer! Message shouldnt be registered on the output but loaded directly from memory!!!

SOLVED: txBuffer not used therefore optimisation is not necessary 11.7.2015

### 3.7.2015

Reimplement FaultConfinement once again. Rework the interface of Fault Conf and coreTop. Move the logic of errors to the Protocol control!!

SOLVED: Fault confinement reimplemented functionality verified in core\_top\_tb. 11.7.2015