## 15.5.2015

Development log started today. Work already in progress for about two weeks. So far simple circuits and according testbenches are implemented. Just few days spent of actual work.

**BitStuffing.vhd** module with internal buffer and generic value of number of bits after which insert stuff bit. Fixed bit stuffing for CRC phase of CAN FD implemented.

**Prescaler.vhd** module for generating time quantum clock and bit time clock for normal speed and for data speed, based on driving signals from DRV\_BUS. Generating also signal for sample point. Generating two pulses with delay of one and two clk\_sys cycles from start of bit time.

**tsGen.vhd** time stamp generator with 64 bit time stamp. Connected via 32-bit bidir bus. Dedicated not as part of CAN controller but on system level (same level of components as EMIF decoder, GPR...). 4 Internal registers for storing time stamp value and prescaler register.

**timeStampGen.vhd** deprecated component already developed for timeStamp generation inside the CAN controller. After precise assignment disscusion time stamp generation was moved to upper level to be common for different controllers. Functionality of time Stamping now at tsGen.vhd.

## 20.5.2015

**MessageFilter.vhd** implemented. Four filters (A,B,C,D) for frame type and bits of frame are implemented. Output value of filter registered with clk\_sys. Filter frame types are selectable.

**CRC.vhd** block for CRC calculation implemented with three generator polynomials (data length dependant). Serial data input. Synchronous to clk\_sys. Starts with 0 to 1 transition on enable signal. Trig signal has to be in logic 1. Operates as long as enable=1.

## 25.5.2015

**RxBuffer.vhd** implemented with generic length. RAM type memory used . Read pointer and write pointer used for proper simultaneous read and write. Write of one message (up to 18 words) in one clock cycle. Read of one word (4 bytes) in one clock cycle by propagating the word to the output. Moving to the next read word by drv\_bus signals operation.

## 30.5.2015

DRV Bus ranges transfered to constants in CANConstans file. TXT Buffer for sending messages in specific time implemented! Single message can be stored (RAM memory of 640 bits). For message format see document describing driving bus.

**txArbitrator.vhd** circuit was implemented for voting between TX Buffer messages and TXT buffer messages by comparing the timestamp value (input signal assumed synchronous with clk\_sys)

### 1.6.2015

**messageFilter.vhd** bit filter D erased and replaced by range filter with same control bits and two ranges. Message is accepted if its identifier fits between lower and upper boundary. Output of range filter is "or" gated with other filters

### 4.6.2015

**txBuffer.vhd** circuit implemented. Normal TX Buffer FIFO like memory, without time priorities. settable size of 32,64,128,256 words (32 bit). Testbench not yet implemented!!

**Prescaler\_v2.vhd** started today. Reasons for implementing new prescaler:

-Prescaler.vhd didnt support bit time synchronisation (hard, resync). Therefore new prescaler have to be implemented! Also in the bus time up to three adders were used in combinational logic (to sum upp prop,ph1,ph2 segments). Therefore new state machine like bit time counter will be introduced (Four states for each part of the bit time). The development of the circuit just started. The synchronisation logic moved to the prescaler from **busSync.vhd**.

### 10.6.2015

**busSync.vhd**. circuit started. Provides synchronisation chain for bus input. Detects edges which are suitable for synchronisation. Provides 'edge\_rx\_valid' signal for **prescaler\_v2.vhd** signaling edge on the RX data. On this edge prescaler can synchronise hard or resynchronize. Circuit provides measuring of transciever delay compensation in clk\_sys periods. Circuit can sample the income bus values on nominal bit time, data bit time. Sampling with the compensation not yet implemented.

### 11.6.2015

**busSync.vhd**. circuit finished. Secondary sample point implemented. Edge detection implemented. Bit Error detection based on comparison of sampled value and transcieved value implemented. One simple testBench implemented for edge detection and synchronisation chains. One complex textbench implemented for sending the data with bit rate shifting. EDS, r0 and BRS bits are sent! During this time transciever delay is meausured. Then bit rate is shifted and data are sampled with secondary sample point. Bit Error detection is enabled and errors are detected whenever delayed transcieved data is not equal to recieved data.

**bitStuffing\_v2.vhd** implemented. For simple handshake operation with CAN Core without internal buffer

### 12.6.2015

**bitDeStuffing.vhd**. circuit implemented with appropriate testbench. Bit Destuffing for normal bit stuff rules as well as for fixed stuffing. Stuff Error detection whenever 6 consecutive bits are detected.

Note: Stuff length changed to input signal from generic constant, due to different stuff length for fixed stuffing and normal stuffing.

**IntMan.vhd** implemented for handling interrupt based on different interrupt sources.

### 15.6.2015

TestBench implemented for Interrupt manager. **faultConf.vhd** implemented for fault confinement. Implemented separated Error counters for normal CAN and CAN Flexible DataRate. All counters are presetable from driving bus. In the driving registers idea is to make the error counters read/write registers. Since the errors have to modify counters, registers of counters are placed in faultConf block and propagated out as input to driving registers. This way read behaviour is guaranteed. Write/Preset behaviour is guaranteed via driving bus.

Signals for driving the error counters have part of the logic in **the faultConf.vhd** block , part will be implemented in CAN Core. Possible optimization of the circuit.

### 16.6.2015

CanCore implementation started. So far the design of on paper drawn state machines. Triggering signals logic retought, IO ports of CAN Core defined. Block scheme of the CAN Core designed. Unit **core\_top.vhd** implementation started.

### 17.6.2015

**protocolControl.vhd** unit started with state machine handling CAN Protocol. So Far SOF, Arbitration field implemented and Control field started.

### 18.6.2015

**protocolControl.vhd** unit finished in first version. Simple testbench for sending message created and verified. Error Frame, Overload frame, Intermission remain to be implemented.

### 19.6.2015

**protocolControl.vhd** unit development. Error Frame implementation. EOF Implementation and Intermission implementation. All registers in the state machine preset to previous value to avoid latch synthesis. Protocol control state machine refactoring.

### 22.6.2015

**protocolControl.vhd** overload frame implementation. Finishing the states implementation and implementation of starting state (Intermission - Idle). Adding control signals into states for : Bit Stuffing and Destuffing, Bus Synchronisation, Secondary sample point calibration. Bit rate shifting.

**operationControl.vhd** state machine implementation. Transciever reciever, bus Idle, Integrating. Communication of OperationControl and protocolControl.vhd

**protocolControl\_tb1.vhd** testBench to verify all types of frames as transciever! Implementation and verification!

### 23.6.2015

Fault confinement unit moved to the CAN Core. Bit Stuffing and Bit destuffing moved to the CAN Core. Interface change at **FaultConf.vhd** . Implementation needs to be reworked due to complete interface change. New definite interface defined.

**protocolControl.vhd** implementation of control signals for errors. Errors divided into two groups:

Detected by PC\_control : ACK error, CRC error, form error

Detected by fault confinement : stuff error, bit Error

First group of errors is handled inside PC control FSM and switches to error state. Second group will be validated in **FaultConf.vhd** whenever error truly appeared (ex. Bit error wil be always detected when unit is reciever, because unit sends just recessive bits, CAN Bus sync unit doesnt have the information about OP State!) and signalled back to CAN Core by signal : bit\_Stuff\_err\_valid. If this Error appears unit starts to transcieve Error Frame from next Frame!

Implementing support for disabling FD frames for recieval and sending error frame. Implementing support for Bus Monitoring mode with internal reroute of TX,RX data lines.

**core\_top.vhd** implementation started! So far components of the CAN Core defined as well as common signals! Two CRC units will be used all together then calculating 6 CRCs at the same time. One CRC for bit stuffed stream , one for bit destuffed stream!

Trigger signals logic remains, Retransmit logic remains, Storing valid recieved data logic remains, CRC multiplexing logic remains, Loopback multiplexing logic remains!

### 24.6.2015

Fault confinement counters reimplemented with new interface!

tran\_brs signal added into **TxArbitrator.vhd, TxBuffer.vhd, TxtBuffer.vhd, RecieveBuffer.vhd.** for optional bit rate shifting in CAN FD frames.

### 25-26.6.2015

**core\_top\_tb1.vhd** testbench implemented. Two CAN Core nodes are instantiated one acting as transciever second as reciever. Random data is sent and checked if correctly recieved. Bit Rate shifting is not working properly so far because only asynchronus triggering signals are used for both bitRates . New prescaler will handle this trigger signal synchronysation!

Simple Testbench for arbitration verification developed.

### 27.6.2015

Status Bus Implementation. Basic signals from core\_top assigned to STAT\_BUS. Still control pointer and transcieve pointer need to be added to support arbitration lost functionality!

### 29.6.2015

**prescaler\_v2.vhd** implemented with synchronisation, hard synchronisation time quantum clock generation and triggering signals clock generation. Appropriate testbench implemented. Synchronisation mechanism not absolutetly properly verified so far.

### 30.6.2015

**registers.vhd** implemented. Register map created and driving bus signals assigned. Avalon compatible bidirectional bus used for the memory access. Bidirectional 32 bit bus approach not considered. Therefore EMIF Decoder, General register, Output multiplexor from bachelor thesis has to be re-implemented!!!

### 1.7.2015

**registers.vhd** finished and STAT\_BUS debugged.

Error debugging ofCAN Core. Creating testbenches for error detection in arbitration field, SOF Field. Error frame debugging and stuff error detection by other node.

### 2.7.2015

**CAN\_top\_level.vhd** implemented. Components instantiated. Error testbenches finished. Error passive flag verification! Debugging of all signals in the design, removal of unnecessary signals! Error frame modifications. DRV\_BUS debugging and shortcircuit removal.

### 3.7.2015

Fault confinement interface reimplemented. Logic of error counting moved to the protocol Control. Error counting changed. Two error counters used , RX,TX for fault confinement and two separate counters used for error detection in two different bit rates!

Error testebenches reTested once more!

**tsGen.vhd** memory acess reimplemented to be compatible with 32 bit Avalon compatible bus!

**outMux.vhd** component from bachelor thesis reimplemented to be compatible with Avalon 32 bit bus, instead of bidirectional bus. Component refactored and support for CAN controllers output signals driving added (with additional registers). Conditional code synthesis used if controllers of some bus are not availiable!

**dataMux.vhd** component on system level implementation started for data\_out switching of CAN,LIN,FLEXRAY nodes in FPGA for EMIF\_DECODER.

### 6.7.2015

**EMIFDecoder.vhd** reimplemented from bachelor thesis. Synnchronnisation chains added. Interface changed from EMIF / 32 bit bidirectional to EMIF/Avalon bus.

**top\_level\_1.vhd** Top level entity with example system implementation started.

### 7.7.2015

**top\_level\_1.vhd** finished in first version . Testing with hardware began. Hardware problems. Solving programming issues

### 8.7.2015

EMIF reading and writing issues testing of real EMIF Decoder. MPU configuration in HALCoGen changed to DEVICE type memory for EMIF segment to achieve good behaviour.

### 9.7.2015

EMIF testing of writes and reads finished. Logic analyzer Signal Tap II used for waveform checking. Automatized testing implemented in Code Composer Studio.

First Synthesis of whole system! Due to huge size of decoders of TX Buffer and RX Buffer (FIFO memory with for generate, around 49 000 logic blocks) architecture changed. TX Buffer removed from design. Instead second TXT Buffer was used. TxArbitrator functionality modified in the appropriate way! The dataPath of TX Arbitrator remains unchanged! TX Data now have only one format (format B from before).

**RX\_Buffer.vhd** storing of recieved message changed to save space on FPGA. Sequential storing of the data used instead of concurrent storing! For generate construct was generating too much logic

### 10.7.2015

**top\_level\_2\_tb.vhd** implementation started! procedures for read, write on EMIF bus. The same top level entity as the final top level entity into FPGA tested. Simple functionalities verified as commiting message into TXT1 Buffer, sending message on the bus. Retransmittion limit tested. Aborting of transmittion tested. Self test mode tested. First synthesis with good size. FPGA configuration, writes to registers. Implementation of simple program to insert message in Code Composer studio. Configurating of external transciever. Solving issues of synthesis with IF Generate. TXT Buffer of different sizes not working, synthetiser implements somehow both at the same time. Functionality was corrupted. Conditional synthesis for CAN bus removed from top level entity and TXT buffer.

### 11.7.2015

Commenting uncomented desgins, refactoring debugging errors.

**logger.vhd** implementation started. Triggering conditions and type of events to log determined. Register structure for acessing the logger implemented. Driving bus signals for controlling the logger added . Stat bus signals added to allow comfortable trigerring. Trigerring logic implemented!

### 13.7.2015

Debugging on hardware, measuring and decoding with oscilloscope. Synthesis configuration. Simple CAN tests in C language for whole system. Creating top level entity of FlexRay controller by Martin Patak . Creating common ModelSim project.

### 14.7.2015

**top\_level\_tb\_2.vhd** testbench for testbenching top level enity which will be loaded into test platform! Implementation of interrupt tests. Implementation of transcieve recieve buffer test.

Major flaw in design found! Transciever dela compensation mechanism in FD frames was causing Protocol control machine to not work correctly. Correction of implementation. Secondary sample point not used as recieve trigger! Used only in busSync.vhd to detect bit error while transcieving FD.

Delayed transcieve trigger used as recieve trigger. This trigger created in separate proces in core\_top.vhd! CRC calculation has to be changed in this case! Calculation of CRC not from recieved data for data phase of FD transciever but from transcieved data. Multiplexors for this purpose implemented in CANCore. If CRC would be calculated from recieved data recieve trigger would be used (secondary sample) this caused errors since this signal was delayed by transciever delay. Thus sequence : SYNC,SAMPLE,SYNC,SAMPLE.... contained several SYNCs without sample. This caused state machine to sent ESI bit twice!

Note: Solution was verified for recieving valid data!!!

### 15.7.2015

**top\_level\_tb\_2.vhd** finishing solution of secondary sample point problem! Implementation of Interrupt tests. Debugging of Recieve buffer test with random data! Implementation of Fault confinement state test, Data Overrun Test, TXT arbitrator test, Acceptance filter test, message counters test!

### 17.7.2015

Real hardware tests started. Several errors in PC State debugged!

### 21.7.2015

prescaler circuit reimplemented. **Prescaler\_v3.vhd**  implemented. Separate state for hard synchronisation used! Hard synchronisation in EDL bit of FD Frame not used . New Hard synchronisation mechanism was causing problems!!! Arbitration tested with new prescaler in real hardware. Hard Synchronisation and resynchronisation tested!

### 23.7.2015

Writing of sample functions in TestPlatform project to acess the controller.