Projeto IF747 / IN1055
Redes Automotivas

Main Goal:

- **Part A:**
  - Groups of 2 people
  - Implementation on Quartus using the Testbench for simulation
  - Available topics:
    i. **CAN Decoder**
       - Recognizing incoming data and associate it to the different CAN frame types (Data, Remote, Error and Overload)
       - Bit-stuffing needs to be taken into account.
       - Error detection

    ii. **CAN Bit timing logic**
       - Recognizing the different phases associated with the bit-timing of each of the incoming bits, generating the probe sampling point for the decoder to read the bit value.
  - The graduate students will implement topic (ii) and the undergraduate students topic (i).
  - VHDL or Verilog may be used (except on the Arduino).
  - CAN frames generator using the Arduino CAN shield for both groups

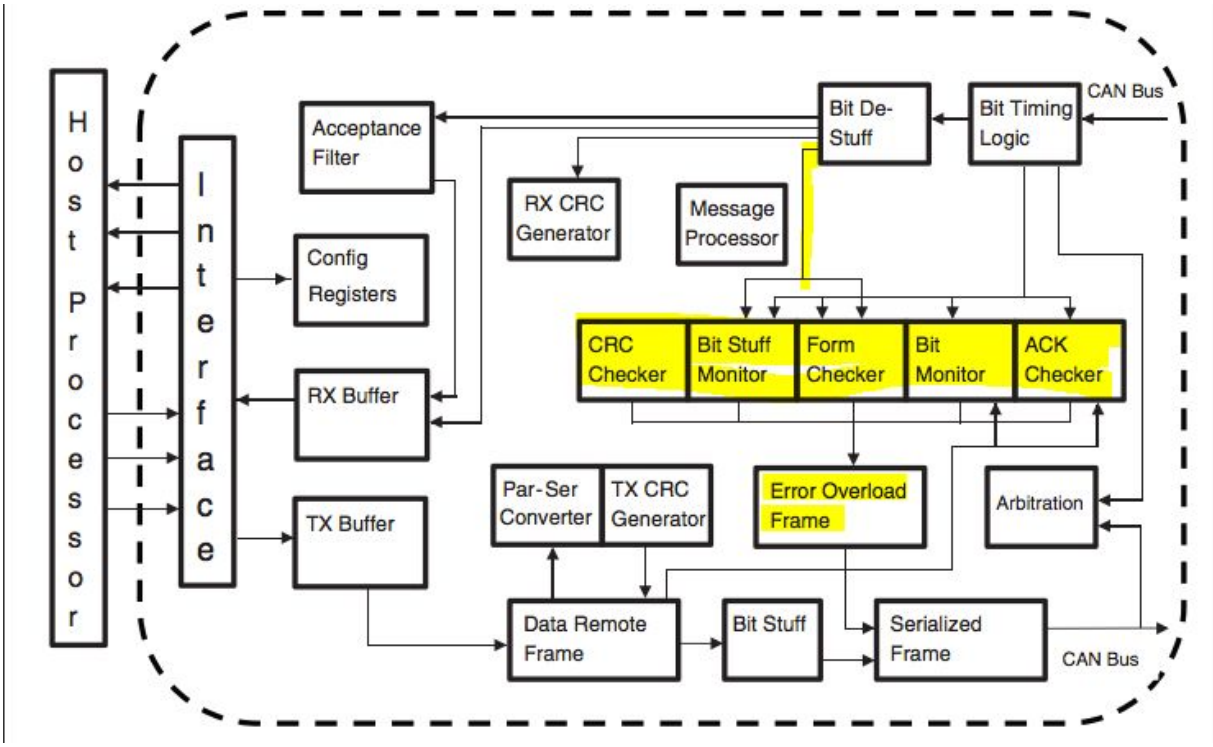
- **Part B:**
  - Each group assigned for topic (i) will join another group assigned for topic (ii).
  - The use of an FPGA board with a CAN transceiver will be used as node 1. It will merge the decoder and bit-timing logic, implemented on Part A.
  - An Arduino board with a CAN shield as the frame generator (node 2)

Resources:

- DE2i-150 FPGA Development Kit (available in the Hardware Lab)
- CAN transceiver, resistors, capacitors...
- Arduino + CAN Shield
- Quartus

Schedule:

- **Part A: 12/06/17**
- **Part B: 03/07/17**



Seguem algumas explicações sobre o projeto.

1) A atividade principal do projeto é projetar e implementar um decoder CAN, ou seja, um módulo (que pode e deve ser composto por vários outros módulos) capaz de receber bits (0s e 1s) serialmente e montar o frame CAN que está sendo transmitido no barramento. Quantos e quais submódulos vão constituir o decoder é uma decisão de projeto e constituirá parte da avaliação. Desse modo, cabe a vocês decidir se é necessário ou não implementar os blocos em amarelo.

2) O importante aqui é montar o frame a partir dos bits recebidos, e a partir desta montagem conseguir afirmar se o frame recebido é standard ou extended, quantos bytes tem o seu datafield, se ocorreu erro de bit-stuffing, se ocorreu erro de bit dominante no final do frame, etc. Desse modo, a partir da montagem do frame vocês devem retornar o maior número de informações/características possíveis sobre o frame, atentando em especial para: identificação do start of frame, da ocorrência de frames de erro (das condições que ocasionam um erro) e da transição entre frames de erro, overload, remoto ou data.

3) A arquitetura global da rede é a seguinte: 1 nó CAN que transmite

mensagens para o barramento (seria o arduino + a CAN shield); 1 nó CAN que recebe as mensagens do barramento (seria a fpga + CAN transceiver); possivelmente 1 nó CAN que recebe as mensagens do barramento e transmite um acknowledgment (poderia ser outro arduino + CAN shield).

4) Quanto ao nó composto pela fpga e pelo transceiver tem-se que o transceiver é conectado ao barramento através do CAN H e do CAN L, e à fpga através do RX e do TX, que serão utilizadas duas portas GPIO 3.3V distintas para este propósito, uma fazendo o papel do TX e outra do RX. A fpga lê os bits do GPIO correspondente ao RX e escreve bits no correspondente ao TX. Deve-se escrever sempre 1 no TX de modo que não se altere o barramento. A função do BTL é exatamente fornecer um probe "sample point" que corresponde aos momentos em que se deve fazer a leitura do RX. Esses bits lidos são então entrada do decoder.