NTNU
Norwegian University of
Science and Technology

Project Thesis TFE4580

Automotive embedded system implementation using System-on-Module

Trym Sneltvedt

December 2019

Supervised by Bjørn B. Larsen

Unfinished version!
Notes and comments are colored red.

Contents

1 Introduction

In our digital world, the demand for computing platforms with higher performance and smaller footprints increases on a daily basis. In later years a class of integrated circuits (ICs) has emerged as the go-to solution for embedded devices where performance, features and small form factor are important parameters, System on Chips (SoCs).

SoCs are fully fledged computer platforms with the necessary memory and storage for operation and powerful peripherals like field programmeable gate arrays (FPGAs) or modems for wireless communication like WiFi, Bluetooth and ZigBee. All of this is packaged on a single chip, thereby the name. In contrast to traditional microcontrollers which has been a favourite in the embedded landscape for decades, SoCs pack a bigger punch. They are equipped with more powerful microprocessors, sometimes even multiprocessors. This makes them a sought after platform for applications where high throughput are crucial, like aerospace, automotive, signal processing, communication, multimedia and military use.

A downside to the increased complexity of the electronics and the demands for smaller devices is the packaging of ICS. Integrated circuits with a high pin count very often use ball grid array (BGA) packages. These are ICs with the connectors placed in a dense grid on the bottom of the packaging, instead of on the tradition side placement. While very space efficient, this way of connecting ICs to printed circuit boards (PCBs) introduces several challenges for hardware designers. Firstly, the clearance between each pin requires PCBs with more than 2 layers. Tradition methods of PCB manufacture like copper etching or milling is only possible (or practical) for 2 layer boards. This means prototypes has to be produced at larger production factories that has the expensive equipment available. Multi-layer boards are also more expensive. It is true that the manufacturing cost and lead time for PCBs has decreased drastically over the last years with the technological advances made, but it still imposes a high cost and increases the time to market. Another important factor is debugging. Advanced multi-layer PCBs are challenging to debug and slows down the development and prototyping phase of product development. Time-to-market is an increasingly important factor when developing embedded systems, as the market is highly competitive and disruptive.

1.1 Background

1.1.1 Revolve NTNU

Revolve NTNU is a student organization dedicated to constructing electric formula cars for the international Formula Student competitions. In the course of one year, an electric vehicle (EV) is designed, constructed and tested before entering multiple competitions, facing off against comparable vehicles and teams from other universities from around the world.

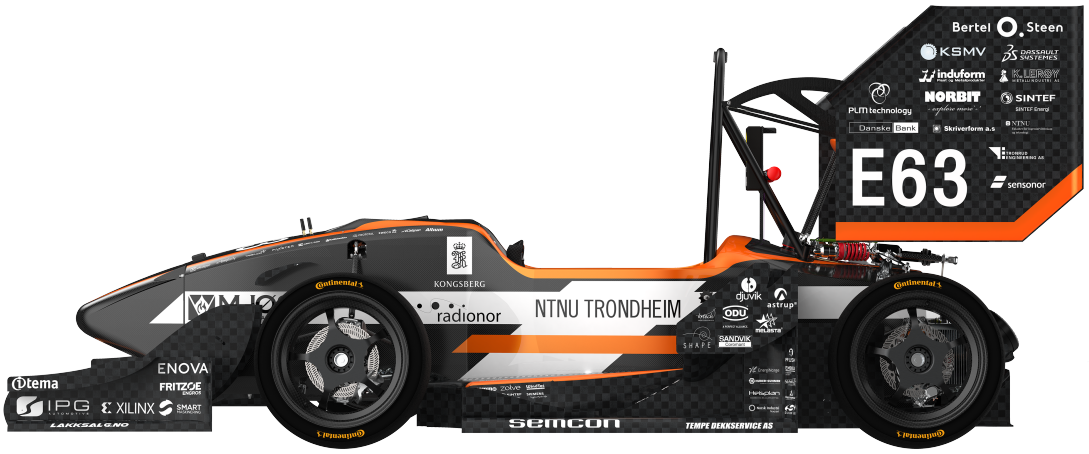

Figure 1: Revolve NTNU 2019's electric vehicle, Nova

The combination of short development time and a team consisting of mostly new, inexperienced members makes for several interresting problems. This report will focus on a specific embedded system, the vehicle control unit (VCU), and the issues encountered when developing it. As the EV is almost exclusively made from custom parts, the different systems on the car dependends heavily on each other. This means that both changes done the VCU will invariably affect the rest of the vehicle and issues with other systems will most certainly affect the VCU.

To get a clearer picture of necessary changes to the VCU, we start with the system requirements. analyze last year's VCU, VCU19.

1.1.2 The vehicle control unit

The vehicle control unit is the central control system of the electric vehicle. That is, it is responsible for inputting sensory data from the vehicle and the driver, and then output fitting data to electric motors connected to each wheel.

In addition, the VCU is to

- put the vehicle into Drive-Enable mode when certain criteria are met

- perform safety checks on inputted data and take appropriate action if errors are found
- interface with the inertial navigation system (INS)
- interface with the data logger provided by competition officials, as per EV 4.6 [2]

The team member responsible for the VCU is not responsible for the implementation of the control algorithms that takes vehicle and driver input and outputs motor set points. The control algorithms performs torque vectoring, a technique for intelligently varying the torque on each wheel (reference here), increasing the maneuverability of the EV. The VCU is responsible for providing an environment for the control algorithms to run.

Since the vehicle is capable of speeds exceeding 110 kilometres per hour [4], the control algorithms cannot use more time than necessary. As the deadlines the VCU has to uphold are critical for the survival of the vehicle and possibly the driver, it is per definition a *hard real-time embedded system*. However, the exact deadline is unknown. The rule of thmub used by members is that the control loop has to run in at no lower than 100 Hz. (Should really find some data on this.)

1.1.3 Analysis of VCU19

To fulfill the requirements of the control algortihm, VCU19 was designed around a new processing platform, the Xilinx Zynq-7000. It is a SoC with a dual core ARM Cortex A9 processor and embedded FPGA [3]. There were however issues with this implementation. Advanced processing units like the Zynq-7000 are commonly only available in ball grid array (BGA) packages, meaning packages with all pins placed in a grid pattern on the bottom. This makes hand soldering impossible, a reflow method has to be used instead. This makes assembly and debugging harder, a serious concern considering the rapid development methodologies necessary in Revolve NTNU. The high complexity of design also compelled the previous designer to opt for the smallest available package, a 225 pin BGA package. This limits the available peripherals, but was deemed a necessary compromise during the 2019 season.

1.2 Analysis of Nova

As the VCU is an integral part of the EV, issues regarding the vehicle are also issues for the VCU. This section is dedicated to some of the issues encountered during last season, which hopefully can be solved by improving the design of the VCU.

A major issue with the embedded systems on last years EV is the load on the CAN-FD buses. The vehicle is equipped with 2 buses which are shared between all embedded systems on the car. From the bus load analysis in Figure 2, it appears that the first bus has a peak of 80% load, and the second bus peaks at 70% when under load. The sudden jump in load on the second bus happens when the vehicle enters *drive enable mode*, i.e. when the VCU starts transmitting data to the inverters that drive the motors. This suggests that data going to and from the VCU accounts for much of the total load on the buses. This can be verified by examining the CAN message overview, see figure ...

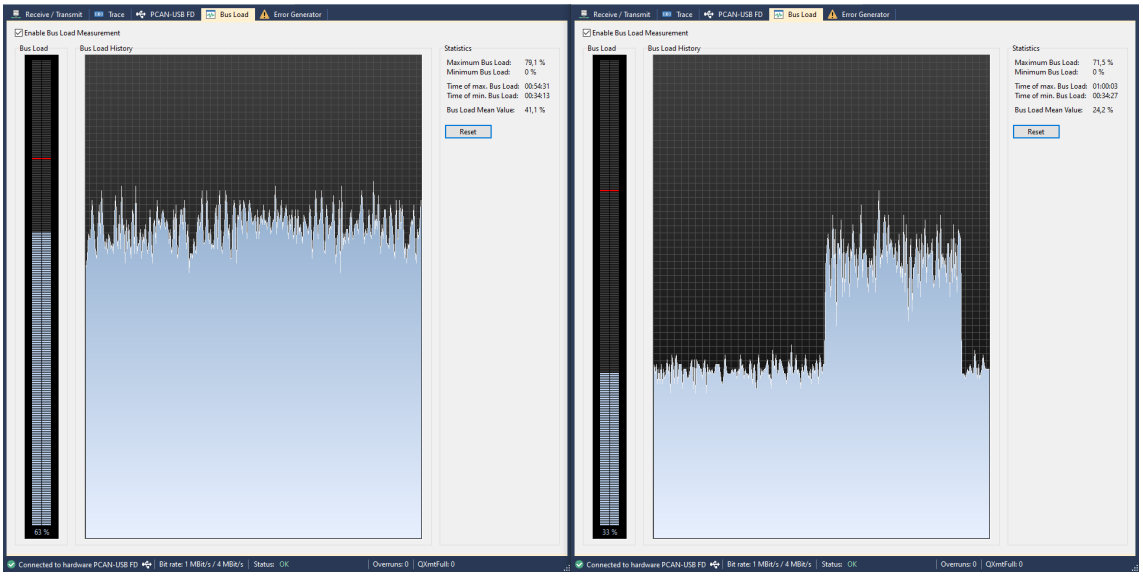

Figure 2: CAN-FD bus load

From the aforementioned numbers we can see that much of the data

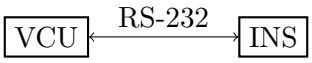

Figure 3: CAN-FD bus load

1.3 Scope

This report covers the design and testing of VCU20. The main focus of the new design is to reduce the complexity of the system while keeping the performance of a SoC. In addition we will take a closer look at the CAN-FD buses connecting the embedded systems on the vehicle.

2 Theory

2.1 Ball grid array PCB layout

Ball grid array (*BGA*) refers to a packaging type used for surface mount IC's. BGA packages feature all interconnects on the bottom of the package and allows for a higher number of connected pins than with pins on the edges of the package, see figure 4.

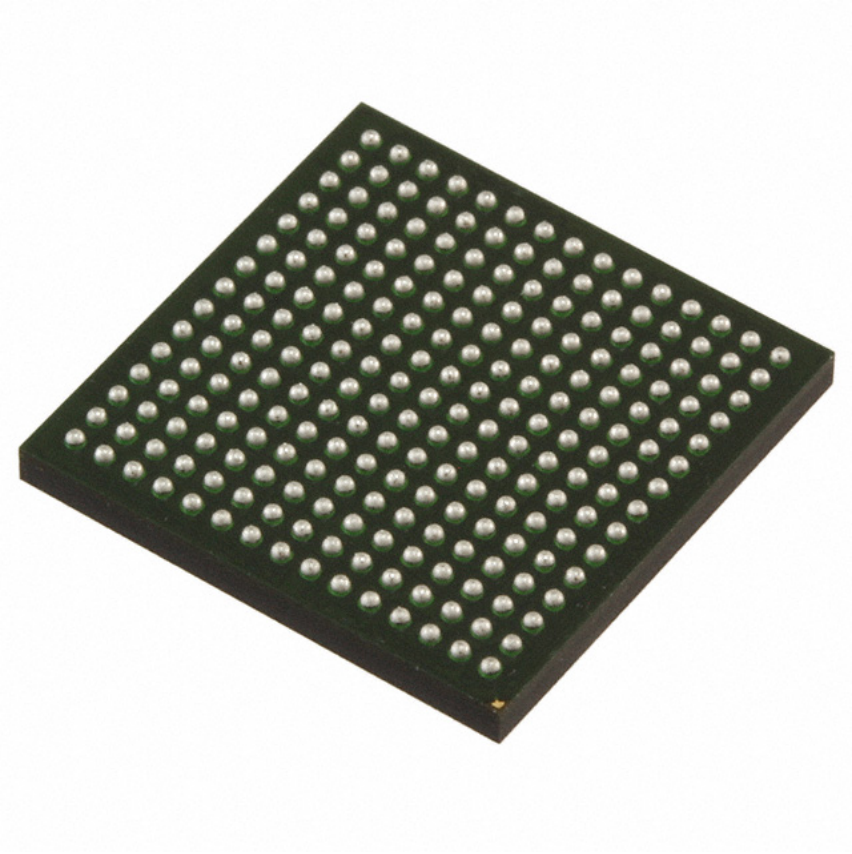

Figure 4: CLG255, a 15-by-15 pin BGA package used by the Zynq-7000 series SoC's [5]

As the pins are hidden when the package is mounted on a PCB, BGA packages are must be soldered by other methods than classical hand-soldering. Reflow soldering is a popular method for soldering BGA's where the entire package and adhering PCB are heated to the point where all the solder melts. This is typically performed in a dedicated reflow soldering oven, but can also be performed using a hot-air gun or similar.

2.2 CAN-FD

The different embedded systems on the vehicle communicate with each other using CAN-FD. CAN, or *controller area network* is a network bus developed by Bosch. It has been the de-facto standard communication platform for automotive applications since the mid-1990's. An improved version, CAN-FD (flexible data-rate) was released by Bosh in 2012, it differs from CAN in that the body of each message (called *frames* for CAN) are transmitted at a higher frequency than the header and the allowed frame size increases from 8 to 64 bytes. This is the communication bus currently used at Revolve NTNU.

CAN and CAN-FD is a one-to-many bus system where each frame is given an identifier which determines the priority of the frame. When the bus is free, all connected devices can begin transmitting their frame, but the frame with the highest priority will take precedence on the bus. When a frame is transmitted on the bus, any connected device are able to read it. When the frame has successfully been transmitted the bus is again free and a new

transmission can begin.

2.2.1 CAN-FD bus utilization

Almost all messages sent over the CAN-FD bus are sent at a regular interval. The size of the different messages are also known, this means that we know how long each message will occupy the bus for. These facts means that the frames on the bus can be modelled in the same way as tasks in a rate-monotonic scheduled system, known from real-time theory. This means the standard formula for utilization can be used when analyzing CAN and CAN-FD buses. The equation for utilization U on a bus is given in equation 1.

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \tag{1}$$

C_i is the execution time and T_i is the execution period for task i. In our case this is translated to the time frame i is active on the bus and the period at which it should be transmitted.

2.2.2 Maximum utilization

We can use the *Utilization-based schedulability test* to determine whether all deadlines are met for a set of tasks. The criteria can be seen in equation 2. It is important to note that this test is sufficient but not necessary. This means that a passing test guarantees all deadlines are met but a failing test does not guarantee missed deadlines.

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N\left(2^{\frac{1}{N}} - 1\right) \tag{2}$$

N denotes the number of different tasks being scheduled. Since the number of different messages on the CAN-FD buses is quite high, we might as well look at what happens to the criteria when N goes towards infinity.

$$\lim_{N\to\infty} N\left(2^{\frac{1}{N}} - 1\right) = \ln 2 \approx 0.693 \tag{3}$$

From equation 3 we can deem that as long as the bus load on each of the CAN-FD buses are below 69.3%, no messages are lost. This does of course not mean that the vehicle becomes unusable, figure 2 shows that the load was above the the limit for both buses, and the vehicle worked fine. The embedded systems are naturally made to be resilient to frame loss.

However, the limit serves as a good target for max bus load, and we will work towards lowering bus load to below this limit.

The high load on the CAN-FD buses poses one of the big challenges with the current system. We want a mathematical model of the load on the CAN-FD buses which can tell us whether messages will be lost or not.

2.2.3 Rate-Monotonic Scheduling

As the messages sent over the CAN-FD buses are sent at a somewhat regular interval, the buses can be modeled as a rate-monotonic scheduled (RMS) system. This is a way to model the scheduling of software tasks when the tasks are run at a set interval and with a fixed execution time. We begin by looking at the equation for utilization used with RMS.

Equation 1 shows how utilization is calculated for tasks, i.e. how much of the time that a task executes on a single core processor.

2.2.4 CAN-FD utilization

From the RMS utilization equation we know that we need the execution time C and period T for each task. While the period for each CAN-FD message is easy to determine, the execution time takes slighly more effort.From now on we will call the execution time for a message m the *transmission time* and we will denote it C_m. For a standard CAN message it would be sufficient to divide the number of bits in the message by the operating frequency of the bus. This is made slightly more difficult as CAN-FD operates on two different frequencies, one for header transmission and one for data transmission. We can split look at the total transmission time as a sum of the transmission time for each of the frequencies , see equation 4.

$$C_m = T_s + T_f \tag{4}$$

T_s, the transmission time for the message header is a constant number of bits, how it is calculated can be seen in equation 5.

$$\begin{aligned} T_s = \frac{(SOF + ID + r1 + IDE + EDL + r0 + \frac{BRS}{2} + \frac{CRCdel}{2}) \cdot 1.2}{t_x} \\ + \frac{ACK + DEL + EOF + IFS}{t_x} \end{aligned} \tag{5}$$

The different equations can be seen in equation 6.

$$T_f = \frac{(D_f + \frac{BRS}{2} + ESI + DLC + \frac{CRCdel}{2}) \cdot 1.2 + CRC + BS}{t_y} \tag{6}$$

Because of error detection mechanisms defined in the CAN-FD protocol, the transmission time for payloads larger than 16 bytes is different from payloads smaller than 16 bytes. For payloads smaller than 16 bytes the Cyclic Redundancy Check (CRC) is 17 bits and

the Bit Stuffing (BS) is 5 bits. For payloads larger than 16 bits, CRC rises to 21 bits and BS to 6 bits.

$$U = \sum_m \frac{C_m}{T_m} \tag{7}$$

We will also

To make reflected decisions about the CAN-FD buses on the vehicle, we need mathematical tools to look at the worst case bus load. Equation 7 shows the equation we will use to determine the utilization, or load, U on the CAN-FD bus. C_m is the worst case transmission time (WCTT) for a message m while T_m is the period of message m, i.e. how often it is transmitted. It is of course necessary to compute the load from each of the different messages on the bus to be able to calculate the load.

WCTT is pretty straight forward to calculate, but it needs some explanation. What separates CAN-FD from CAN is flexible data-rate (FD). This means that the header or metadata of each frame is transmitted at a lower frequency, just like for CAN, but the payload of the frame is transmitted at a higher frequency, here called t_y, while the slower transmission rate is called t_x. The total WCTT for a message can be seen as a sum of the worst case time for the header transmission and the higher speed payload transmission, see equation 4.

2.3 Differential signalling

When dealing with high-speed, low-power signals in electronic systems, minimizing noise is of the utmost importance. One technique for dealing with this is *differential signalling*. Simply put, it means driving two conductors instead of just one. Single-ended signalling, the conventional method, means that signalling lines are referenced to a common ground. In differential signalling, two lines are used for each signal line. These lines are referenced to each other instead of to a ground shared by all signals. This serves to remove noise that might be present on the ground plane and, when the two conductors are placed close to each other, excellent protection against external interference. When the lines are close to each other, electromagneting fields that normally would introduce noise to the signal will optimally affect both conductors by the same amount, and since we only look at the difference between the two lines, the noise is elegantly subtracted away.

3 Method

3.1 System-on-Module

As the the Zynq-7000 platform provides a sufficient platform for running the VCU control loop and the TV algorithm, but at the cost of high design complexity. By utilizing a Zynq-7000 based System-on-Module (SoM), we avoid the hard parts of PCB layout, and simply design the VCU as a *breakout board* for the SoM. Using a SoM also gives us access to

peripherals that would have been difficult to implement by ourselves, specifically Ethernet, external flash memory and external double data-rate (DDR) memory.

3.2 Ethernet for telemetry

It is necessary to retrieve data on the vehicle and the different systems on it during races. This is both for later analysis and to indicate to the team if there might be issues with the vehicle before things go wrong. To achieve this, Nova utilized a military grade wireless communication solution by RadioNor, the CRE2-144-LW [1]. UDP is used to interface with the radio and the physical layer is Ethernet. As no other system on Nova was equipped with Ethernet, the solution was to place a Raspberry Pi 3 B+ in the vehicle, and connect it to the CAN-FD buses with two PCAN-FD USB dongles. The Raspberry Pi would simply gather all available data from the CAN-FD buses and transmit it to the radio over Ethernet.

As the SoM chosen for VCU20 is equipped with an Ethernet-PHY interface, it is an opportunity to greatly simplify the telemetry system. By connecting the VCU directly to the radio directly using Ethernet, the total weight and complexity of the vehicle can be reduced. The downside to this is that the complexity is moved to the VCU, as it now has to communicate with the radio in addition to everything else.

3.3 Partitioning CAN-FD buses

As we want to reduce the load on the CAN-FD buses, we should look closer into ways to partitioning the communication channels in a different way, even add communication channels where that might be applicable.

The inverter is heavily dependent on the VCU, but not many other systems. Having a dedicated communication channel between these two systems could potentially reduce the load on the CAN bus by a lot.

3.4 Hardware design process

As mentioned in the introduction, the team has to design and test hardware within a very short time.

By reusing software and hardware design from last year's design, issues were reduced to a minimum.

The electronic design automation *(EDA)* software used for schematics and pcb layout, *Altium Nexus* supports multiple features which shortened the design process significantly. Firstly, since VCU19 used *hierarchical design* (i.e. project is composed of multiple sheets and sub sheets), large parts of the schematics developed during last years design period could be reused.

Simple sub circuits like CAN-FD transceivers could be directly copied without issue. In addition, the sub sheets are modular in the sense that they appear as an electrical compo-

nent with a specific set of interfaces, or *ports*. This means that sheet using the same set of ports can be interchanged freely. The modularity of the sheets was heavily used during the development of VCU20. The new processing unit features a super set of the VCU19 SoC interface, meaning they are perfectly compatible.

There is also the possibility of reusing sheets in the same project. VCU20 has 4 CAN-FD transceivers, but since the transceiver schematic is identical for all 4 interfaces, the sheet can simply be repeated.

There are also features for reuse in the PCB layout part of the design. Altium Nexus has a feature called *rooms* which is a grouping of component footprints and how they are connected. It is possible to copy the layout if a room to other rooms. The designer chooses how rooms are generated, and by default they are created per sheet. This means that the layout for one CAN-FD transceiver can be perfected and then copied over to the other 3 transceiver rooms.

4 Results

4.1 CAN-FD bus partitioning

Calculations on bus load for 2 and 3 buses

4.2 CAN-FD interface performance

Benchmark of CAN-FD speed, make sure it performs as well as necessary (from calculations)

4.3 Ethernet speed

Benchmark Ethernet interface bandwidth. Calculations showing wether it is able to transmit all data from CAN-FD buses over telemetry

5 Discussion

Reflect on results.

References

[1] Redionor Communications AS. *CRE-144-LW product information page*. URL: `https://radionor.no/product/cre2-144-lw/`. (accessed: 05.12.2019).

[2] Formula Student Germany. *FSG rules 2020*. URL: https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf. (accessed: 05.12.2019).

[3] Xilinx Inc. *Zynq-7000 product brief*. URL: https://www.xilinx.com/support/documentation/product-briefs/zynq-7000-product-brief.pdf. (accessed: 04.12.2019.

[4] *Revolve NTNU 2019 Nova information page*. URL: https://www.revolve.no/electric-vehicle/. (accessed: 04.12.2019).

[5] *Xilinx CLG225 BGA package*. URL: https://media.digikey.com/Photos/Xilinx%20Photos/122;225CSBGA-1.5-13x13;CLG;225.JPG. (accessed: 09.12.2019).

Appendix

Layout

Schematics

CAN-FD calculations

Benchmarks