1 Assignment Description

Note: Check regulary for updates on this document!
The design project of this semester's Embedded System Lab makes extensions to your previously developed hardware and software.
Your task is to develop a client in a distributed system. The systems main function is to synchronize a set of data between all clients, giving each client the option of modifying said data (henceforth called 'the data'). Each development board takes on the role of a client in this system.
To indicate the state of each client, the data is to be displayed on the LCD display. The individual data shall be manipulated using an input device, i.e., a keyboard. For synchronization, a simple CAN based protocol is given to exchange data.

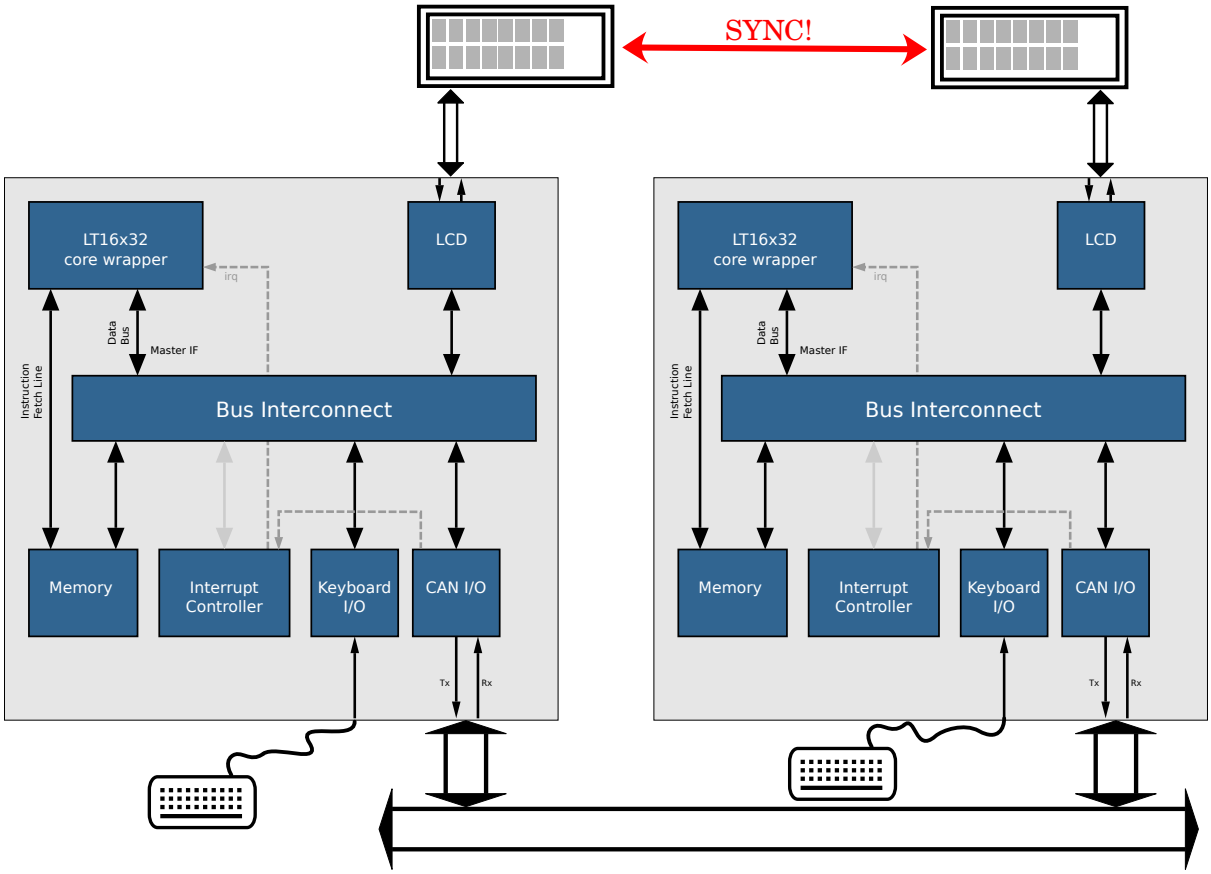

Figure 1

2 The 'Project' Aspect

This this a development project. Plan the architecture of your system ahead of time. Identify tasks and distribute them among group members. Make a schedule for the tasks with the allotted times for each task and for integration, testing and debugging.
You will present your plan during the review meetings. Refer to the slides from the introductory meeting for guidelines on the presentation. (You need to make an appointment for the review meeting.)

3 Development Target

As previously stated, this project is about developing a client in a distributed system. Each group shall develop and implement one client, instantiating it on the development board.
Each client has a set of data that consists of 16 individual bytes. These are to be displayed on the LCD display. The data is shared among all clients in the system. For the purposes of this project, the data contained in the set are ASCII characters (One per data byte).
Clients can make changes to the data set with their input devices. In this case, this applies to the keyboard. When a client changes the data, it needs to synchronize the change with all other clients in the system. This is to be done utilizing the CAN bus and the CAN bus controller which is to be made part of the system. Details on the CAN controller and setup are given in Section 4. The specific frame used for this is explained in depth in Section 5.

4 The CAN Controller

The lab's git repository includes the hardware design for a CAN controller. This controller's interface is functionally compatible with the *SJA1000* (and *PCA82C200*). It is recommended to only use the basic transmit features, and abstain from using the PeliCAN mode.
The controller is written in Verilog, and has an 8-bit Wishbone interface. A hardware wrapper in VHDL is provided (`<project root>/soc/peripherals/can_vhdl_top.vhd`) to connect the device to the rest of the system. The wrapper does not extend its data interface beyond 8 bit, so only byte-sized accesses on the device are permitted.
The configuration vectors (the generics `memaddr` and `addrmask`) need to be supplied. Also, you should add the component declaration of the VHDL wrapper to the `lt16soc_peripherals` package.

4.1 The CAN-Transceiver Pmod

To physically interact with the CAN network, an electrical adapter is needed. This adapter chip implements the physical layer of the CAN protocol. It consists of an RS485 driver on a separate circuit board that has a convenient connector adapter for the board. These are called "Pmod" adapter- or I/O-boards.
The teaching assistants and technician will only hand out these Pmod after a successful simulation has been demonstrated (See Sec. 6). Be careful with them, as mishandling can damage both the Pmod as well as the FPGA board! The documentation of the Pmod is a available in the `documentation` directory in the repository.
The converter-circuit-board can be connected to any of the Pmod connectors on the FPGA board. Add appropriate I/O-signals to your top-level entity. Depending on which connector you choose, you have to map connector data pins to ports in your top level entity. For this purpose, edit your `.ucf` file by adding the chosen connectors pins and mapping them to signals. Consult page 27 (Listing "Pmod Connector Pinouts") in the Genesys data sheet for the correct pins names.
As noted in the data sheet, the maximum bit rate is 16 Mbit per second. It is recommended to chose a baud rate far smaller than the maximum.

5 The Data Protocol

Each time a client changes a byte in the data set, it needs to notify the other clients in the system.
To achieve this, each change should trigger the sending of a CAN frame, while simultaneously listening the CAN bus for update frames.
Update frames are normal CAN data frames. The ID of a update frame consists of a 6 bit fixed update code `"000001"`, and a 5 bit number that is unique to the client in the system. The data field consists of 2 byte, a selection byte, which identifies the data byte which is updated, and the updated byte value. This data frame is schematically illustrated in Figure 2

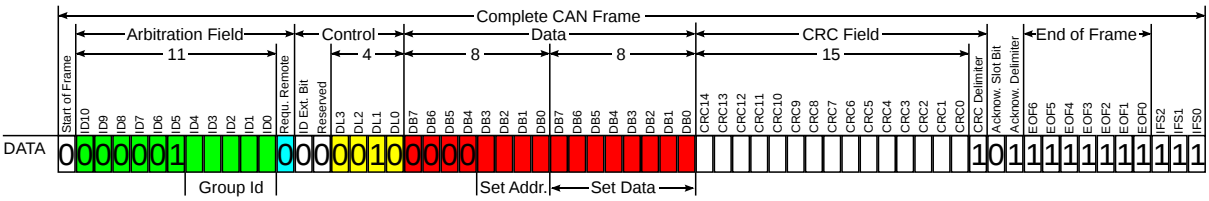

Figure 2: The “update”-frame illustrated

Upon receiving an update frame, the client should update the selected data set byte with the updated byte value. The data selection address ranges from 0 to 15, selecting the bytes in the order they are displayed on the clients LCD display.
This is the only kind of frame your client needs to support for this assignment. However, your client should check and validate the values it is receiving.

6 Testing

For successful testing it is necessary to simulate a system with multiple clients.
For this purpose, several platforms need to be instantiated, and their CAN connectors properly connected. As for the software, it is recommended to write up mock-software which does not utilize the LCD.

7 Initialization Problem

The system is asynchronous and clients may be added or removed from the system at any time. This leads to the problem that a new client does not have synchronized copy of the data set.
Think about possible solutions to this problem. Come up with a possible extension of the previously established protocol, and present it in the Review Meeting (See Sec. 8.1). You do not need to implement a solution.

8 Completion

In the beginning of the Design Project, you will need to develop an overview of the task and present a project plan in the Review Meeting. Finally, you need to demonstrate your implementation.
A working demonstration should include more than one client. To achieve this, additional FPGA boards can be provided to be used with your own implementation. It is recommended and to connect clients from different groups into one system. Additionally, the teaching assistants may introduce a “control client” (our implementation) to be connected to the system.

8.1 Review Meeting

In the Review meeting, you need to present your plan in a way that conveys that your understanding of the given task is sufficient, and present your plan on how to proceed.
Among the mandatory items your project plan needs to incorporate are

- hardware-software-partitioning
- identification/partitioning into individual tasks
- Time line for implementation and testing
- distribution of the individual tasks among group members

8.2 Demonstration

The successful implementation needs to be demonstrated in a sample system with multiple clients to teaching assistant or supervisor. Only one client in the system will run your implementation.